GENERAL™ User Guide

Report Writer Version 6.0

GENERAL is published under license by:
Synergetic Data Systems, Inc.
2195 Talon Drive
Latrobe, CA 95682
USA

Email: sdsi@synergetic-data.com
Web page: http://synergetic-data.com

The GENERAL™ Report Writer License Agreement

NOTICE: OPENING THIS PACKAGE INDICATES YOUR ACCEPTANCE OF THE FOLLOWING TERMS AND CONDITIONS. PLEASE READ THEM. IF YOU DO NOT AGREE WITH THEM, RETURN THE PACKAGE UNOPENED, AND RETURN OR DESTROY ANY COPIES OF THE PROGRAM IN YOUR POSSESSION. THE DEALER FROM WHOM YOU PURCHASED THE SOFTWARE WILL REFUND YOUR PURCHASE PRICE.

"Program", as used herein, refers to both this documentation and the software programs described by this documentation. "Developer", as used herein, refers to Allen D. Miglore.

LICENSE
You may use the Program on a single machine, and you may copy the Program into any machine-readable format for backup purposes only. If you transfer the Program to another machine, you agree to destroy the Program, together with all copies, in whole or in part, on the original machine. You may not copy, modify, or transfer the Program, in whole or in part, except as expressly provided herein. You may not sublicense, assign, or otherwise transfer the Program to any third party except by the express written consent of the Developer.

TERM

The license is effective until terminated. You may terminate at any time by destroying the Program together with all copies of the Program in your possession. It will also terminate automatically upon failure to comply with any of the terms of this agreement. You agree upon such termination to destroy the Program together with all copies in your possession in any form.

CONFIDENTIALITY OF THE PROGRAM

You understand that the Program is proprietary to the Developer, and agree to maintain the confidentiality of the Program. You agree that neither you, nor any person or entity acting on your behalf, will copy or otherwise transfer the Program, in whole or in part, in any form (including printed source code), to any third party. You agree to retain the Developer's copyright notices, in all forms, throughout the Program.

LIMITATION OF LIABILITY

The Program is provided "as is" without warranty of any kind, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the Program is with you. In no event will the Developer be liable to you for any damages, including any lost profits or other incidental or consequential damages arising out of the use or inability to use the Program, even if advised of the possibility of such damages.

SUPPORT

Support for the Program should be obtained from the Dealer from whom it was purchased. The Dealer, not the Developer, establishes support pricing and terms.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND THE DEVELOPER AND IT SUPERSEDES ANY PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATION BETWEEN YOU AND THE DEVELOPER RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

GENERAL is published under license by Synergetic Data Systems, Inc. (SDSI), 2195 Talon Drive, Latrobe, California 95682 USA.

View SDSI's web site at http://synergetic-data.com or e-mail SDSI at sdsi@synergetic-data.com

INTRODUCTION

GENERAL is a powerful report writing and data analysis tool for applications based on the Business Basic language. There are several implementations of this language which GENERAL is compatible with, and together they comprise support for many hundreds of thousands of commercial business installations worldwide.

Business Basic is a programming language that is well suited to business application development, and there are many business-related software products written for this language.

Traditionally, users of Business Basic applications were limited as to which reports could be printed from their applications. Only those reports that had been pre-programmed could be generated. Over time, several report writers have been developed to eliminate this restriction. GENERAL is the most widely used of these, and with this latest version, it is the most powerful and flexible one available.

GENERAL provides many capabilities, which are described in the paragraphs below.

Report Writing

GENERAL provides three modes for report writing, each designed to appeal to a specific type of user.

Interactive Mode, or *PROMPT mode*, is a full-screen menu-driven mode that leads the user through each step of producing a report. To select and place the data to put on the report, the user enters information into a table, or optionally can "paint" the information onto a visual image of the report line. If calculations or inter-file links are required, they can be entered via a function key, and embedded text can be either entered via a function key or can be painted in the proper position.

Sorting, break points, selection criteria, and custom headers and footers can also be defined, and the report may be executed at will.

Direct Entry Mode, or *command mode*, is the fastest method for experienced users. While the other two methods are simply shells around GENERAL's query language, direct entry is the real thing. For short, quick reports direct entry is the favorite of experts. While in direct entry mode, a function key toggle enables the assist mode, so even non-experienced users should feel comfortable with this mode.

Assist Mode, toggled by a function key when in direct entry mode entry, provides the user with split-screen pick and point of fields and keywords. When keywords requiring parameters are selected, GENERAL prompts the user for those parameters. Assist mode requires a basic knowledge of GENERAL's LIST command syntax, but helps the user by providing field names and keywords on-screen.

Report Management

In addition to report writing, GENERAL provides additional tools for report management.

Report History
The STACK command provides access to the previous 18 LIST commands entered (by any of the report writing modes). Each of these reports can be re-executed, edited, or saved permanently.

Stored Reports
Any report that has been saved from the report history can be re-executed or edited with the RUN command. With a technique called "run-time prompting", the same report can be used for different circumstances, different selection criteria, different export type, and so on.

Analysis

GENERAL is an excellent tool for data analysis. The data collected for reports is "live" data, right out of the application GENERAL is linked to. The user has a great deal of control over how the data is presented:

- Data can be placed at any column and line position.
- Sorting can be based on as many characters as the underlying Basic allows, split into any number of segments, each either ascending or descending.
- Selection criteria is unlimited, with Boolean control (AND and OR), and parenthetical control.
- Data can be exported into a variety of formats for further analysis, charting, mail merges, and more.

In addition to data presentation, GENERAL provides five aggregate keywords, to produce column totals, averages, counts, minimums, and maximums. When used in conjunction with break points, automatic subtotal generation is also provided.

If the automatic sub-footers and report footers aren't what the user is looking for, GENERAL also supports the creation of custom headers and footers, complete with their own calculations and data positioning control.

Lastly, GENERAL supports a cross-tabulation feature that can be used to produce summaries of data printed in spreadsheet fashion.

Dictionary Support

As Business Basic has no intrinsic method of defining the data stored in its files, GENERAL must be told where to find each element of data that is required for a report. This definition of where to find the data is called a *dictionary*.

In many cases, the data may be stored in a manner that is unsuitable for a report. A date, for example, might be stored simply as a number, such as 2,448,501, instead of a readable date, like 9/1/91. Part of GENERAL's dictionary includes an ability to describe how to decipher the physical data into meaningful data.

In addition, the dictionary can be used to describe calculations and inter-file lookups that may be used frequently, so that they don't need to be entered in each report that will need them. For example, a PROFIT dictionary element could be defined that was a simple calculation of SALES - COST.

GENERAL also has the ability to use some external dictionary formats. Specifically, they are SDSI's Filix dictionary and Basis International's Basis Data Dictionary. Over time, additional external formats will probably be supported. When a report is produced, GENERAL looks for the file specified for that report first in its own dictionary, then in Filix, then in the Basis Dictionary.

When using an external dictionary, calculations and inter-file links will generally have to be defined in each report, as external formats typically only define physical data, and not calculations.

Security

GENERAL provides security through a password-controlled login. The login names are associated with an *access level*, which is a number from 0 to 9. Files are also assigned access levels from 0 to 9, and GENERAL ensures that a user's access level is at least as high as the file's access level before allowing reports to be generated from that file. An administrator-level user (level 9) can assign login names and optionally passwords to other GENERAL users, and can also maintain the file dictionary, where file access levels are specified.

In addition to file-level security, General 6 supports report-level security. Each Prompt and Run report design has a Run level and Modify level. For a user to be able to run a report, the user access level must be at least as high as the Run level. For a user to be able to modify the report design, the user access level must be at least as high as the Modify level.

Run-time Replacements

One of the key features of GENERAL is the ability to control the contents of many elements, such as file names, expression constants, and selection criteria, at run-time. The operation behind this ability is called *run-time replacements*. Virtually anywhere in GENERAL where a value is required, a run-time replacement can be used instead.

Imagine a report that selects just customers whose sales have been over $100,000 this year. The selection criteria for this report would be YTD.SLS> 100000. If the same report is required at another time, but for customers whose sales have been over $500,000, this report definition would have to be modified, or another report defined.

With a run-time replacement, the selection criteria could be changed to:

YTD.SLS > [[Enter minimum year-to-date sales]]

GENERAL will issue a prompt to the user when the report is run, and use the answer in the selection criteria.

Traditional and Graphical Interfaces

The traditional mode of operating General is with character terminals or terminal emulators. These screens provide an efficient point-and-click interface, which makes extensive use of cursor motion keys and function keys. Many users are familiar with, and prefer, this type of interface, so Version 6 retains it and in fact adds several enhancements to it.

In addition to this interface, Version 6 adds a graphical user interface (GUI), which can be run on Windows systems. These systems operate as clients, accessing General on a server via a graphical server daemon (background task). When running reports with the GUI client, the report generation is performed on the server by a background task, which creates reports and returns them to the client for viewing or printing. Reports are generated by the same engine in both traditional and GUI modes.

Most of the functions available to the traditional user are also available to the graphical user. The screens used provide similar, if not identical, options. In some cases, the graphical user has more capability, particularly with regard to report viewing and in managing reports from the main menu. However, there are some configuration functions that can be accessed only through the traditional interface. One key function currently only offered in the traditional interface is dictionary maintenance.

The GUI interface provides context-sensitive help for all screens. The documentation provided here is primarily oriented to the traditional user, but information that doesn't pertain exclusively to the use of the traditional interface will be of use to a GUI user as well.

Features Only Supported in Traditional Mode

While all report writing and viewing features are supported in both traditional and GUI mode, some administrative features are supported only in traditional mode. These include:

- Dictionary Maintenance
- Configure Options
    - Terminal attributes
    - Printer definitions
    - Report heading defaults
    - VDT options (this video)
    - User function definitions
    - User replacement definitions
    - Import secondary dictionary
    - Export secondary dictionary
    - Report menu maintenance

Future versions of General may include GUI support for some of these features.

Traditional Mode Keyboard Navigation

The traditional character mode version of General supports a point-and-click, and function key oriented interface. You can use arrow keys and page keys to move your selection or field entry cursor. At many points, function keys are available, and are always identified on the bottom of the screen. The use of function keys is generally consistent throughout. For example, F2 is generally used for popup selection lists, and F10 is always used for exiting a screen.

Pop-up Menus

Pop-up menus may appear at times throughout GENERAL. Whenever a list of items and associated descriptions or other information is needed, a pop-up menu is used. Sometimes, these menus have column headings, as in GENERAL's main menu. Note that the two numbers in the lower right corner of the menu, *x/y*, can be read as "page *x* of *y*".

Items are selected from a pop up menu by highlighting the desired choice and pressing <Enter>. There are several methods for highlighting a desired option.

- Use the up/down cursor motion keys, or page up/page down to move between multiple pages.
- Press the first letter of the item desired. This will highlight the first item starting with the letter. Subsequent presses of that letter will highlight each item starting with that letter, in series.
- Press the number associated with the item desired.

If you press the **F10-exit** key, no selection is taken, and GENERAL will re-issue the original prompt or step back in the menu levels.

Ring Menus - Verification Prompts

Ring menus are lists of single word selections presented on a line at the bottom of the screen. All verification prompts are presented in ring menus. When first presented, the cursor is positioned at the first option, so it acts as the default. There are two methods of selecting an option.

- Move the cursor to highlight the desired option, and press <Enter>. To move the cursor, use the left- and right-arrow keys. The <Tab> or <Spacebar> keys simulate a right-arrow, and a <Backspace> simulates a left arrow.
- Press the highlight letter (usually the first letter) of the option desired. GENERAL immediately takes the option.

If you press the **F10-exit** key, then no selection is taken. This is useful in file verification prompts, where you want to abort any changes made, so they are not permanently stored.

Text Entry and Editing

Description of Movement	Standard Keystroke	Alternate Keystroke
Move character left	Left arrow	^H (control-H)
Move character right	Right arrow	^L
Move up one line	Up arrow	^K
Move down one line	Down arrow	^J
Move to end of line	End	^Z
Move to beginning of line	Home	^A
Move to next page	Page down	^Y
Move to previous page	Page up	^U
Toggle Insert/Replace mode	Insert	^T
Delete character	Delete	^X
Clear to end of line		^W
Delete line (clears to end of line first)		^D
Insert line		^N
Function keys	F1..F10	Escape,1.. Escape,0
Calculator/Calendar		^C

Keystroke Editing

At any entry point in GENERAL, keystroke editing is available for the entry of text. In some cases, such as in LIST command entry, additional keystrokes like the insert-line and delete-line functions are available. As keyboards vary from one terminal type to another, standard PC-style keyboards are not always available, and GENERAL provides alternate keystrokes to perform the functions noted above.

The alternate keystrokes identified above are the defaults supplied with GENERAL.
Note that for the most part, only the "Standard" keystrokes work in ProvideX. If these keys don't work on your terminal, you need to run the keyboard mapping utility *uck.

The calculator/calendar option can be used at most entry points, unless Control-C is your interrupt character. If the cursor is in a numeric field, the calculator may be used to insert a numeric value; in a date field, the calendar can insert a date.

Non-printable Characters

Enter non-printable characters into a command, such as a <tab> or <bell>, by using GENERAL's ASCII notation. ASCII notation is a 4-character code, starting with a tilde (~), followed by three digits.

~009 is ASCII value 9 (a <tab> character).
~013 is ASCII value 13 (a carriage return).

Quoting

Literal values in reports must be quoted. By quoting, GENERAL is told to ignore spaces and other special characters, and treat the quoted value as a whole. Quoted text simply is surrounded by quotes: **"This is quoted text"**.

If the text to be quoted contains quotes, then use single quotes: **'This text contains "quotes inside"'**.

On-line Help

GENERAL provides on-line help at almost any entry point. This is accessed from the **F1-help** key. This keystroke invokes a display of help for that particular entry. In addition, from the help ring menu, both keyboard help and additional topics can be displayed.

Help displays can occupy more than one screen. In those cases, the Page option (or page-up and -down keystrokes) can be used to scroll through the screens.

INSTALLATION AND CONFIGURATION

General Version 6.0 is composed of three elements:

- Traditional character mode programs that run on a server
- A server daemon that runs on a server
- A Windows graphical (GUI) client that runs on Windows computers on your network

When you install the server software, the first two elements are installed. This should be performed on the system that contains the Business Basic (PRO/5, BBj, or ProvideX) data files. There will be a primary directory and two sub-directories, such as this:

/usr/lib/sdsi/gen60
/usr/lib/sdsi/gen60/gen60_bb
/usr/lib/sdsi/gen60/gen60_pv

The top-level directory contains the server daemon software, while the gen60_bb and gen60_pv subdirectories contain the character mode report writer programs for BBx (PRO/5 and BBj) and ProvideX, respectively. It is possible to move the character mode programs to a different directory, but if you do so, you will need to configure the search prefix for the server daemon. See the section "Configuring the Graphical Server" for details.

If you are upgrading from a previous release, you will have a GEN4MST or gen5mst file that contains your dictionary and report specifications. Before trying to run General 6, you should copy that file to the gen60_bb or gen60_pv subdirectory. The first time you run gen6, that file will be converted automatically to Version 6 format. This can be done later, by removing or renaming the default gen6mst file and copying your old dictionary to the appropriate subdirectory. However, any reports or dictionaries added to the default gen6mst file will be lost.

The graphical client software may be installed on any Windows 98 or higher computer on your network. Graphical clients do not require a license or activation key. Instead, the server daemon is licensed for a set number of concurrent connections, based on client IP addresses.

Unix Server Installation from CD

1. Login as root.

2. Mount the CD as a file system that supports lowercase file names. If you are unsure how to do this, check your man pages: **man mount**. The following table illustrates sample mount commands for various operating systems, assuming the mount directory, /mnt, is available, and standard CD device names. You may need to adjust these commands according to your configuration.

SCO UNIX OS5	mount –o lower /dev/cd0 /mnt
SCO UNIX	mount –r –f HS,lower /dev/cd0 /mnt
UNIXware	mount –F cdfs –r /dev/cdrom/cdrom1 /mnt
AIX	mount –vcdrfs –r /dev/cd0 /mnt
Sun Solaris	mount –rt hsfs /dev/sr0 /mnt
HP/UX	mount –r –F cdfs –o cdcase /dev/dsk/c1d0s2 /mnt

3. Change to the General 6.0 Unix directory in the mount directory: **cd /mnt/gen60/unix**

4. Run the install script: **./install.sh**, or if you do not have execute permission to the file, **sh install.sh**. Follow the prompts to select a directory for Gen 6, and choose if you will be supplying your own run-time PRO/5, BBj, or PVX engine, or installing a bundled version that includes a PRO/5 engine. If you install a bundled version, you will need to select the version that is appropriate for your operating system.

5. General 6 will then be installed by copying files to the selected directory, and executing the set up script ./setup.sh in the General directory. Once General has been installed from the CD, ./setup.sh can be executed at any time from the install directory.

6. Use the **gen60d –v** command to ensure General's server daemon is installed and set up correctly. The output from this command will display the serial number used and state that this is a demo version. The character mode product for BBx or ProvideX will be in the gen60_bb or gen60_pv subdirectories, respectively.

7. See the **Licensing** section for activation instructions.

Unix Server Installation from download

1. Login as root.

2. Create a directory to hold the General files, and change to that directory.

   Example: **umask 0**
   mkdir /usr/lib/sdsi/gen60
   cd /usr/lib/sdsi/gen60

3. Uncompress and extract General from the download file:

   uncompress gen60_xxx_tar.Z
   tar xvf gen60_xxx_tar

4. Execute the setup.sh script, which will normally ask you where the PRO/5, BBj, or ProvideX executable is located on your system. If you have a version of General that includes a run-time, then this question won't be asked.

   ./setup.sh

   If you see the question "What directory contains PRO5, BBj, or PVX?", then you must answer with the correct directory, or the installation will not complete, and you will have to re-run this step again when you know the correct directory information.

   Once a valid directory has been entered, the setup.sh script will create a script called "/usr/bin/gen60d". This is the server daemon start/stop script.

5. Use **gen60d –v** command to ensure the server daemon is installed and set up correctly. The output from this command will display the serial number used and state that this is a demo version. The character mode product for BBx or ProvideX will be in the gen60_bb or gen60_pv subdirectories, respectively.

6. See the **Licensing** section for activation instructions.

Microsoft Windows Installation Instructions:

1. From the CD, use explorer to locate the D:\gen60\win directory (or D:\gen60\winbun for a bundled install that includes a run-time), and double-click the setup.exe program. If you downloaded General from the Internet, simply execute the downloaded executable. Follow the on-screen prompts from the installer to install General to your system.

2. Click the Start, Programs, General 6.0 Server, **Gen60 Server Setup** option. This will conditionally rename certain files and prompt for where your run-time Business Basic is located. The full path to the executable must be given. This value will be stored in gen60d.ini.

3. Use the **Gen60 Server Information option** to ensure the server daemon is installed and set up correctly. The output from this command will display the serial number used and state that this is a demo version.

4. See the **Licensing** section for activation instructions.

Windows Client Installation

From the CD, use explorer to locate the D:\gen60\client directory and double-click the setup.exe program. If you downloaded the General client from the Internet, simply execute the downloaded executable. Follow the on-screen prompts from the installer to install the General client software to your system.

Operating the Graphical Server

On Unix, you can use the following commands with the graphical server:

gen60d start
gen60d stop
gen60d –v
gen60d –act

The start option starts the server. You should see a message on the terminal indicating that the server is listening on a port. Once started, graphical clients can connect to this server. If this is a bundled installation, the server will check to see if the license manager is running for the bundled runtime. If not, it will be started automatically.

The stop option stops the server.

The –v option displays version information.

The –act option will prompt for the activation key for the graphical server license.

On Windows, use the Start, Programs, General 6.0 Server, Gen60 Server Status option. This provides a window that displays the log of the currently running server. It also provides Start and Stop buttons to start and stop the server. These buttons will be disabled if the server is installed as a Windows Service, and manual stopping and starting must be performed via the Control Panel Services applet.

In addition to the Server Status option, you can use the Server Information option to display version information, and the Server Activation option to enter an activation key for the server.

For information about accessing traditional character mode, please see the Accessing General chapter.

Configuring the Graphical Server

Configuring the graphical server requires editing of some files and possibly some programs. The server reads the file gen60d.ini on startup, and on PRO/5 or BBj, uses the config.gen file as a "config.bbx" configuration file. These two files can be edited with any text editor. Here is a sample of gen60d.ini:

```
[defaults]
# values here can be overridden from the command line
port=8414
logfile=gen60d.log
logdetail=0
timeout=3600

[security]
# allow=list (;delimited) of IP addresses
# addresses may contain wildcards, like 1.1.1.*
# optionally, any=any address, filename containing ips
# procports allows a set list of ports to be used for secondary
# connections.  If commented out, random ports will be used.
# hideconn=0|1 on Windows controls visibility of process windows
allow=192.168.*.*; 10.*.*.*
procports=9000,9001,9002-9100
hideconn=1

[param]
caption=Company
call=genparam.bb

[params]
01=Acme Distribution
02=Owner's Fat Cat Company
```

port	The TCP/IP port that the server listens on for connections from graphical clients. On the login screen of the client, the user selects a server and port, which must be the server system and the port identified here. The default port is 8414.
logfile	The name of the log file that records connections. This file will be in the General server directory unless a full path is provided.
logdetail	Should be 0 to record just connections, or 1 to record details of client-server transactions, which should only be used for debugging purposes. If you specify 1, the log file can grow very large very quickly.
timeout	The time in seconds that the server will wait before an idle connection is terminated.
allow	A list of IP addresses or wildcards that are allowed to access the server. Multiple items can be separated by semi-colons.
procports	If set, forms a list of secondary TCP/IP ports that can be used for connections. Each client connection requires two or more secondary ports, so the number of assigned procports should be at least twice the number of licensed graphical users. Ports can

	be separated by commas, and ranges can be specified with *low-high*, such as 9001-9100. If not set, then random ports will be selected as needed. If you operate a firewall, or for other reasons want to control these ports, then specify procports.
hideconn	This parameter is used on Windows to hide the connection windows that appear when connections are made. If set to 0, then the connection windows will appear on the server task bar as connections are made. If set to 1, then connection windows are hidden.

In addition to the gen60d.ini file, the PRO/5 or BBj version of the server uses a configuration file called config.gen, in which you define starting setopts, file system parameters, a search prefix, and server printing devices and other aliases. This is a standard config.bbx file that is referenced when the server programs are launched. Configuration under ProvideX must be provided by the start.pv program, discussed in the next chapter.

Custom Startup Program

The graphical server supports the automatic execution of a program called “start.bb” (“start.pv” on ProvideX). This program is called with a single argument, the numeric port number on which the server is listening. The program is called once per client session, as the Windows client connects to the server.

Within the program, you can customize things such as global strings or directory prefixes. At the time this program is CALLed, the user has not yet logged into the server, so no parameter passing is possible (for parameter passing and programmatic response, see the next chapter), but if there are static, system-wide global strings that can be defined, this is a place to do so.

The startup program is not overwritten by normal installation/update procedures, but to prevent accidental loss, it is a good idea to back up this file when changes are made.

Parameter Passing for Run-time Variables

The [param] and [params] sections of the server's gen60d.ini file are used to define the proper prompt and valid values to the user when the GUI client needs to communicate a parameter, such as a company code, to the server. The purpose of this parameter is to provide the data needed to establish run-time variables, which the General data dictionary supports for dictionary parameterization.

For example, if a dictionary defines a customer code with a length that may vary from one installation to the next, the dictionary could be parameterized with type codes and expressions:

Type code: T,L[[@CUSTLEN]]
Expression: @PF1(3,[[@CUSTLEN]])

In this example, a run-time variable CUSTLEN could be defined when General is started, and reports would use its value in place of [[@CUSTLEN]] when resolving the dictionary.

Generally, parameters such as these are set up in a program that is executed when General is started. To configure this operation, you must define three things.

In the [param] section of gen60d.ini, specify two lines:
caption=*display prompt*
call=*program*

The caption value defines the label used by GUI Login screen to describe the parameter, and the call value defines a parameter setup program to CALL on the server (see below).

In the [params] section, specify one or more lines that indicate parameter values and descriptions, such as:

[params]
01=Acme Distribution
02=Owner's Fat Cat Company

With these items defined, the following occurs:

The GUI client login screen will build a list of valid parameter options, based on the [params] values, and label that list with the caption value. The user can then select the parameter at login time. In this example, the user would choose between Acme Distribution and Owner's Fat Cat Company. The server would then receive a 01 or 02 parameter value in the global string "$paramval".

When the user logs into General, the server will CALL the program named in the call= line. That program can use the value of stbl("$paramval") (gbl("$paramval") on Providex) to set run-time variable values. This program is CALLed with one argument, errmsg$, which can be set to a non-null value in the program to stop a report from running.

Additional global strings that can be used include:

*GENUSER for the user login name
*GENUSERLEVEL for the user's access level (0 though 9)

Then the CALL program can set global strings for the run-time variables, or conditionally return an error message to the client. For example, if the user level must be at least 8 to access Owner's Fat Cat Company, you could use logic like:

if stbl("*GENUSERLEVEL")<"8" then if stbl("$paramval")="02" then errmsg$="Access not allowed to this company"

Be sure your program has good error trapping and no user interface features (it will be running in background). Also be sure to close file channels opened in your program before exiting.

An example of a CALL program is genparam.bb or genparam.pv, found in the server directory.

Licensing

General provides three licenses, one for traditional character mode users and two for GUI client users (standard and run-only). Each license is for concurrent "users", though the interpretation of a "user" is different for each version. A user in character mode is a screen that is logged into General. A user in graphical mode is a seat identified by IP address on the network. It is possible for a GUI user to have more than one General client window open and still count as one "user". A run-only graphical user can only execute pre-defined reports, while a standard graphical user can create, edit, and delete reports, and will also have administrative options if logged in as an administrative user.

User counts are available in these increments for each mode: 0, 1, 3, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, and unlimited (unlimited graphical clients allows any number of both standard and run-only executions).

A license consists of a serial number and activation key combination. The serial number is simply that of the BBx or ProvideX interpreter used on the installation system (on ProvideX, it is specified as "PV" plus the last seven digits of the 16-character SSN value). The activation key links the serial number to a user count. When you purchase General, you specify the serial number and user count for each mode, character, standard graphical, and run-only graphical, and you receive two activation keys (you may also receive two serial numbers, if the character mode product and graphical server are to run under different interpreters).

Character Mode Activation

To license the character mode product, you can do one of two things:

- If you have not yet licensed the product, then each time you run or call "gen6", you are prompted for an activation key. Enter the key provided and the product will be licensed.
- Alternately, enter this command from a BBx or ProvideX console mode prompt: **call "gen6","activate"**. Enter the activation key when prompted.

Graphical Mode Activation

Activation for graphical clients is performed with the graphical server, as it controls the number of clients logged in at one time.

If you purchased a bundled version of the graphical mode server, then you will need to license the runtime before activating the server. Follow the steps outlined in the license information page provided, which in summary do this:

- On Unix, run the license.sh script and select the request a license option.
- On Windows, choose the Request a License option from the Start menu.

Each of these options will produce a license request text file, which must be emailed to licreq@synergetic-data.com. It is important that this request file be sent in its entirety, and as text rather than as a binary attachment. It may be necessary to copy and paste its contents into a message body. In response to this email, you will receive a license file, which should be saved as the file /usr/lib/sdsi/rt/pro5.lic, or as c:\sdsi\rt\pro5.lic on Windows. On Unix, use license.sh to stop the license server or re-read the license file. The next time the server is started, it will use the new license file and correct serial number. Then you can proceed to activate the server.

To license the graphical server, you must stop the server. On Unix, enter the command **gen60d stop**. On Windows, use the Gen60 Server Status option, or the control panel Services applet if the server has been installed as a service.

Finally, activate the graphical server. On Unix, enter the command **gen60d –act**; you will be prompted to enter the activation keys for standard and run-only modes. On Windows, use the Gen60 Server Activation option, and enter the activation keys when prompted.

Report Writing

General provides two methods of designing and running reports: Prompt mode and List mode. The Prompt mode provides a full-screen, interactive session that builds various aspects of a report by filling in forms and tables. These designs can be saved and re-called at any time. Once executed, a List command is built from the report definition, and that command is executed.

List mode provides direct access to the List command itself. The list command is an English-like query language with support for all of General's report writing features, such as field selection, row layout, break points, selection criteria, and sorting. When entering a List command, you can turn on Assist mode, which provides pick-and-point access to data fields and keywords. In graphical mode, the Assist mode is automatic. In character mode, Assist mode is enabled by pressing F2 when in the List command entry screen.

Accessing GENERAL

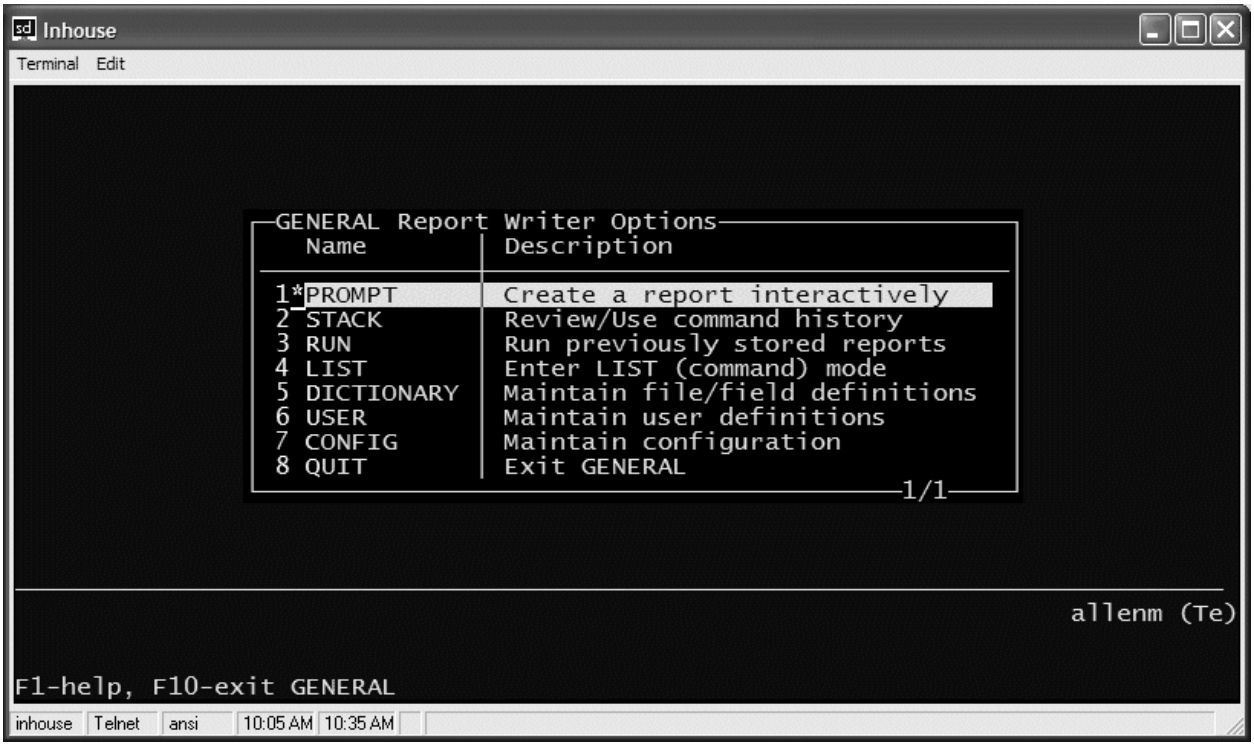
To execute GENERAL in traditional mode, the application to which GENERAL is linked should provide a menu option that RUNs or CALLs the program "gen6". GENERAL will then load and run, and in most cases display the login screen.

To login, enter a User Code, and if that user code is password protected, enter the password. User codes and their associated passwords are defined by GENERAL's USER command, which is accessible to administrative-level users. An administrative-level user is simply a user whose User Code provides the highest security level possible (level 9).

If no user codes have been established (this is a new installation, for instance), then the user code GEN may be used. Sites are encouraged, however, to establish regular user codes and to delete the default GEN code.

If you see a prompt for an activation key, then General is not yet licensed for the serial number of the PRO/5 or ProvideX under which it is running. For up to 30 days, you can run General in demonstration mode by leaving the activation key field blank and pressing Enter.

Main Menu



The main menu provides navigation to General's features. All users with access to the standard menu have access to Promp5, Stack, Run, List, and Quit options. Administrative users (those with an access level of 9) see all options, including Dictionary, User, and Config. Users that are configured to go directly to a custom report menu will not see the standard menu.

Here is an overview of each menu option:

PROMPT is the interactive program that provides full-screen report development. This mode is generally easier to use than LIST mode. Most List command options are supported by the entry of fill-in fields and tables. Experienced users can use LIST mode for quick, simple reports, or to take advantage of features that may not be supported by PROMPT mode. While PROMPT mode is interactive, when a report is executed, a List command is generated and executed.

STACK provides a recent history of List commands executed from any report writing mode. These commands can be edited, saved as RUN commands, an re-executed. Command history is stored across logins for each given General user ID.

RUN mode provides access to List commands saved from the STACK command. These commands can be edited and re-executed on demand.

LIST mode provides an ability to enter List commands directly, optionally with a pick-and-point assisted mode for choosing files, fields, and command options.

DICTIONARY is the program used to maintain General's data dictionary. The dictionary defines the data fields in your data files, as well as calculation fields, internal file sorts, and link specifications between files.

USER is the program used to maintain user Ids for access to General.

CONFIG is the program used to edit several configuration options, such as display formats, default heading characteristics, user-defined functions, and more.

QUIT exits General, returning to the CALLing program or RUNning a specified menu program.

PROMPT Reports

One of the two methods GENERAL offers to create reports is the PROMPT mode. This mode provides a full screen, interactive method of selecting files and fields, choosing sort criteria and break points, and establishing filtering criteria to print just selected records on the report.

PROMPT mode differs significantly from the LIST mode in its user interface, but both modes ultimately can produce the same report. GENERAL will convert reports designed in the PROMPT mode into LIST commands.

PROMPT mode is a maintenance program. Reports are identified by a report ID, and there are several options presented to maintain portions of the report definition.

A report definition is divided into the following elements, most of which are accessed by the ring menu at the bottom of the screen:

Header

The initial screen is used to maintain elements such as the report title, the file's name, column width, output device, and so on.

Field

Data fields are specified in a table, or can be "painted" via a visual mode within this section. Data can represent fields defined in the file's dictionary, calculations, linkages to related files, and text. In the visual mode, text can be typed where desired, and data positions can be defined or moved with cursor motion keys and function keys.

Break

GENERAL can generate subtotals whenever the value of a field changes while printing the report. The specification of the field to test is called a *break point*. Up to nine break points can be specified per report, with subtotal calculations nested automatically.

Sort

Normally, a report will be sorted by the natural order of the report file. The sort option allows the specification of one or more sort criteria, each nesting within the prior one. Each segment of the sort criteria can be ascending or descending.

Criteria

Records can be selectively printed in GENERAL. By imposing selection criteria based on simple relational expressions, such as YTD.SLS > 0, a report can filter the data before printing it. Filtering based on several criteria is also possible, with each expression connected with an AND or an OR.

LinkSel

One to many links provide access to detail records related to the report's primary file. The LinkSel option provides the ability to apply criteria to those detail records. This criteria is applied at a file level, so all links to that file will return only records that meet the criteria specified.

Hdr/Ftr

GENERAL automatically creates logical headers and footers for each report. If the automatic headers or footers are not what is desired, then they can be customized. Both report-level and break-level headers and footers are supported. Any headers or footers that are not customized, or are set to null, revert to the defaults.

Run

This option compiles and executes the report, after first prompting for an execute option. The execute options available are:

- The report, as generated, can be executed.
- The LIST command generated can be edited. Once edited, the options are again presented, so that the edited version may be executed or saved, using the next option.
- The LIST command generated/edited can be saved as a RUN command, just like a command saved from the command history stack.

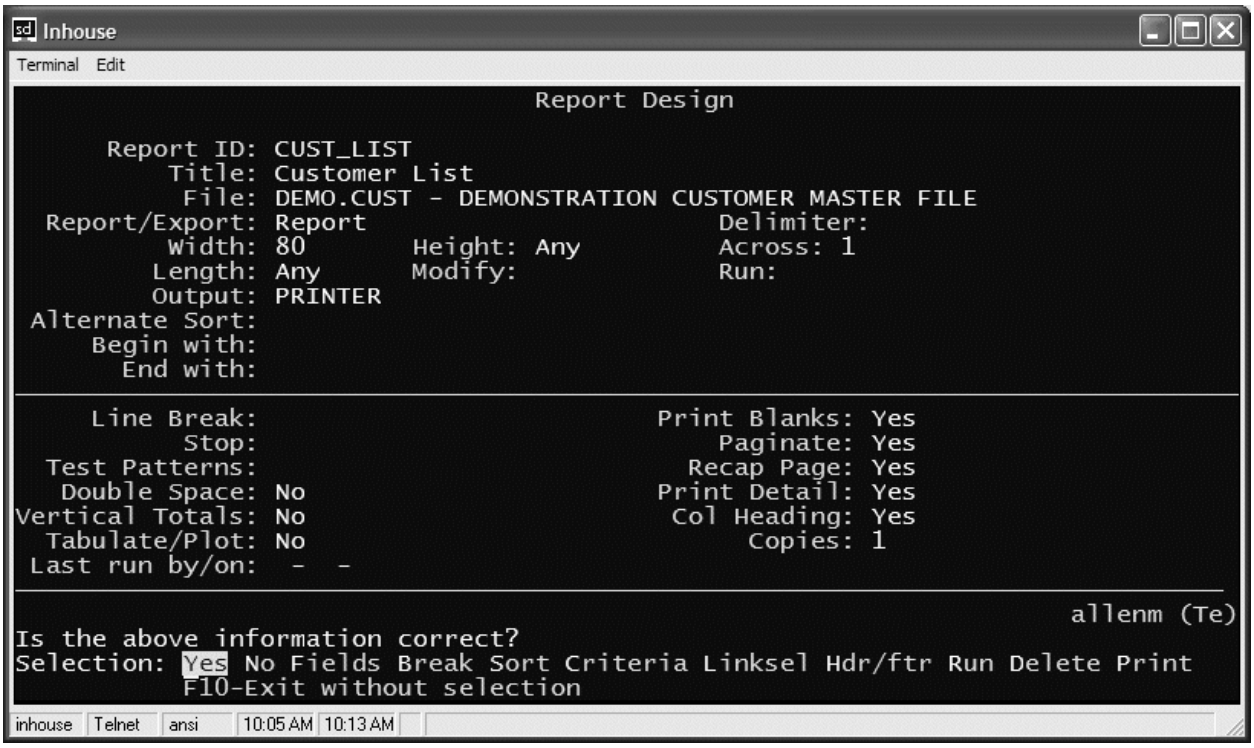
Delete

This option will delete the report definition, after verification.

Print

This option will print a hard copy of the report definition.

Header



Report ID

A report name can be from 1 to 20 letters, digits, and underscores. It cannot contain spaces. If the name entered isn't found, GENERAL will ask you if it is a new report definition, and offer an option to copy the definition of an existing report.

If you don't know the name, **F2-list** will present a list of previously defined reports and their descriptions.

Press **F10-exit** to return to the main menu.

Title

Enter a description for this report. GENERAL uses this as a title, if the report-level header is not customized.

File

Enter a file for this report to print from. The file must have been defined in GENERAL's dictionary, or in a supported external dictionary.

If you aren't sure of the name, enter a wildcard, such as AR*, or press the F2-list files key.

The file entered here is called the *primary file* for the report. If the dictionary specifies linkages from this file to other files, then data presented on the report can include data from those related files.

Report/Export

Enter the format of this definition.

1	if this is a report, with columnar formatting
2	for a delimited ASCII export
3	for a fixed position ASCII export
4	for a Data Interchange Format (DIF) export
5	for a WordPerfect merge format export.
6	for a delimited ASCII export with column headers.
7	for an Excel export using sdOffice (sdOffice is a separate product) or DDE (Windows platforms, or WindX, only). The mode is detected automatically in most environments, though an sdOffice server machine may need to be identified by specifying an environment variable "SDHOST" or STBL or GBL table value "$sdhost" before General is started.

Delimiter

Enter the delimiter to use for a delimited export. The default delimiter is a comma ",". Use ASCII designation for non-printable characters like ~009 for a tab.

Width

For a report (as opposed to an export), enter the number of columns for the report. This value defaults to 132.

If you are creating labels, with multiple labels across the page, then this should specify the width for each label.

Height

If the report data fields are to be stacked, as in a mailing label, you can specify a fixed number of lines that GENERAL should *always* print, regardless of how many lines are actually filled with values. If this setting is left blank, or is set to 0, then GENERAL will base lines printed on the data itself at run time.

Across

To print more than one record across a page on a line, specify the number of records to spread across the page here. This is useful for labels, when the number of labels per line is more than one. Take care to properly set the Width value for each element going across the page. GENERAL will figure out how many total columns are required by multiplying the Across value and the Width value.

Output

Enter the standard output device for this report. If not specified, then the report will print to the VDT.

To prompt for a printer, a VDT, or a file name at run-time, enter the word **PRINTER**. For an export, if only a file name should be prompted for, enter the word **FILE**. To print to a work file first, display the report then optionally print the report, just enter the word **PREVIEW**.

Length

Enter a specific number of lines to print per page, or 0 (Any) for a value that is determined by the system at run-time.

Modify

Enter the minimum user access level required to modify this report design. A value of 0 allows any GENERAL user to modify the design.

Run

Enter the minimum user access level required to run this report design. If a user has an access level of 5, and a report design has a Run level of 6, the user could not view, run, or modify the report.

Alternate Sort

If the file specified has Sorts defined in its dictionary, then any of those sorts can be specified to alter the natural order of processing for the report. When a sort is specified, the Start With and End With range specifications apply to the Sort order rather than the file's default sort order.

If you don't know the name of a sort, then press the **F2-list alternate sort** key to choose from a list.

Start with

To process just a range of records, rather than the whole file, enter a starting point here. This starting point relates to either the natural order for the file, or to the Alternate Sort, if specified, above. The report will begin processing records at the first record after (or matching) this entry. Press the **F2-Expand window** key to expand the size of the entry from 40 characters to 900 characters. To use an expression rather than a literal value, start and end the field with a single quote (‘).

End with

Enter the ending point of the range. Press the **F2-Expand window** key to expand the size of the entry from 40 characters to 900 characters. To use an expression rather than a literal value, start and end the field with a single quote (‘).

If either the Start with or End with element is left blank, then the report will process from beginning or through the end of the file, respectively.

Examples of Start with or End with values:

A literal value:	01
A prompted value:	[[Enter starting company code]]
An expression:	‘ ”01”+HTA(BIN(JUL(12,1,2001)),3) ’

Line Break

To generate a blank line every *n* lines, enter the number, from 2 to 99.

Stop

To stop the report after a certain number of lines have printed, enter that number here.

Enter **1-No** or **2-Yes** to each of these toggles. They each do the following:

Double Space will add a blank line after each record printed.

Vertical Totals will force any footer to format total presentations in a vertical, table oriented structure.

Print Blanks will print all lines specified, even if they contain no data. For example, if a name and several lines of address are stacked, setting Print Blanks to No will suppress printing of empty address lines.

Paginate controls whether or not page headers are printed. If it is set to No, then data is printed in a stream. Normally, this would be set to Yes.

Recap Page, if set to No, will suppress the printing of the recap page. The recap page is a summary of the report, printed after the report is complete.

Print Detail indicates that each record should be printed on the report. If it is set to No, then only subtotals and report totals are printed.

Col Heading indicates that default page headings should include column headings. If it is set to No, then only the title is printed. If the report heading is customized, this option has no effect.

Label Printing

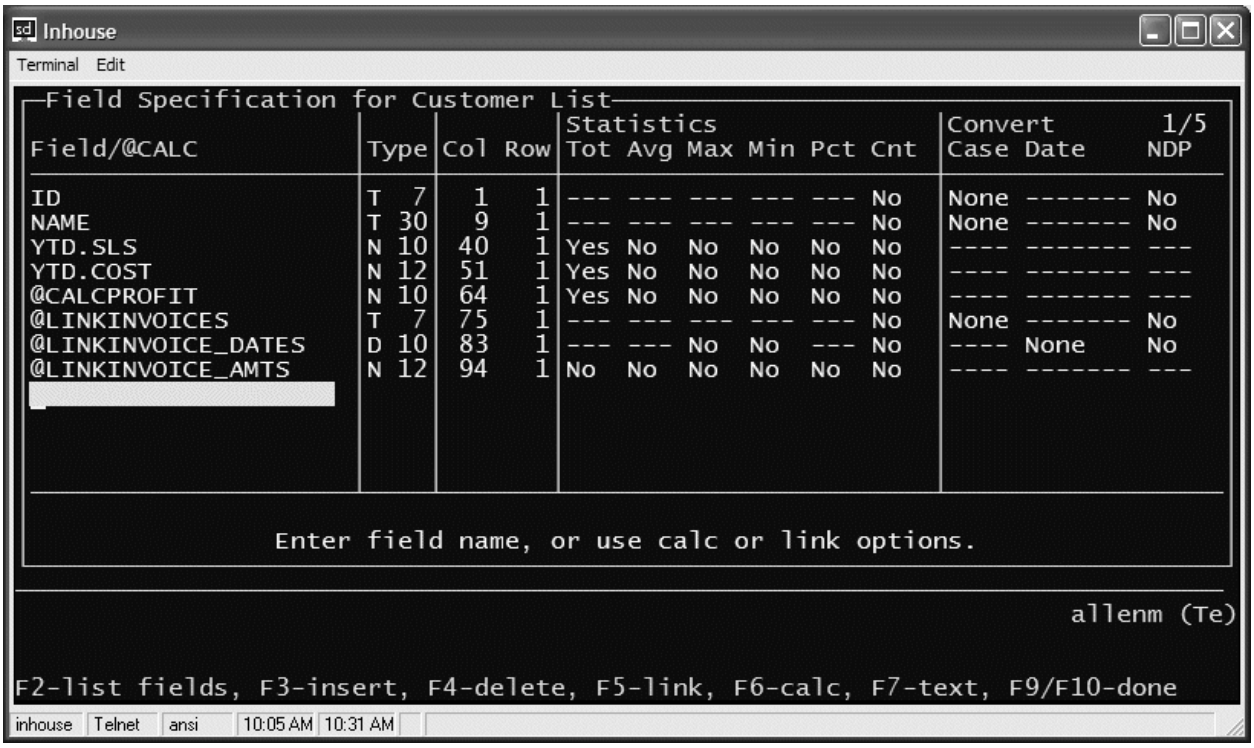
To establish parameters for labels in Prompt mode, determine the number of columns and rows each label uses in the mode your printer will print. A typical 3.5" by 1" label will allow 35 columns and 6 lines per label when the printer is printing a 10 CPI. When there is more than one label per line calculate the columns from left label edge to the next left label edge, as opposed to just the exact dimensions of a single label.

The **Width** element should be set to the number of columns from label to label, and the **Height** element should be set to the number of lines from label to label. Finally, the **Across** value can be set to the number of labels per line.

To keep GENERAL from producing a page heading and form feeds between "pages", set **Paginate** to No.

To cause GENERAL to shrink up any blank lines on the label, set **Print Blanks** to No. As long as the Height element has a value, the labels will print with the proper number of lines.

Field Specifications



The data for a report is specified through the Data Specification table. Field names, calculations, linkages (fields from related files), and text, can all be specified in any row of the table.

Moving across the table, GENERAL prompts for optional information, based on the type of data specified in any given row. Pressing the **F9-exit** or **F10-exit** key will invoke the verification prompt.

From the verification prompt for the table, Visual mode can be accessed, which formats an image of the report lines and allows editing of that image to add text or new fields, or to modify existing ones. Changes made in visual mode are reflected in the data specification table, and vice versa.

Field/@CALC

Enter a field name to place on the report, or choose one of the function key options.

F2 list fields	will present a selection list of fields defined in the dictionary for the report file.
F3 insert	will insert a blank row into the table.
F4 delete	will delete the current row. If there is a field specified in the row, GENERAL will verify the deletion.

F5 link	will issue prompts to generate a link field definition. A link field definition instructs GENERAL to get data from a related file for this report. More information on link field definitions is presented later.
F6 calc	will issue prompts to generate a calculation field definition. More information on calculation fields is presented later.
F7 text	will issue prompts to place literal text on the report. Note that the Visual option also allows placement of text on the report.
F9 done or **F10 done**	will exit table maintenance and present the verification prompt

Column Specifications

Type is not maintained. It indicates the date type (Text, Number, or Date) and the columns for the data on that row, as specified in the data dictionary.

Col and **Row** specify the position for the data. GENERAL maintains this information for you, or it can be overridden.

As new data is inserted, column and row positions will "float" as long as GENERAL keeps track of column and row positions. Once the column or row position is manually changed, it becomes "fixed", and GENERAL will not attempt to change the position.

If the position has been fixed, it is possible to force it back to a floating position calculation by pressing the **F2-float position** key, or by entering 0 as the column position.

Note: once Visual mode is used to maintain the format of the report, all data positions become fixed, unless Visual mode is exited with the **F10-exit** key or no data is changed while in Visual mode.

If a position conflict is detected, GENERAL will place an asterisk (*) between the column and row position fields of the rows that would overlay other data.

Tot, for numeric fields only, may be set to yes to generate automatic calculations for subtotals and report totals for the data.

Avg., for numeric fields only, may be set to yes to generate average calculations.

Max, for numeric or date fields, may be set to yes to print the column maximum at subtotal and report total levels.

Min, for numeric or date fields, may be set to yes to print the column minimum.

Pct, for numeric fields only, may be set to yes to print a calculated percent of total for each line, including subtotal lines.

Cnt, for any field type, may be set to yes to print the number of records printed at subtotal and report total points.

Case, for text fields only, may be used to override the default case for the field:

1	No case conversion
2	Force to upper case
3	Force to lower case
4	Force to proper case (This option will slow report printing.)

Date, for date fields only, may be used to override the default format and precision of date fields:

1	Jan/2002
2	01/2002
3	01/02
4	Jan/02
5	2002
6	2002/01
7	02/01

NDP, for text or date fields, if it is set to Yes will suppress the printing of duplicate sequential values in the column.

Link Field Definitions

Pressing the F5-link key for a field name accesses the link field definitions. GENERAL will then:

- Prompt for a 1 to 15 character link definition name
- Present a window of related files to choose from
- Present a window of fields from the target file.

Once each of these elements has been selected, GENERAL creates a link definition field in the table, and continues prompting for other table elements.

To modify the definition of an existing Link, press the **F5-link** key while the cursor is on the link field name.

Note that you can also enter Link Field Definition mode by entering a unique @LINK*name* in a new row. GENERAL detects the @LINK and begins the prompts for the definitions.

To abort the definition at any time while in the prompts, press the **F10-exit** key.

Calculation Field Definitions

Calculations are defined with the F6-calc key. GENERAL will prompt for:

- A calculation name (1-15 characters)
- Type codes
- A column heading
- An expression

The name uniquely identifies this calculation, and in later calculation definitions (defined after this one), the calculation can be referred to by its name.

The type codes, column heading, and expression instruct GENERAL how to derive and format the calculation. These elements are described in detail in the Dictionary chapter, in the Fields section, and also in the Keyword Reference, under @CALC.

Briefly, type codes always contain a data type and length:

T,L30 defines a 30 character text field.
N,L12 defines a 12 character numeric field.
D,L10 defines a 10 character date field.

Additional codes can be used to define justification, line depth, numeric precision and formatting, and time precision.

When defining an expression press **F2-list fields** to choose a field from the current file, **F3-link expression** to choose a field from another file, or **F4-expand window** to increase the size of the expression field.

Visual Mode

Editing the Report

PROMPT's visual mode provides an on-screen view of how the report columns will look when the report is generated. Visual mode is a simple text editor, with the fields from the Field Specification table placed at the row and column positions indicated. Fields are indicated by a two-character code, with the balance of the field length filled with tilde (~) characters.

Within this mode, fields can be moved or modified, new fields can be defined, and text can be added, all through use of the function keys, cursor motion keys, and typing.

Once editing is complete, GENERAL scans the text and updates the Field Specification table.

Take care, when editing the visual image, that the field identifier codes don't get changed. Also, ensure that there is always a space in front of the code. It is possible, for example, to type a new version of a code and tilde sequence, and then erase the old one, but if you make a mistake, or exit the image prematurely, you may lose the field specification. The preferred method is to use the **F3-move** function key.

Note that changing the length of a field by erasing tildes doesn't modify the field definition length. GENERAL will reinstate the defined length once visual mode is exited.

Editing data fields

Press the **F2-data** key at any point. If the cursor is on an existing field, then that field definition is placed in a definition window. The definition window provides maintenance to each table column (Tot, Avg., NDP, etc.) for the field specified only.

If the cursor is on blank space, the field definition window is opened, and you can enter or search for the field you wish to place at the cursor point.

GENERAL will verify that the length of the field chosen will fit where the cursor places it, so be sure you allow space after the cursor for the field you intend to place there.

Moving fields or text

To move any word in the text, including fields, place the cursor on the word and press the **F3-move** key. GENERAL will block out the field and present a "move to" edit screen. Move the cursor to the point where you want the left edge of the word to be placed, and press the **F3-move** key again. If the word will fit in the space chosen, it is moved there. If it is a field definition, the column and row positions are updated immediately.

To abort the move, and restore the word to its original position, press the **F10-exit** key before completing the move.

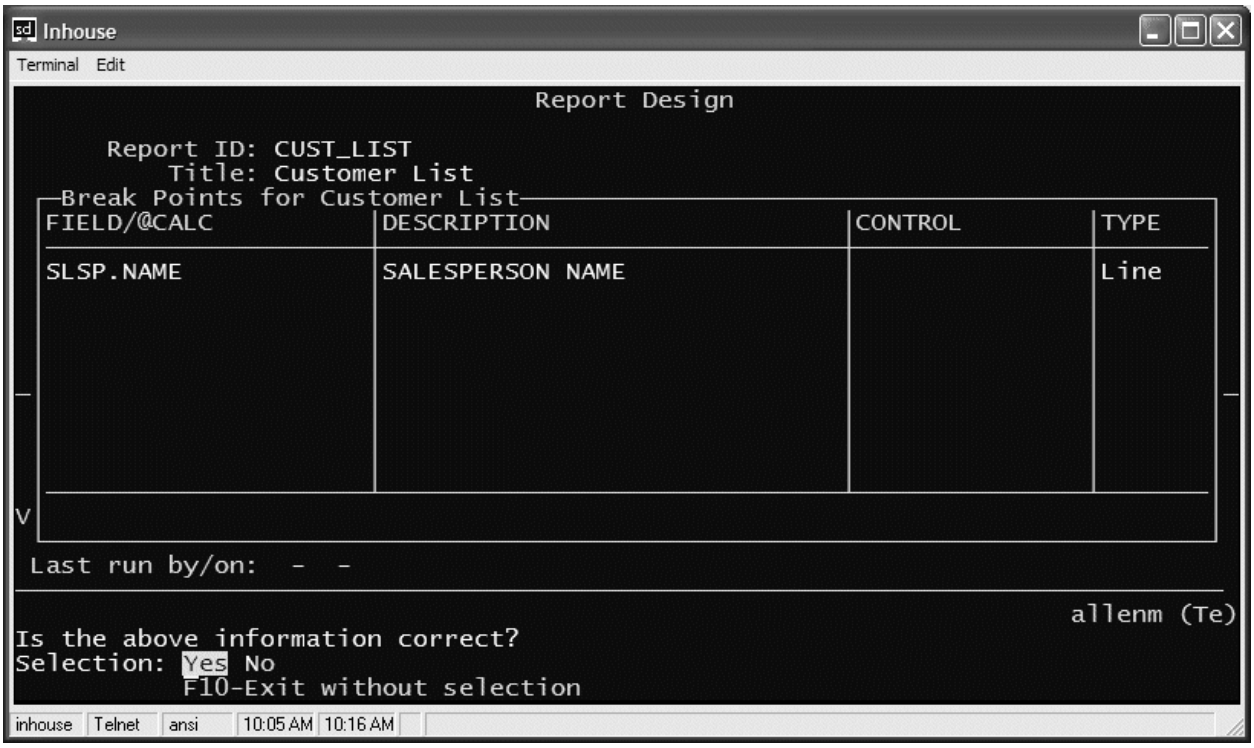
Saving or Aborting

When visual editing is complete, press the **F9-exit** or **F10-exit** key. GENERAL will issue verification prompt.

Yes saves the image and updates the field definition table for any changes made.
No re-invokes text editing.

Press the **F10-exit** key to abort any changes made while editing the image. The original field definition table will be restored.

Break Specification



Break points tell GENERAL how to group data on a report for subtotal calculations. Normally, a break point will be associated with the sorting order for a report, but GENERAL doesn't force a relationship. It is therefore the report designer's responsibility to ensure that break point definitions are logical with regard to the order in which records are printed on the report.

A break point can be based on any field or calculation, and the data specified can be further modified when GENERAL calculates its value for testing whether or not to generate a break. For example, a date can be modified so that a break will only be generated when the year changes.

Break points can cause either a line break or a page break. A line break simply prints a break level footer, then skips a line, then continues, and either with the next group's header, or the records associated with the next group. A page break prints the footer, but then skips to the next page before continuing.

GENERAL supports up to nine break levels. The break point definition table provides nine rows. The order in which breaks are defined determines how GENERAL calculates any associated subtotals. The calculations are nested, so that a break in level 2 will trigger breaks in levels 3 through 9, with subtotals printed and initialized.

Break Control

If the break to be generated isn't to be based on the full value of the field specified, then specify a "control" value. Control values convert the data value for purposes of reducing the detail level of the break.

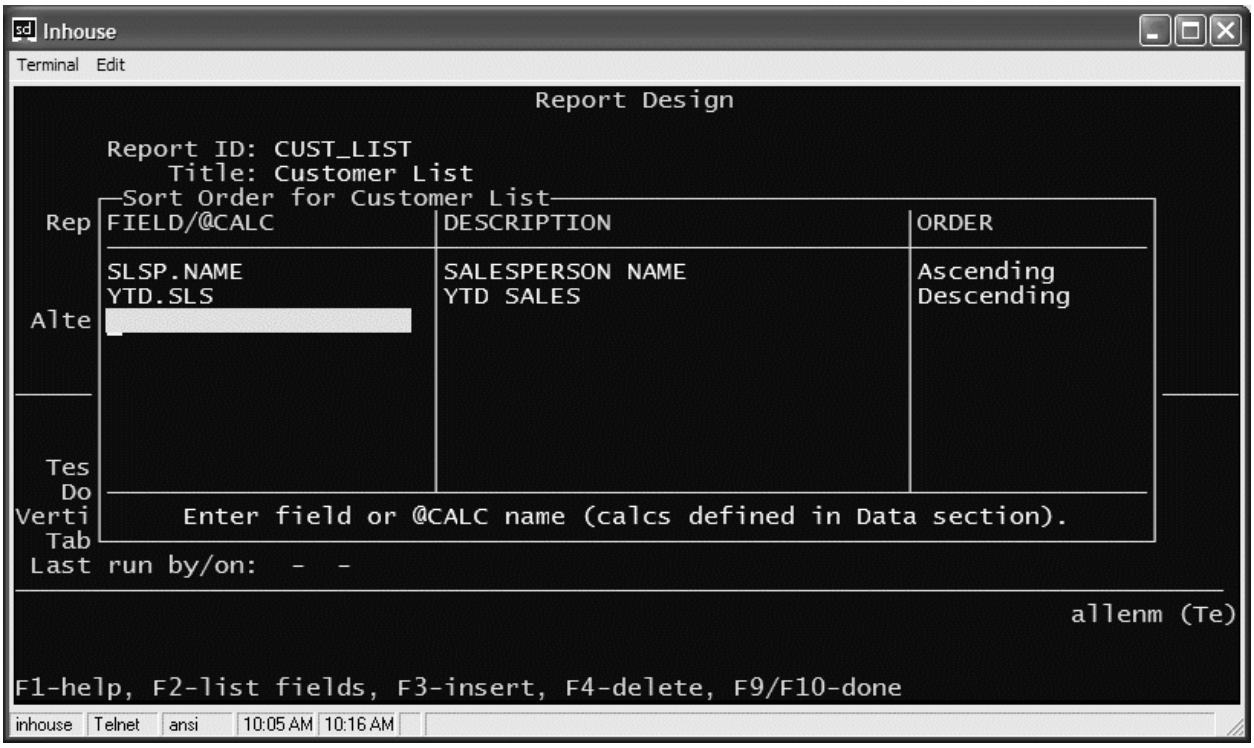
For a **text field**, you can modify the number of characters on which to base the break. If the number of characters were set to 1, then a break point would be generated for each letter of the alphabet and each digit.

For a **numeric field**, you can to group values together in even amounts. For example, entering 5000 would group all values 0-4999 as "0", 5000-9999 as "5000", 10000-14999 as "10000", and so on.

For a **date field**, GENERAL will normally break when the date changes, however, you can adjust the break to a month or year precision, and specify a format for the break value:

MMMYYYY	Jan/2002
MMYYYY	01/2002
MMYY	01/02
MMMYY	Jan/02
YYYY	2002
YYYYMM	2002/01
YYMM	02/01

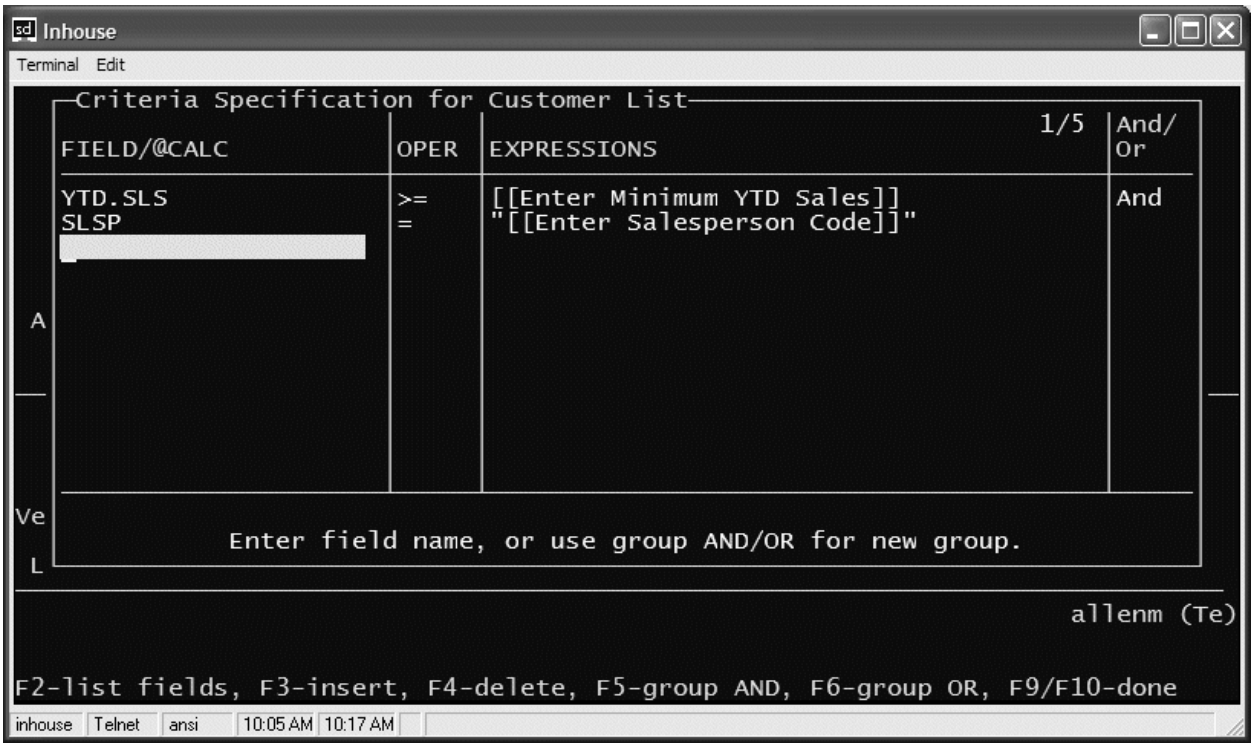
Sort Specifications



Reports in GENERAL may be sorted on any number of fields, so long as the number of characters used by all sort fields, plus GENERAL's 2-character overhead, doesn't exceed the underlying language's key size limit. When calculating field lengths, note that numeric and date fields always use 10 characters in a sort. If the fields specified do exceed the key size limit, GENERAL will truncate automatically to that limit.

The PROMPT mode sort table can handle up to 9 sort field levels, which should accommodate any report. If more sort levels are required, then the report must be generated through List Mode entry. The sort levels specified sort the report in hierarchical fashion. The second level sorts within the first, the third within the second, and so on. Each level can be sorted in either ascending or descending sequence, specified in the Order column.

Criteria Specifications



In GENERAL, the method used to impose filtering criteria on the data to be placed on a report is the *select phrase*. A select phrase is made up of one or more *relational expressions* that compare two fields, or a field and a constant, to determine if the criteria of the expression is true or false. If multiple expressions are required, they may be connected by the keywords AND or OR. The AND connector indicates that the two expressions so connected must each be true, while an OR indicates that either may be true.

For still more control, GENERAL supports the use of parentheses. Normally, AND takes precedence over OR, so that: ***expr-1* OR *expr-2* AND *expr-3*** evaluates as *expr-2 AND expr-3* first, then if both expressions are true, or *expr-1* is true, the selection passes. What if the desired evaluation is actually to test *expr-1* and *expr-2* first, then if either passes, and *expr-3* passes, the selection passes? To do this, parenthetical control is necessary:**(expr-*1* OR *expr-2*) AND *expr-3***.

In PROMPT mode, GENERAL supports both AND and OR connectors, and one level of parenthetical control through the "Group AND" and "Group OR" functions. In especially complex selection phrases, sometimes a greater level of parenthetical control is required. In such cases, List Mode entry of the report will be required.

Expression elements

The criteria specification table provides a row for each relational expression, and the connector to the next relational expression, if required.

The **Field/@CALC** column can be any field from the file being LISTed, or any calculation or link name defined in the Data section of the report definition.

If you aren't sure of the names, use the **F2-list field's** key. Other function key actions available in the Field/@CALC column are:

F3	inserts a blank row.
F4	deletes the current row.
F5	denotes a new parenthetical level, connected to the prior level with an AND.
F6	denotes a new parenthetical level connected with an OR.
F9/F10	exits selection criteria maintenance.

The **Oper** column (operators) provides for specification of how the relational expression elements are to be compared. The operator is a Boolean math symbol from this list:

=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

If the Field/@CALC column specifies a text value, as opposed to a numeric or a date, then the following operators are also available:

==	Contains
+=	Contains at position 1 ("starts with")
-=	Does not contain
~=	Approximately equal to (may vary by implementation; on BBx this implements regular expression matching).

The **Expressions** column allows entry of a comparison expression. This can be a simple field, calculation, or link name, a constant, or a mathematical expression involving field names, constants, and simple math symbols (+ - * /).

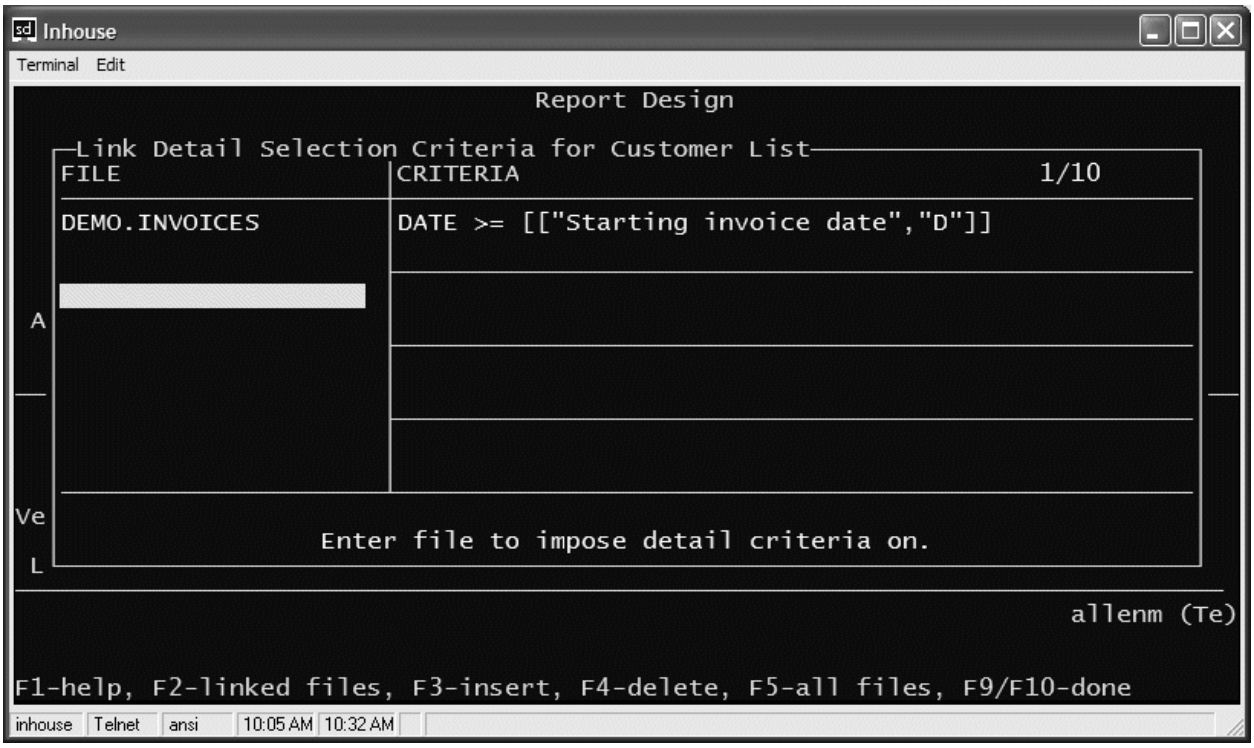
The expression must evaluate to the same data type as the Field/@CALC data. For example, numeric fields must be compared with a numeric expression, text fields with a text expression. Examples, in each case, might look like this:

Numeric	0 1000 YTD.SLS + 9500
Text	"" "92701" STATE+ZIP
Date	CND "123101" (uses CONVERT-DATE keyword) "12/31/01" implicit date conversion in select phrase DATE.DUE + 30

The first three columns make up an entire relational expression, and the last column can be filled in if required, to connect two relational expressions.

If the field or @CALC value is a text field, and the expression entered doesn't appear to be a field name or an expression, it will be quoted automatically.

LinkSel Specification

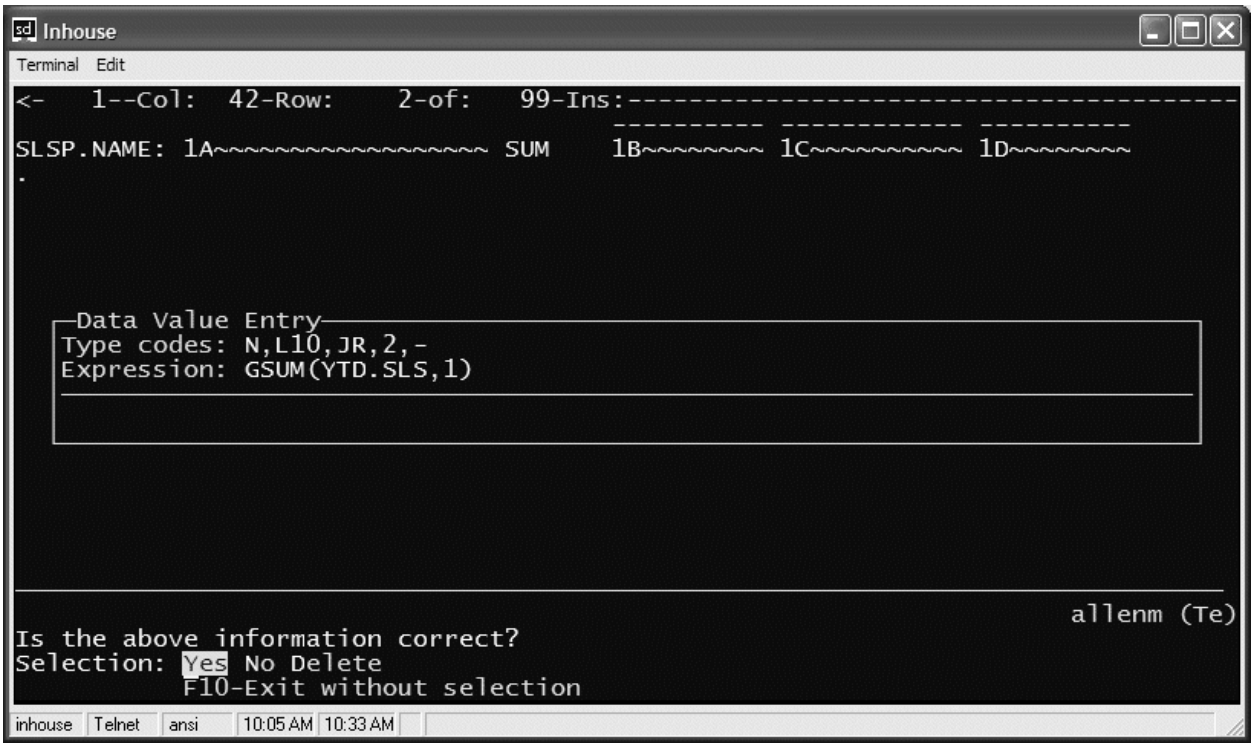


This table provides for entry of criteria that applies to one-to-many link data. Links that display or summarize one-to-many data normally use all the related records in the target file. Using LinkSel, however, you can define criteria to apply to those records.

The File column specifies the target file to apply the criteria to. Any one-to-many links in this report that use the file will have records limited to those that match the criteria. You can press F2 to select the file from a list of files related to the primary file for this report. This list should represent the only files that make sense to apply criteria to. If you would like to see a complete list of files, press the F5 key.

The Criteria column provides space to enter selection criteria, using fields from the target file and normal Boolean operators, such as =, <, <=, >, >=, and <>.

Headers/Footers



GENERAL supports the definition of custom headers and footers; in the LIST mode through the HEADER*n* and FOOTER*n* keywords; in PROMPT mode through the Hdr/Ftr section.

Hdr/Ftr first issues a selection window for the header or footer to customize. All reports have a Report Page Header and Footer. The page header is printed on every page, while the footer is only printed once, at the end of the report.

Break levels also have headers and footers. By default, GENERAL will produce break point footers, but not a break point header. Therefore, to produce such a header, customization is required.

Text Editing

Headers and footers are maintained in a text editor. Data and text can be placed anywhere in the edit region. The following function keys are active in the text editor:

F2-data
Add or maintain a data element in the header or footer. Data elements can include simple field or break value references, or can be calculation expressions.

F3-move
Move a word to another position in the text. If the word is a data position, then the definition is updated to reflect the new position.

F4-re-display
Re-displays the edit window.

F5-copy defaults
Overlays the text with the default header or footer that GENERAL would normally generate. If there is already text, GENERAL will verify the replacement before the old text is lost.

F6-copy from
Prompts for another footer or header level, and copies the custom footer or header to the current level being maintained. Any calculations that reference @BREAK*n*, or any group or report summary functions, such as GSUM(), will be modified to reflect the current level.

Data Definition

To add a new data element to a header or footer, move the cursor to the point where the data is to be placed, and press the **F2-data** key. To modify an existing data reference, place the cursor on the definition label, and press the same key.

Data elements are defined by type codes and an expression. The type codes are identical to those described in the Dictionary chapter for Field definitions.

Briefly, type codes always contain a data type and length:

T,L30 defines a 30-character text field.
N,L12 defines a 12-character numeric field.
D,l10 defines a 10-character date field.

Additional codes can be used to define justification, line depth, numeric precision and formatting, and time precision.

The expression can be a simple data element or break value reference, or can be a more complex expression. Use **F2-list fields** or **F3-expand** window to help create expressions. Some examples might be:

@BREAK1, which prints the value of the first break point level.

100*DIVIDE(GSUM(YTD.SLS,1),RSUM(YTD.SLS)) to calculate the group level 1 percent of the report total for the field YTD.SLS.

Moving text

To move a word from one point to another, use the **F3-move** key. Move the cursor to the word, and press the key. Then move the cursor to the new position and press the key again.

Blank lines
Add blank lines to a heading by placing a dot (.) or the word @LINE by itself at the left margin of the line.

LIST Command Entry

GENERAL's List mode is an "English-like" report writing language designed to make the process of creating reports both fast and intuitive. All reports in GENERAL are ultimately created by the report writing language, though in other modes the command is generated through some means other than direct entry. This mode is called "List Mode" because the command used to generate reports is called the "LIST" command.

List mode is comprised of a text editor, for direct entry of a LIST command, and also an *assist mode*, for point-and-shoot assistance with the creation of a LIST command.

Users of early versions of GENERAL will be pleased to see that the new command editor features multi-screen formatting of commands, with a special "execute" keystroke, rather than the basic one-screen, <Return> execution. The assist mode will likewise please users of older versions.

Executing a LIST command

To execute a LIST command:

- Type the command (or use the assist mode).
- Press the **F9-execute** key.

The elements of the command can be entered all together, or can be spaced across several lines or pages. After pressing the execute key, GENERAL examines the command, builds the appropriate run-time code, and produces the report.

While in the command editor, you can enter assist mode by pressing the **F2-assist** key, or copy a command from the command history stack by pressing the **F3-history** key.

To abort the command, and return to the main menu, press the **F10-exit** key.

LIST Command Specification

A LIST command is made up of five elements. Three of these elements are required; the other two, optional:

LIST	starts the command.
File-name	is the second word. The file must be in GENERAL's dictionary or in a supported external dictionary. This element is called the *primary file* or the *LIST file*.
Field-phrase	consists of field names, calculations, links, text, header and footer sub-phrases, and interspersed keywords. The field phrase controls both what will print on the report, and how it will look.

Sort-phrase	is one of the optional elements. The sort phrase instructs GENERAL how to sort the records before the report is produced. Sort phrases can consist of field names, calculations, links, and the DESCENDING keyword, which can precede any data element to change the sort order for that particular item. If the default case-insensitive sort is not desired, so that "Smith" would sort differently from "SMITH", then the FULLCASE keyword can be used before any field.
Select-phrase	is the other optional element of the LIST command. A select phrase filters the data to be placed on the report. Instructing GENERAL to only print records that meet the selection criteria imposed by the select phrase performs the filtering. The selection criteria is made up of one or more relational expressions, Boolean (AND/OR) connectors, and parentheses where required.

An example of a complete LIST command might be:

LIST	(LIST command)
DEMO.CUST	(*file-name*)
ID NAME TOTAL YTD.SLS	(*field-phrase*)
SORT DSND YTD.SLS	(*sort-phrase*)
SELECT YTD.SLS > 0	(*select-phrase*)

Run-time Replacements

At any point in the LIST command, a *run-time replacement* can be used rather than text. The most common run-time replacement is a user-prompt for selection criteria.

Run-time replacements can take one of two similar forms. The first one simply encloses a prompt in double brackets:

LIST DEMO.CUST ID NAME TOTAL YTD.SLS SELECT YTD.SLS > [[Minimum YTD Sales]]

This will issue the prompt **Minimum YTD Sales** at run-time, and use whatever value the user enters as the value for the report.

The second form provides more control. If the text prompt is enclosed in quotes, GENERAL looks for a type parameter, and an optional default value. The type can be T, N, H, or D, to require text, numeric, hidden text, or date input, respectively. A type of N can be immediately followed with a digit, indicating the decimal precision.

[["Minimum YTD Sales","N2",1000]] will prompt for a precision 2 number, defaulting to 1000.

[["Enter start date","D","123101"]] will prompt for a date, defaulting to December 31, 2001. Note that when prompting for a date, the value that will replace the prompt at run-time will be an internal date "number", not a date formatted in readable form. It would be useful in a select phrase, but not in a page header, for instance.

When a report is executed, all prompts are issued together at the end of the parsing process. Once all have been answered, the report can be executed by pressing the F9 key.

Sub-stringing

Data fields can be sub-stringed on the command line to print just a portion of the field. For example, **ITEM_CODE(1,3)** would display, sort, or select on just the first 3 characters of the ITEM_CODE field. GENERAL makes no attempt to verify the validity of the sub-string. If a field is defined in the dictionary as 15 characters, and the sub-string reference is *field-name*(5,20), an error will occur when the report begins to print.

This feature is a shortcut method for performing a simple command line calculation, and can be useful in sort phrases and select phrases. For instance, if a product code is in the first three characters of an inventory item, you could sort a report by product code and description like this:

SORT ITEM_CODE(1,3) DESCRIPTION

Compound LIST commands

Separating them with a double semi-colon can chain several LIST commands together. Each command will be produced as a complete report in sequence.

The following example will run three reports. The last report is a stored command name.

LIST DEMO.CUST ID NAME TOTAL YTD.SLS ON LP ;; LIST DEMO.CUST BREAK SLSP ID NAME TOTAL YTD.SLS ON LP ;; REPORT_1

Comments

Comments can be added to a LIST command by enclosing them between exclamation points (!). For example:

LIST DEMO.CUST ID NAME TOTAL YTD.SLS !Prompt user for selection criteria! SELECT YTD.SLS > [["Enter cutoff",N,0]]

Direct Entry

LIST DEMO.CUST ID NAME TOTAL YTD.SLS TOTAL YTD.COST

This example is a simple list command, used to generate a four-column report from the demonstration file, DEMO.CUST. This command has the three required elements of a LIST command:

- The command itself: **LIST**
- A file name: **DEMO.CUST**
- And a field phrase: **ID NAME TOTAL YTD.SLS TOTAL YTD.COST**

The field phrase contains four field names from the dictionary of DEMO.CUST, used by GENERAL to create the four-column report. The field names are ID, NAME, YTD.SLS, and YTD.COST. The command also contains the keyword TOTAL in two places, used to tell GENERAL to calculate a total for the two columns that the keyword precedes, YTD.SLS and YTD.COST.

Aggregate keywords

In addition to TOTAL, GENERAL has several other keywords used to calculate column aggregate values. They are AVERAGE, COUNT, MAXIMUM, MINIMUM, and PCT-TOTAL.

Each of these keywords can be used in front of a field on the report. AVERAGE, PCT-TOTAL, and TOTAL are valid for numeric fields, MAXIMUM and MINIMUM for either date fields or numeric fields, and COUNT for any type of field.

With the exception of PCT-TOTAL, more than one aggregate keyword can precede a field. GENERAL will do its best to line up the column calculations under the column, or it may switch to a vertical format if all the column calculations can't be made to fit under their respective columns.

Sorting

This next example introduces some new concepts. Only one additional column has been introduced, SLSP, but more control has been taken over the presentation of the report.

LIST DEMO.CUST BREAL SLSP ID NAME TOTAL YTD.SLS TOTAL YTD.COST SORT SLSP DESCENDING YTD.SLS

First, there is a sort phrase in this report. The sort phrase begins with the keyword SORT, and contains two fields, SLSP and YTD.SLS. The report is sorted by SLSP, then within SLSP, it is sorted by YTD.SLS. The keyword DESCENDING precedes the YTD.SLS field, so that segment of the sort is in descending order.

Sorting by SLSP causes all the records from each salesperson code to be grouped together, which leads to the next enhancement to the report: break points.

Break points and Subtotals

GENERAL supports the generation of report break points with the BREAK keyword. When the BREAK keyword is used, GENERAL watches the value of the next field while the report is printing, and when it changes, generates a break point. A break point causes the report to skip a line, normally to print subtotals for the group of records just printed. When a break point is generated, GENERAL prints a *break level footer*, then a *break level header* for the start of the next group.

The *level* indicates which break point is generated. GENERAL supports up to nine nested break points in a report. The first one encountered in the LIST command is the highest level, level 1; the second is considered level 2; and so on. Imagine a report of invoices, sorted by salesperson and date. A report could issue break points whenever either the salesperson or the month changed, by including BREAK SLSP ... BREAK MMYY DATE.... Whenever the month changed, a level 2 footer would be printed, and initialize any subtotals for the monthly calculations. There could be many month breaks for each salesperson, and there would be many footer/initialization cycles before finally a level 1 break is generated, which triggers both the final level 2 break, and the level 1 break.

GENERAL by default will generate footers for each break point. The footer will contain data for the break point value and any column calculations, and also some formatting text. GENERAL will not, by default, create a header for each break point. The only default header is the page header.

To generate a break level header, or to customize footers, the LIST command can contain HEADER*n* or FOOTER*n* keywords and associated sub-phrases.

There are several types of breaks. The example above shows a *visible line break*. That means that the field used to generate the break is visible on the report, and the break will skip a line before continuing on with the report. A page break can also be generated, with the BREAK-PG keyword. In this case, the break will skip to the next page before continuing the report. There are also *silent* versions of the line and page breaks SBREAK and SBREAK-PG. A silent break doesn't print a column for the break point value. Instead, the break value is only displayed in the footer. Break points generated by the PROMPT mode are always silent breaks.

Column Positioning

GENERAL supports multi-row reports, so that the data from each record can span several rows. Normally, when fields are simply entered into the LIST command, GENERAL will place them from left to right, one space apart, and will move to the beginning of the next line and continue if a field will cross over the right margin.

If that default positioning isn't what is desired, there are several keywords that can be used to override it. The most precise of these is the TAB keyword.

The TAB keyword accepts one or two numeric parameters. If there are two, then a period must separate them (and there can be no spaces). The first number is the column number for the next field. The second number, if present, is the row number. GENERAL remembers where the field would normally be placed, relative to the prior one, and if the row number isn't specified, the default row will be used. TAB 60.2 means "place the next field at column 60, row 2".

Other keywords used in formatting are NEWLINE and SPACE. NEWLINE causes the next field to be placed at the beginning of the next line, relative to the prior field. SPACE takes a simple numeric parameter, which defines how much space to place between the end of the last field and the start of the next field. The default is one space; SPACE 20 would use twenty spaces instead of one.

Link Field Definitions

GENERAL provides several ways of looking up data in files related to the LIST file. One of those methods is called a *link field definition*. A link field definition uses a pre-existing link specification between the LIST file and some target file. These specifications are defined in the dictionary. For example, there is a link specification between the DEMO.CUST file and the DEMO.SLSP file. The link field definition DEMO.SLSP:NAME uses it to look up the salesperson name from the salesperson file, and place the name on the report.

GENERAL recognizes the ":" as a link field definition operator. Immediately preceding the ":" is a file name. GENERAL will look for a link specification between the LIST file and that file. Immediately after the ":" is a field name. GENERAL looks in the target file for that field. Assuming that all the information is found, a cross-reference calculation is built to retrieve the desired data.

Link field definitions can be more complex than this. For example, if there is a partial key (one-to-many) relationship to the target file, then the file name can be preceded with an aggregate operator: @SUM:, @AVG:, or @CNT:. These instruct GENERAL to total, average, or count all the related records in the target file. The definition @SUM:DEMO.INVOICES:AMOUNT will total the AMOUNT field from all records in the DEMO.INVOICES file that are related to a given customer.

Also, link field definitions can be nested, so long as there is no more than one partial key relationship in the chain. The link field definition DEMO.CUST:DEMO.SLSP:NAME would chain from whatever the LIST file is, to the DEMO.CUST file, then from there to the DEMO.SLSP file, finally retrieving the NAME field.

Literals

You can place literal text anywhere in a LIST command, by enclosing the text in quotes.

Run-time calculations

GENERAL's dictionary provides the ability to pre-define calculations, so that a user creating a report can simply reference the calculation as any other field. The fields CITY_ST_ZIP and MTD.PROFIT in the demonstration customer file are examples of pre-defined calculations.

Sometimes, however, a calculation may not be defined ahead of time. At those times, rather than adding the calculation to the dictionary, the calculation can be defined at run-time with the @CALC keyword.

@CALC is a function keyword, so that its parameters must be surrounded by parentheses. The parameters are:

- Type codes, which define the data type, length, and optional formatting information. The type codes must be enclosed in quotes.
- A heading also enclosed in quotes, to be used by the default page header.
- An expression from which the data to be printed is derived.

Each of these items is described in detail in the Dictionary chapter, under Field specifications.

Once defined in the LIST command, the calculation can be referred to by name in other calculations, or as a standard field.

Sorting by calculations

Reports can be sorted by a run-time calculation, just as if it were a normal data field. Like data fields, if the calculation results in multiple records in a partial key cross-reference, then the sort is invalid. However, any calculation that resolves to a single value is valid in a sort phrase.

One benefit of this is the ability to sort on a summary calculation, using the GSUM() and related functions. Here's an example:

LIST DEMO.CUST BREAK SLSP TAB 40 TOTAL YTD.SLS NO-DETAIL SORT @CALCSUM("N,L10,2","",GSUM(YTD.SLS,1)) DEMO.SLSP:NAME

The above report will produce a summary report of total year-to-date sales (YTD.SLS) by salesperson. The report will be sorted by the total YTD.SLS value calculated for each salesperson. This is accomplished by the GSUM() function, which summarizes a numeric field based on a break level.

The second sort level may be required due to the way GENERAL resolves ties in sort levels. Without the additional level, if two salespersons have the same subtotal, the detail for both salespersons will be processed in customer order, resulting in mixed up subtotals. The second level (DEMO.SLSP:NAME) will keep records associated with a particular salesperson together, after they have been sorted by the summary value first.

Other controls

The DOUBLE-SPC keyword turns on double spacing. The STOP keyword requires a numeric argument, and tells GENERAL to stop the report after that many records have been printed.

There are many keywords that execute various controls over the report. A whole series of switch keywords (NO-BLANK, NO-DETAIL, etc.) turn off various features. A glance through the Keyword Reference chapter provides a great deal of information.

Custom Headers and Footers

Custom headers and footers may be defined in the command with the HEADER*n* and FOOTER*n* keywords. These are described in detail in the Keywords chapter.

Printing the output

By default, when GENERAL produces a report, it is printed to the screen. To send the report to another output device, the ON keyword is used.

ON requires one parameter. That parameter is usually a printer name, such as LP or P1, or a filename, in quotes, such as "/usr/files/export.csv". There are also four special names recognized by GENERAL:

VDT	forces output to the VDT.
PRINTER	prompts the user for a printer device, the VDT, or a file. This is the most flexible printer "name".
FILE	prompts the user for a file name.
PREVIEW	prints the report to a work file, in any dimension (WIDTH and LENGTH can be specified), then presents a scrolling window view of the report on screen. If the report looks good, it can be sent to a printer.

Label Printing

Label printing with the LIST command is a simple matter of determining the number of columns and rows each label uses. A typical 3.5" by 1" label will allow 35 columns and 6 lines per label when the printer is printing a 10 CPI. When there is more than one label per line calculate the columns from the left label edge to the next left label edge, as opposed to just the exact dimensions of a single label. The keywords typically used for labels are:

WIDTH	sets the number of columns for each label, from edge to edge.
HEIGHT	sets the number of lines per label. This is required if the NO-BLANK keyword is used.
ACROSS	sets the number of labels across the page. ACROSS * WIDTH is the number of columns used by the report.
TESTPRINT	is used to specify a number of pattern labels that print before the actual labels begin.
NO-PAGE	turns off pagination. When printing labels, normally there should be no page headers or form feeds between pages. The NO-PAGE keyword turns them off. If form feeds are needed, then don't use NO-PAGE, but instead set HEADER0 to a blank value or one or more blank lines.
NO-BLANK	suppresses printing of blank lines. If the labels include multi-line addresses, where certain lines may be blank, the text of the label can be printed on sequential lines in all cases by using this keyword. Be sure, however, to use the HEIGHT keyword to ensure that the labels will be of the proper height. NO-BLANK will also eliminate blank lines from the top of the label. If a blank line is required at the top of each label, and the labels can't be physically aligned to accommodate this, then place a literal dot "." by itself on the line. TAB 1.1 "." TAB 1.2 NAME, for example.

Select Phrase

If there is no select phrase, then GENERAL will produce a report from all the records in a file. The only exception is if the dictionary has been defined to include an *auto-select phrase* or a *skip keys definition*.

When there is a select phrase in a LIST command, then the criteria indicated by the select phrase expressions are tested before a record is printed on the report.

Select phrases can be refined to include any number of relational expressions, separated by the Boolean connectors AND or OR, and grouped with parentheses, if necessary.

Evaluation Order

Parenthetical grouping may be required to control the order of evaluation. **A>B AND C>D OR E>F** would group the first two expressions together, so that either: **A>B AND C>D** or **E>F** would pass the selection criteria.

If the desired selection is to test the second two expressions together, then parentheses are required: **A>B AND (C>D OR E>F)**.

Numbers, Text, Dates

The different data types used by GENERAL each must be referenced properly in a select phrase. Of course, when comparing two fields, then the fields must be of the same type. When comparing fields to constants then the constants must be properly formatted.

For text fields, constants must be quoted text. The phrase fragment **SLSP = "100"** is an example of this.

Numeric fields must be compared with unpunctuated numeric values. The fragment **YTD.SLS > 5000** shows 5,000 as an unpunctuated number. Negative numbers should be entered with a leading minus sign: **-5000**.

Date fields must be compared to an internal date value, as defined by the host Business Basic language. GENERAL can detect if a select phrase expression requires a date field if the first field reference is a date data type. For example, in the expression **DATE > "011501"**, the text value "011501" will be automatically converted to a date, because the DATE field has indicated to GENERAL that a date value is required. The literal date can also be entered with delimiters, such as "1/15/01 ", as long as the order of the month, day, and year segments matches the date entry convention established for the system.

If a literal date value is required before a date field is encountered in the expression, then the CONVERT-DATE keyword may be used. **CONVERT-DATE "1/15/01"<DATE + 30** would force the literal date value to be converted to a date.

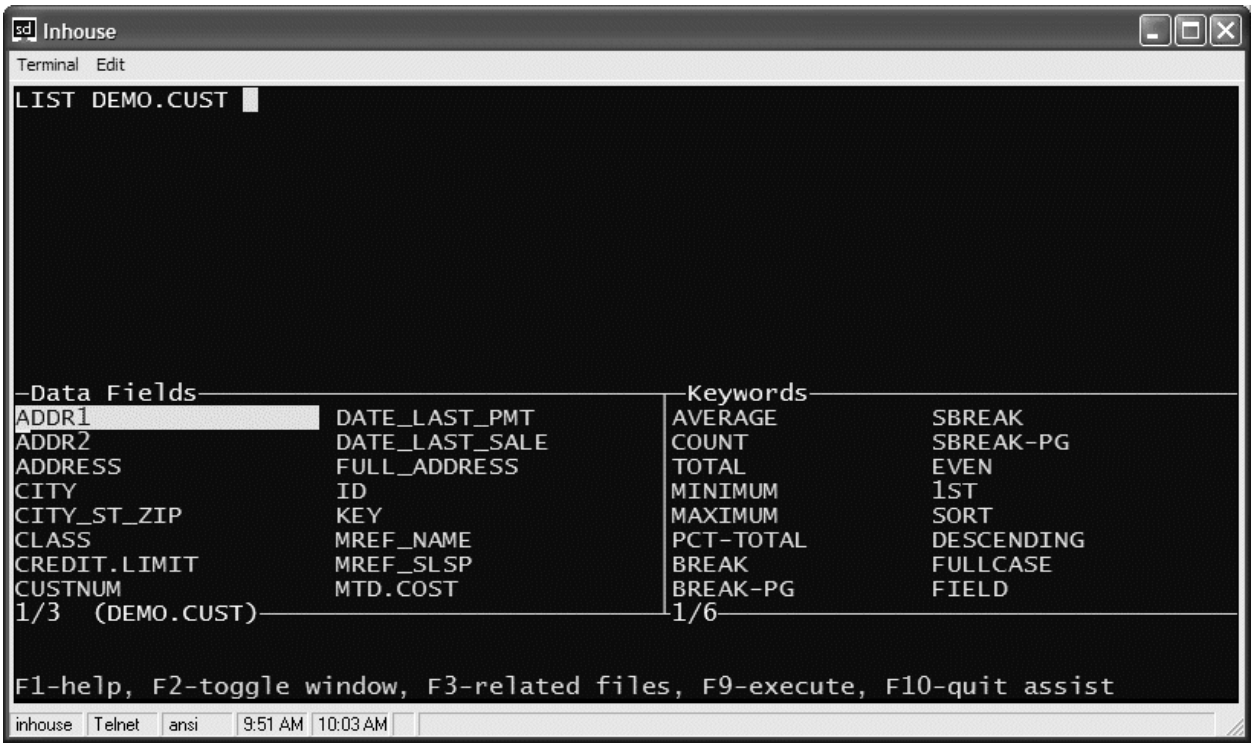
Expressions

Either side of a select phrase relational expression can contain simple math features, by using the operators +, -, *, and / to perform addition, subtraction, multiplication, and division, respectively.

For example, to select just sales figures at least as high as the cost times 1.5, enter this:

```
YTD.SLS > YTD.COST * 1.5
```

Assist Mode



When in List Mode, GENERAL provides a function key option to invoke Assist Mode. This mode provides a split-screen "pick and point" method of generating a LIST command.

The first thing Assist does is check to see if there is a file name specified in the LIST command. If not, it will prompt for one. Upon selection of a file, GENERAL will load the file's dictionary and begin the pick and point session to add fields and keywords to the command.

Assist Screen

Assist splits the screen into three regions:

- A command window in the upper half of the screen
- A field window in the lower left quarter
- And a keyword window in the lower right quarter

In the command window is a reverse video block to indicate where the next selection will be placed. Normally, it is placed at the end of the command.

Initially, the cursor is placed in the field selection window on the first field. By moving the cursor around with the cursor motion keys, pressing the page up or page down keys, or by pressing the first letter of the field desired, you can highlight the desired field and press <Return> to put the field into the LIST command.

To choose a keyword, move the cursor to the right, out of the field window. Then the same cursor motion keys can be used, to choose the desired keyword. Pressing <Return> on a keyword may place the keyword into the command, or it may cause GENERAL to issue secondary prompts to get parameter information.

For example, if you highlight the TAB keyword and press <Return>, GENERAL will prompt for a column and/or row position.

Assist Options

Assist allows toggling between the command window and the field and keyword windows, by pressing **F2-toggle window**. You can edit the command while in the command window. The insert point will be placed where the cursor was left, when returning to the field or keyword window.

As fields and keywords are added to the LIST command, GENERAL maintains an "undo" buffer, so that the character delete keystroke (typically Ctrl-X on Unix, Delete on MS-DOS) will remove elements, most recent first.

Related files

Assist mode can develop link field definitions with the **F3-related file** key.

GENERAL will prompt first for a file to link to. The dictionary of the LIST file determines the files available. Upon selection of a file, the field window changes to show the new set of fields from the currently selected file.

GENERAL remembers what file was originally chosen, so pressing the related files key again will place the original file first on the list, and restore the original field list if it is chosen.

When in a related file, choosing a field places a link field definition in the LIST command.

Executing the report

When the command is complete, press the **F9-execute** key to start the report. GENERAL will invoke the command processor to generate the proper report code, and also place the LIST command in the command history.

To quit assist mode, and return to full-screen command entry, with the command created thus far, press the **F10-quit assist** key. From there, regular command text editing is possible, or another press of the **F10-exit** key will abort the command and return to the main menu.

VDT Output Control

Buffered Output

When GENERAL prints to the VDT (by default, or when the ON keyword specifies the VDT device), pages are stored on disk as they are printed, so that the user can easily page up and down through the report using the PgUp and PgDn keystrokes. GENERAL knows when a particular page has been already printed, and quickly re-displays those that have.

Other options available when a report is being buffered to disk are:

F2-first page
displays the first page of the report, setting the next page pointer to Page 2.

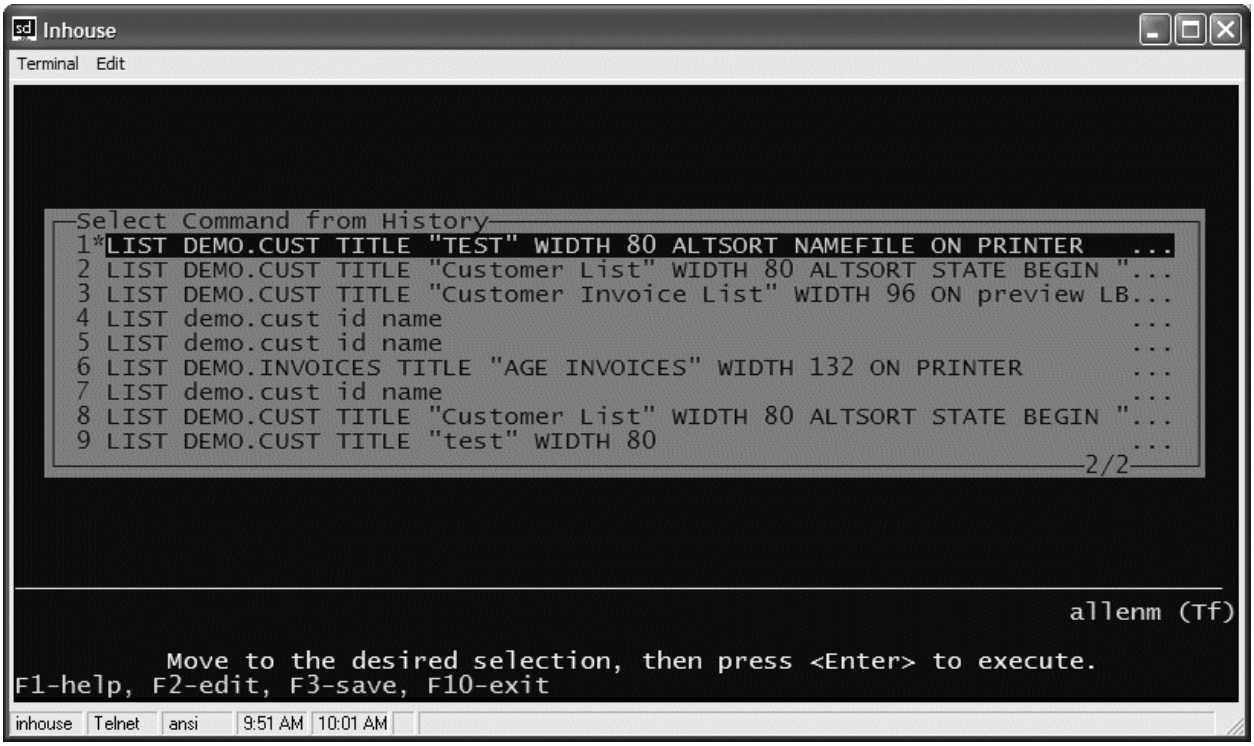
F3-last page
completes the report to disk, then displays the last report page, excluding the recap page. Note that this option can take a lot of time if the report is large. At any time, pressing the <Interrupt> key will stop the search for the last page, and print the page being produced at the time the interrupt is requested.

F4-search
issues a prompt for search text, and processes report pages, starting with the next page, until a page containing that text is found. Like the "last page" search, pressing the <Interrupt> key can stop the text search.

No buffering

As the disk file used to save a large report as it is being displayed can get quite large, the NO-WKFL keyword can be used to turn off buffering. When the keyword is used, the user can only proceed forward through the report, one page at a time.

STACK Command History



While in GENERAL and executing LIST commands, the commands are being saved in a history list, available for immediate recall.

The commands are placed on the top of the list each time they are executed, and the last item on the list drops off. In memory, it looks like a stack, hence the name:

command-1
command-2
command-3
...
command-max##

When exiting GENERAL, the stack is saved under the user ID logged in with, so that the next time that user ID logs into GENERAL, the stack is restored. Note that this can cause confusion if several people use the same ID.

Stack Options

The stack is displayed in a selection window, on one or two pages. The item desired is chosen by moving the highlight with the cursor motion keys, the page up/down keys, or by pressing the desired number. Commands longer than 70 characters display with "..." at the end of the line. Such truncated commands are fully displayed once selected for an operation (see Command Verification, later in this chapter).

Once the desired command is highlighted, choose the desired option:

<Enter> runs the command.

F2-edit invokes the command text editor, so the command can be modified. When you exit the editor, the modified command is placed back on the stack.

F3-save prompts for "saved command" information, and permanently stores the command under a name. That command is then available for processing by the RUN command described later in this chapter.

F10-exit returns to the main menu.

Command Verification

If the command is too long to display in the selection window, then GENERAL displays it in a full screen window before continuing with the selection option for processing. Additional pages of the command can be viewed by pressing cursor motion keys at the verification prompt. Choosing No will re-display the stack selection list. Choosing Edit will invoke the LIST command editor.

Saving Commands

If the Save option is used from the stack command selection window, GENERAL prompts for information about the command, so that it can be stored under a name and recalled later by the RUN command.

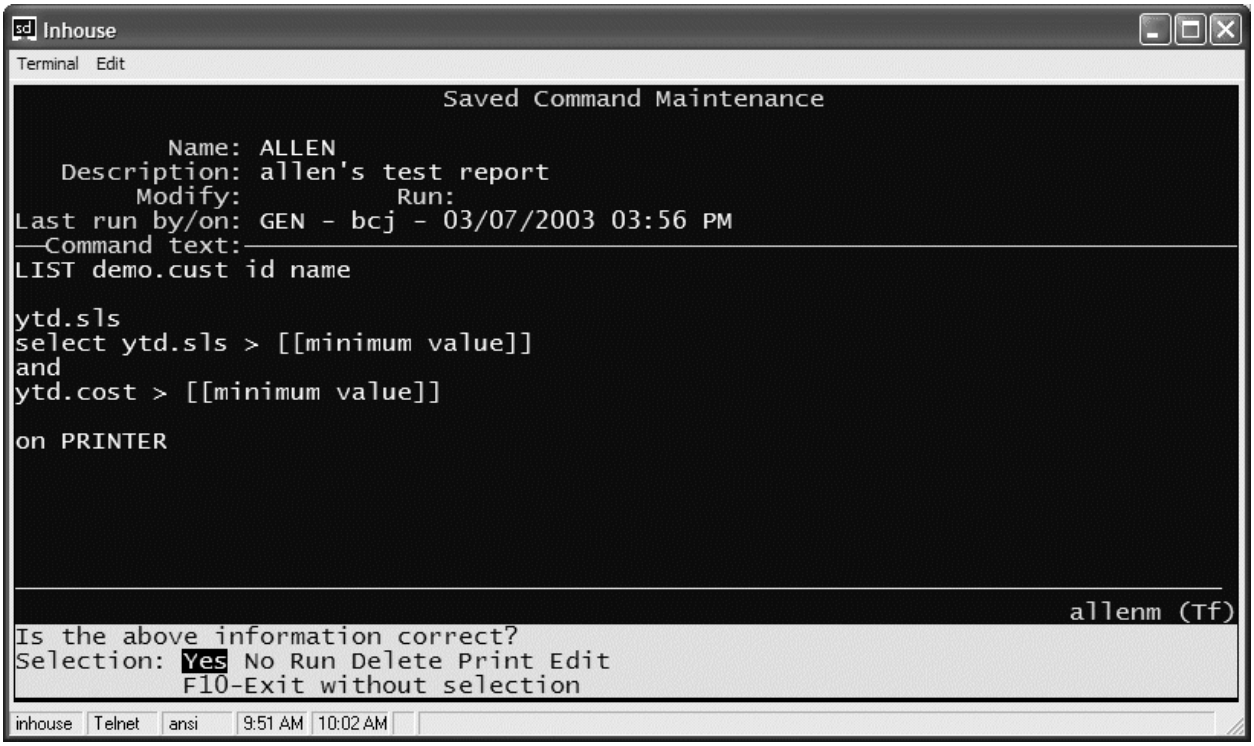
Name
is used to identify the command. It can contain from 1 to 20 letters, digits, and underscores. Spaces are not valid.

Description
is used when listing saved reports in the RUN command.

Open/Private
Open means any user can execute the saved command from the RUN menu, so long as that user has an access level appropriate for the files used by the report.

Private means that only the user who saved the command can execute it from the RUN menu. The user is identified by the login ID used to access GENERAL.

RUN – Edit or Re-Run Saved Commands



A saved command is a permanently stored LIST command, available for re-execution or editing with the RUN command. Generally, the commands are saved from the command stack, using the Save option of the STACK command. However, since RUN is a standard maintenance program, it is possible to create new LIST commands directly within the RUN maintenance screen.

Saved commands can be especially useful when used with run-time replacements. For example, a report that processes a specific range of invoice dates could be modified to prompt the user for the dates. Then, when executed from the RUN command, a different date range could be specified from the same report definition.

Note that saved commands are not the same as PROMPT defined reports. A saved command is a stored LIST command, while a PROMPT report is a stored report definition used to generate a LIST command.

You can maintain the following fields:

Name

Enter the name of the saved command. The name can be from 1 to 20 letters, digits, and underscores. If the name doesn't exist, GENERAL will prompt for verification that it is a new name, and allow the copying of an existing saved command.

To choose the command from a selection window, press the **F2-list** key.

To return to the main menu, press the **F9-exit** or **F10-exit** key.

Description

Enter a description for this command. This is used in the selection list window to help identify the saved command.

Last run

Is a display-only field, indicating the last time this command was run, and who ran it. In this case, the "who" is the operating system login name.

Modify

Enter the minimum user access level required to modify this report design. A value of 0 allows any GENERAL user to modify the design.

Run

Enter the minimum user access level required to run this report design. If a user has an access level of 5, and a report design has a Run level of 6, the user could not view, run, or modify the report.

Command text

This is a text-editing window that will allow editing of the LIST command text.

Verification Options

Yes	File the saved command as displayed.
No	Modify the description or the command text.
Run	Run the LIST command associated with the saved command.
Edit	Invokes the full screen LIST command editor that offers assist mode, making it easier to modify the LIST commands from the RUN screen.
Delete	Delete this saved command. GENERAL will verify the deletion before actually performing it.

DICTIONARY

The GENERAL dictionary module provides the link between the report writing tools and the data files created and maintained by the application software with which GENERAL is installed. The dictionary describes the characteristics and locations of each element of data that GENERAL may use.

In addition, the dictionary can be used to define calculations and inter-file relationships in order to make report generation easier. By placing a calculation or lookup in the dictionary, there is no need to recreate the coding required to generate information, each time it is needed.

External Dictionaries

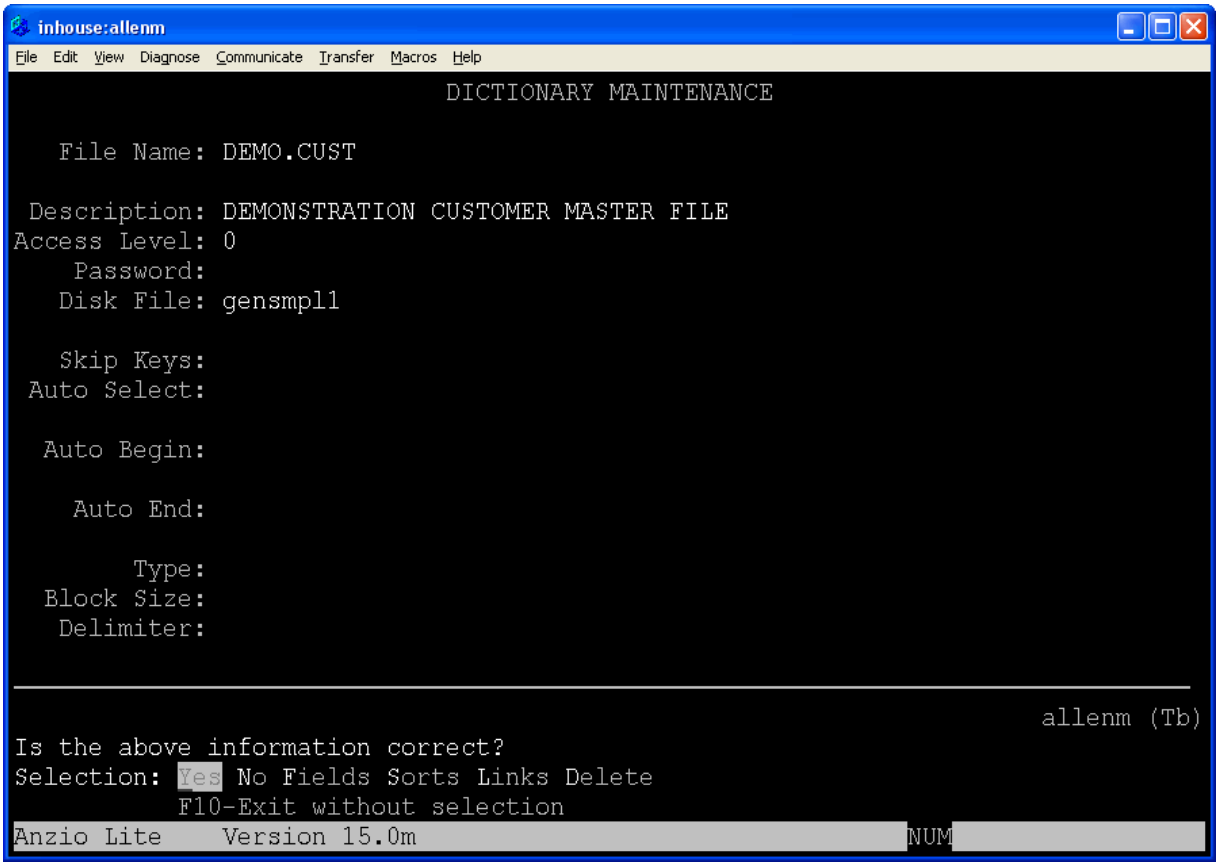
Note: if you define a file in the GENERAL dictionary, it will NOT be recognized in the external dictionary.

Basis Data Dictionary: Files that have been defined in the Basis Data Dictionary do not need to be defined in the GENERAL dictionary. In order for GENERAL to use these definitions, the extended utility **_acu.utl** must have been CALLed prior to execution of GENERAL. This utility establishes several global string table elements that are looked for by GENERAL upon execution, which if set will point to the Basis dictionary.

Once the Basis dictionary is active for GENERAL, any report generation jobs will automatically look in the Basis dictionary if the GENERAL dictionary for the file isn't present. The Basis definitions are interpreted at run-time without any additional action required on the part of the user.

Filix: GENERAL can detect and properly report from Filix data files. In order to detect a Filix file and find its dictionary, GENERAL must be executed from within Filix, as a PROGRAMS menu option, or the Filix data files and program directory must be in the prefix.

Header Definition



File Name

Enter the name of the file to maintain. To create a new file definition, just enter a new name. File names in GENERAL may be from 1 to 20 letters, digits, periods, or underscores. The name should start with a letter, and no two periods or underscores may be adjacent.

AR.CUSTOMERS is a valid name.
AR__CUSTOMERS is invalid, since there are two underscores in a row.

Note that the GENERAL file name is just a synonym for a physical file. It does not need to match the name of a disk file. In fact, any number of GENERAL file definitions can reference the same physical disk file, and a single GENERAL file can reference several disk files.

When a file is defined in a supported external dictionary, such as Taos or Filix, it **should not** be defined in GENERAL's dictionary. If it is, then a complete definition for the file must be established, with all fields defined. The external definition will be ignored.

Select from list

In order to choose the file from a list window of files, press the **F2-list files** key, or enter a wildcard or the first few letters of the name you are searching for.

Wildcards can consist of letters and digits, and special search characters: *, ?, and [x-y]. The * character matches any number of characters (including no characters), the ? Matches any single character, and [x-y] matches a single character that falls in the range x..y. Note that the bracket notation may not be available in all implementations.

AR*SLS will find files starting with AR and ending with SLS.

AR??SLS will find files starting with AR, followed by any two characters, and ending with SLS.

AR[0-9][ABCD]SLS will match files starting with AR, followed by a digit, then followed by a letter A,B,C, or D, and ending with SLS.

These wildcard schemes match the Unix shell wildcards, and closely match the MS-DOS wildcards, except for the [x-y] ranges.

Hard copy of Dictionary

To print the dictionary, for one or a range of files, press the **F3-print files** key. GENERAL will prompt for various options, including a file name range, detail level, error checking, and printer, and produce a hard copy dictionary listing.

Note that the error checking option attempts to compile each field expression, so a hard copy listing may run a great deal slower than if simple print options are selected.

Exit to menu

To exit back to the main menu, press the **F10-exit** key.

Description

The description of the file is used for only for list mode windows, to assist in identifying the file.

Access Level

The file access level is used for security purposes, to restrict access to this file to users with an access level at least this high. A GENERAL administrator defines user access levels with the USER command.

Valid values are numbers from 0 through 9. A level of 0 allows any user to report from this file, while a 9 would only allow level 9 users (administrators).

Password

The password field is used for files that are encrypted and require a password to open. This value can be a literal, though this is not very secure, since it is displayed on screen. More commonly it will be a run-time variable or run-time replacement.

For example, if the application stores a file password in a global string (i.e. x$=stbl("filepass","password") before running General, then the run-time variable reference **[[@filepass]]** would retrieve the value at run-time. Note that global strings must be assigned by a startup parameter program in the GUI server to enable support for GUI clients.

Another option is to use a hidden text run-time replacement, similar to this:

[["Enter password for ARCUST file","H"]]

Note that an alternative exists to password handling within the General dictionary. You can create a custom program to perform the OPEN, and specify that program rather than the actual disk file in the Disk File field. To use this feature, the Password field should be blank. See the Disk File field documentation, below, for the specification of the custom program interface.

Disk File

The disk file indicates to GENERAL what file(s) to open and read when producing a report for this file.

Valid formats include:

File name only	**AR1CST**
Full path name	**/usr/data/AR1CST**
Multiple files	**ORDLIVE;ORDHIST**
Run-time replacements	**/u/AR[[Enter Company Code]]/CUSTOMERS**

If a full path name is entered, then just that specific file is opened. This may make a dictionary site-specific. If a simple file name is entered, GENERAL will allow the Basic file search methods, such as a search prefix, to be implemented.

If multiple files are entered, they must be entered as a file chain delimited with semi-colons. When GENERAL produces a report from a file chain, it uses the same field definitions for each physical file opened, so the files should have the same structure, although some conditional control can be used with the @FILECHAIN variable in expressions.

When file chains are targets in a full-key cross reference calculation (an XREF() function), then each file in the chain is read *only until* a record matching the XREF key specification is found. When they are targets of a partial-key cross reference (MREF(), TREF(), AREF(), and CREF() functions), then each file is read in series until all files have been scanned for records matching the partial key specification of the function.

File types supported by GENERAL include the intrinsic keyed types of the host Business Basic (DIRECT, SORT, MKEYED, etc.), INDEXED, and ASCII text. ASCII text files can contain either line-feed terminated records with delimited or fixed-position fields, or fixed length (blocked) records with fixed-position fields.

If the disk file specified is determined to be a program file rather than a data file, then General interprets the file as a program that should be CALLed to open the data file. The purpose of this feature is to allow a custom program to be used to open a data file, for example to allow password controlled files to be opened in a protected way, via an encrypted program.

The program must be defined with the following ENTER structure:

ENTER FILENAME$,CHANNEL,ERRMSG$

- The FILENAME$ value is passed to the program by General, and will hold the name of the General dictionary file name. Do not change this value.
- The CHANNEL value is also passed by General, and is the channel number that must be used to open the data file. Do not change this value.
- The ERRMSG$ variable will be null when the program is CALLed, and the custom program may set it to a non-null value if any error occurs. If ERRMSG$ is not null when the program exits, General will display the error message and stop processing.

Skip Keys

The skip keys entry is used to identify specific keys that should not be included in a report from this file. Some application software uses special key values to store odd information, such as the "next order number", or the "last run date". This field can be used to tell GENERAL what specific keys in a file should be ignored.

The format of this field is:

*delimiter*key-1*delimiter*key-2*delimiter*

For example:

~01~02~03~

uses the tilde as the delimiter, and tells GENERAL to skip the keys "01", "02", and "03".

!~001! !
uses the exclamation point as a delimiter, in order to allow the use of GENERAL's ASCII notation to skip an ASCII value 1 key, as well as a space key.

Note that skip keys do not provide the same flexibility as an auto select phrase would (such as partial matches or ranges, or data references), but they involve less overhead, so should be used when possible.

Auto Select

The auto select phrase allows the specification of a select phrase fragment that will be implemented each time a report uses this file. This feature provides the means to logically segregate specific record sets from a file that contains multiple record formats.

For example, if a file contains both order heading and order line information, a GENERAL file definition could be set up for each: one called ORD_HEAD and the other called ORD_LINE. Often, the application creating the orders will indicate a header record with a line number of 0, while the order lines will have line numbers from 1 to 999. A field called LINE_NO could be defined in each file, and the ORD_HEAD file would have an auto select phrase:

```
LINE_NO = 0,
```

While the ORD_LINE file's auto select phrase would be:

```
LINE_NO > 0.
```

When an auto select phrase is present and the user enters criteria in a select phrase, then the two phrases are implemented so that records must pass both selection criteria.

The auto select criteria is also used when the file is the target of a partial-key cross reference expression (such as a TREF() function).

Auto Begin/Auto End

The auto begin and auto end phrases define a key range to always process when this file is used as the primary file for a report. A common use for these fields is to segregate data that has a company code as a prefix to each key in the file. An auto key range that specifies a run-time prompt or run-time variable can be used to specify a specific company range to process.

If the user also enters a BEGIN or END criteria for a report, then GENERAL appends the user's criteria to the auto range, so that the user specification works within the range established here in the dictionary.

Type

If the file(s) specified, as disk files are text files, rather than Business Basic files, then the type of text file must be specified.

1. **Blocked** files are files with no record delimiters. Instead, a record is determined by reading a specific number of characters from the file. The number of characters is specified by the block size element. All field definitions must use the @REC data reference, as there are no field delimiters.

2. **Delimited** files have records terminated by line-feed characters (on MS-DOS, the record terminator is a carriage-return/line-feed sequence). In a delimited file, there can be fields defined within the record by delimiter characters, or all the data can be stored in a single "field". In the latter case, the primary difference between a blocked and delimited text file is the use of a record terminator rather than a block size. References to delimited fields can be by the @PF*n* physical field variable, or the @REC record variable.

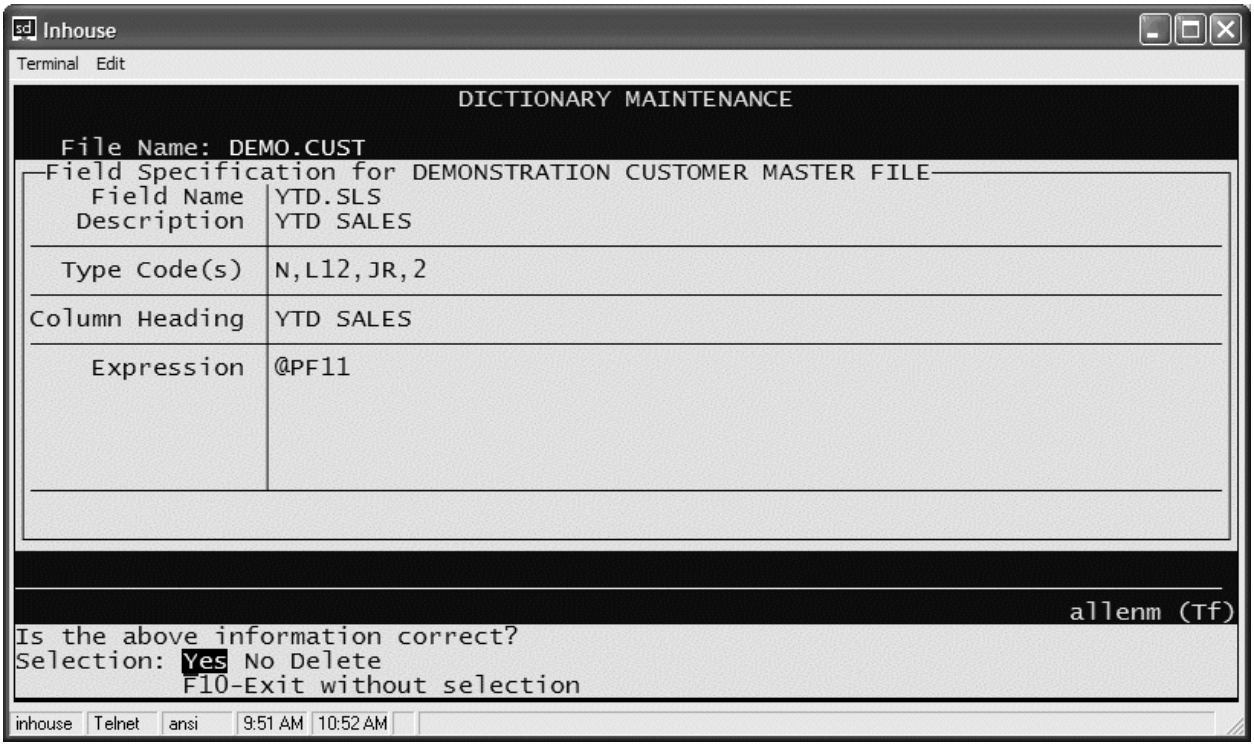
Block size

For an ASCII text file defined as "blocked", above, enter the size of each logical record within the file. Each record is read as a fixed block of data within the file.

Delimiter

For an ASCII text file defined as "delimited", above, enter the character to be used as a field delimiter. If, for example, the file contains commas between fields within each record, enter ",". If a non-printable ASCII character is used as a delimiter, it can be entered using GENERAL's ASCII notation ~nnn. For example, enter **~009** to indicate a <tab> delimiter character.

Field Definitions



Data Field Specification

Data fields are defined in GENERAL's dictionary with type codes and expressions. The type codes indicate what type of data this element contains, and information about how to display or format the data. The expression indicates where the data is in each record, or defines a calculation or manipulation of one or more data elements.

In addition to the type codes and the expression, a field definition also contains a heading specification, which GENERAL uses to produce default column headings for a report.

Field Name

Field names in GENERAL may be from 1 to 20 characters long, and may include letters, digits, underscores, and periods. Underscores and periods must be individual characters. Spaces are not allowed.

- **INV_CODE** is a valid name.
- **ADDRESS1** is a valid name.

PROD..CODE is invalid.
CUSTOMER NAME is invalid.

Description

The description of a field can be any text. It is used in field selection windows to help identify the field.

Type codes

Enter a valid *base* type code, followed by a length code, followed by one or more optional format codes separated by commas. Valid base type codes include:

T text data
N numeric data
D date data, in the host language internal date format

Column width

A length code is simply an **L** followed by a number.

T,L30 indicates a 30 character wide text field.
N,L14 indicates a 14 character wide numeric field.

Length can also be specified with a run-time replacement, such as **L[[@CUSTLEN@VDT]]**.

Additional codes

Each base type supports additional format codes.

Text Format	Description
N	The number of lines this field should occupy. The default is 1. The maximum is 99.
JL, JR	Justification for this field. The default is left.

Number Format	Description
n	The decimal precision for this field. The default is 0 (resulting in integers), the maximum is 9.
P	Punctuate this number. This results in a thousand's separator being used: 1,500 instead of 1500.
-,(,CR,$	Mask options: "-" for trailing minus sign, "(" for negative parenthesis, "CR" for negative CR, "$" for currency format. These options are mutually exclusive.

Number Format	Description
Sn	Scale factor of n, meaning that GENERAL should shift the decimal point automatically by this amount.
JL, JR	Justification for this field. The default is right.

Date Format	Description
T	Display with time. Assumes internal date number is a minimum precision 4 number, with time stored as a decimal portion of a day.
JL, JR	Justification for this field. The default is left.

Type Code Examples

T,L30	A simple 30-column wide text field.
T,L30,3	A 30-column, 3-line text field.
T,L10,JL	A 10 column text field, force left justification.
N,L14	A simple numeric integer, 14 columns, using default (right) justification.
N,L14,P,2	A precision 2 number, 14 columns, right justified by default, punctuated. The format will be like this: **-99,999,999.00**.
N,L12,P,2,S2,JL	A precision 2 number, 12 columns, punctuated, left justified, with the decimal point shifted left 2 positions. A number stored on a disk as 100005 would be output as **1,000.05**.
N,L12,P,2,(	A precision 2 number, 12 columns, punctuated. If negative, the number will be surrounded by parenthesis.
N,L9,P,$	A numeric field to be output with currency symbols (a "money" field). The configuration module controls currency formatting.
D,L10	A date field, left justified in 10 columns. The data provided by the expression must be a Julian number, based on the internal date format specified by the underlying language.//Any Business Basic that provides "date" number types has functions to produce Julian numbers from standard values. In BBx, the function is JUL(year,month,day), for example. The expression for a date field must create a Julian number in order for GENERAL to properly display the value.
D,L18,T,JR	A right justified date, 18 columns, with the time printed after the date. The format of the time is controlled by the configuration module, and defaults to **HH:MM PM**. GENERAL treats time as a decimal fraction of a day. The fraction .5 is 12:00 noon; .75 is 6:00 PM. If a number is stored in precision 4, then the time can be resolved to the minute accurately.

Guessing the Definition

You can have GENERAL guess what the data definition should be, including both type codes and the expression, by pressing **F2-guess from data**. GENERAL will invoke the data assist window, so that data from the record can be selected and marked. Once selected, GENERAL attempts to determine if it is text, number, or date.

- If it is non-numeric, it is assumed to be text.
- If it is numeric, and 6 or 8 digits, GENERAL tries to create a date out of it, first in MMDDYY[YY] order, then in YY[YY]MMDD order. If not successful, GENERAL treats it as an integer. Otherwise, GENERAL creates a date field, with an expression modifier that converts the digits into an internal date value.
- If it is an integer, and the value of the integer is within 365 days of the current Julian date, GENERAL assumes that it is a Julian number, and creates a date field.
- Any number not converted to a date is converted to a number, with precision and length based on the number itself.

Obviously, if you request GENERAL to guess at the data, you should select data that is as representative as possible of the data in the file. Also note that when GENERAL guesses a data definition, the heading element is set to the field name automatically.

Column Heading

The column heading specification is used to define the default column headings used when reports reference this field. This will normally be some descriptive text or abbreviation.

The width of the column heading may override the display width used by GENERAL when producing a report. If the length type code is **L2**, but the heading is **STATE**, the field will occupy 5 columns rather than 2.

To create a stacked heading, use the vertical bar (|) to delimit the heading portions. **CUSTOMER|NAME** will produce a column heading like this:

CUSTOMER
NAME

In a stacked heading, the longest segment is used to determine if the column width needs to be adjusted. Also, in a right justified field, each segment is right justified.

Expression

The expression is used to describe to GENERAL where the data for this field is located within a record, and if necessary, how to manipulate it to produce the desired output.

Physical data from the record can be referenced in two ways. If the record is stored with *field separators*, then GENERAL interprets the separators as physical field delimiters. To reference a physical field, use **@PF*n***, where *n* is the field number, starting with 1 and going up to 255.

In a record structure like this:

00100<sep>**JOHN SMITH & COMPANY**<sep>**123 1ST STREET**<sep>**LOS ANGELES**<sep>...

The physical fields would be as follows:

@PF1 00100
@PF2 JOHN SMITH & COMPANY
@PF3 123 1ST STREET
@PF4 LOS ANGELES

The second method to access data in a record is to treat the entire record as one element. In this case, use **@REC**, and use Business Basic sub-stringing to access the desired portions of the record.

For example, **@REC(11,30)** would reference data beginning at position 11, for 30 characters.

Often times, an application will use *logical fields* as well as physical fields. For example, there may be a physical field used to store several codes, or a series of dates. To reference the data in those logical fields, sub-stringing is again necessary. **@PF2(7,6)** would be used to reference data beginning at position 7 of physical field 2, for 6 characters. When both position and length values are given, GENERAL can ensure that the data will always be padded to accommodate the full length required, so sub-string errors (normally error 47's) won't occur.

The expression field provides some function key options that can be of assistance when creating expressions.

F2-paste Search for and insert the following data:

- Fields already defined in the current file.
- Other file names (quoted, for inclusion in cross-reference functions).
- Fields in other files (not quoted, but again for inclusion in cross-reference functions).
- User defined functions.

F3-data assist

provides a data view and selection window, where records from the file being defined can be displayed, and fields and portions of fields can be selected and pasted into the expression.

Data assist mode will also offer an option to see what data field definitions each physical field uses.

F4-expand
opens a larger window for entry of expressions longer than 300 characters. In this window, expressions can be as long as 900 characters.

Converting data types

Both the @PF*n* and @REC functions reference data as text. For text fields, this is proper, but for numeric or date fields, the data must be converted to a numeric type. In simple cases, where the function stands by itself, or has only simple sub-stringing, GENERAL can implicitly convert the functions to numeric, by automatically using a NUM() function. If, however, the expression is more complex, then functions to convert the data to a numeric type must be explicitly entered.

Sometimes, the data can't be converted to numeric using the NUM() function. For example, if an application writes a record position with the BIN() function, which converts a number to a binary text representation, the DEC() function must be used to extract the data. The expression for this might look like this: **DEC(@REC(200,4))**.

A date field might be stored in a physical field as text formatted in MMDDYY format. In order for GENERAL to recognize it as a date, it must be converted to a Julian number. In BBx, the function to convert regular data to a date is the JUL function. That function requires numeric arguments for the year, month, and day, in order. To convert from the MMDDYY to a date, this BBx expression would work: **JUL(NUM(@PF2(5,2)),NUM(@PF2(1,2)),NUM(@PF2(3,2)))**. Frequently used expressions taking similar arguments can be established as user-defined functions, which are described in more detail later.

Constants

Fixed values can also be referenced in expressions. Where a numeric is required, the value can be an unpunctuated number, such as **1.5**, or **1000**. If text is required, then it must be quoted, like **"Amount: "**, or **", "**.

In the date example, to add 30 days to the date, the expression would be:
JUL(NUM(@PF2(5,2)), NUM(@PF2(1,2)),NUM(@PF2(3,2))) + 30.

Other fields/Internal functions

Other field definitions in the file can be referenced by their names. When referenced by name, a field takes on the data type of its definition, so numeric fields can be referenced as numeric values, text as text values, and dates as Julian numbers.

For example, to define a gross profit calculation, where two other fields in the file are **SALES** and **COST**, the expression would be: **SALES - COST**. To define a gross profit percent calculation, which is just a little more complex, the expression would be **100* (SALES - COST)/SALES**. To avoid a division by zero error, further steps could be taken, using a constant, or using GENERAL's internal function to divide numbers without an error, like this: **100 * DIVIDE(SALES-COST,SALES)**. DIVIDE() is an internal GENERAL function that can be used by field expressions just like an internal Business Basic function.

GENERAL functions can make an expression more readable, or can perform some feature that may not be available in an intrinsic Basic function. However, there are some differences between a GENERAL function and an intrinsic function:

- A GENERAL function may perform more slowly than a Basic function. For example, the GENERAL DT() function is much slower than the BBx JUL() function, although both can produce the same result.

- A GENERAL function isn't syntax tested until a report attempts to execute it, whereas a Basic function is syntax tested when the expression is entered in the dictionary.

Functions, like field references, are classified as either text or numeric. The DIVIDE() function is an example of a numeric function: it returns a numeric result. An example of a text function is the TRIM function, which trims spaces from data:

```
TRIM(CITY)+", "+STATE+" "+ZIP_CODE
```

A complete list of GENERAL functions is provided in a table, later in this chapter.

The current field can be referenced by its name or by the name **@FLD**. This can be a handy shorthand reference in compound expressions or conditional expressions, described later.

Conditional Expressions and Compound Expressions

Conditional and compound expressions take on the look of true Business Basic statements. In fact, that is exactly what they are.

In the examples earlier in the chapter, where simple references are made to physical data and other fields, there is an implied assignment inserted in the expression at run-time: **@PF2(1,30)** implicitly becomes **@FLD=@PF2(1,30)**, so that the data reference gets assigned to the field being defined.

If more control is required, with several statements, or IF..THEN..ELSE logic, for example, then the Basic statements can be entered directly.

The following example shows both compound and conditional expressions to control the calculation of a gross profit percentage:

@FLD=0;IF SALES<>0 THEN @FLD=100*(SALES - COST)/SALES; IF @FLD>=1000 THEN @FLD=999.9; IF @FLD <= -1000 THEN @FLD=-999.9

The above expression performs the following steps:

- The field is initialized to 0.
- If SALES is not 0, then set the field to the gross profit percent.
- If the resulting percentage is greater than or equal to 1,000 then set it to 999.9.
- If the resulting percentage is less than or equal to negative 1,000 then set it to -999.9.

Note: Compound statements are parsed differently in GENERAL than they are in Business Basic.

There is a subtle difference between how GENERAL interprets the compound statements and how Business Basic does. In GENERAL, each statement delimited by semi-colons is treated as an individual statement, regardless of whether or not it follows a conditional statement. Consider the following:

IF @FLD>0 THEN @FLD=@FLD+1; SALES=SALES+1

In Business Basic, SALES=SALES+1 would only be executed if @FLD>0. In GENERAL, both statements delimited by the semicolon get executed independently.

If GENERAL is supposed to execute multiple assignments conditionally, then the special-grouping *operators* must be used. The grouping operators are curly braces: {}. The statement could be modified in a GENERAL expression to this:

IF @FLD>0 THEN {@FLD=@FLD+1; SALES=SALES+1}.

Other variable values

References to variables other than dictionary field names, physical field or record references, or the current field variable @FLD, should be made as **@TEMP*name*** references. @TEMP variables can contain up to 15 letters, digits, and underscores following the "@TEMP". They can be used either as string variables or numeric variables, but string variables need to end with a dollar sign.

@TEMP_AMOUNT	Numeric data storage.
@TEMPCSZ$	String data storage.

@TEMP variables can also be defined as arrays, as in Business Basic. The example below shows how an array of @TEMP values could be assigned from sequential positions within a record by using a FOR/NEXT loop.

```
DIM @TEMPMONTH[12];
FOR @TEMPX=0 TO 11;
   @TEMPMONTH[@TEMPX + 1] = NUM(@REC(1+@TEMPX*10,10));
NEXT @TEMPX
```

User defined functions

In the GENERAL configuration, user-defined functions can be created that may be referenced in field expressions. User-defined functions can help streamline expressions, or make a dictionary easier to port from one Business Basic to another.

A user-defined function can be referenced in an expression with the ampersand (&) operator. For example, if a user-defined function CONVERT_DATE has been defined to convert a six-digit text field to a date, an expression using it might look like this:

&CONVERT_DATE(@PF3(1,6))

In the configuration, user-defined functions look just like Business Basic functions that have been defined with the DEF FNx statement. In fact, when a report is executed that references a user-defined function, GENERAL automatically creates the proper DEF FNx statements.

User-defined functions that return text must be defined and referenced with a dollar sign suffix, just as in Basic. **&MASK$(SSN,"000-00-0000")**, for example, shows the function MASK$, which returns a text value.

Run-time replacements

Run-time replacements can be used within an expression as if a constant had been used instead. The difference, of course, it that a run-time replacement's value isn't fixed, it is instead determined at run-time.

To include a run-time replacement in an expression, just surround it with double square brackets.

@FLD=AMOUNT * [[Enter amount factor]] will prompt the user at run-time for an "amount factor", and use that factor in the expression.

@PF1(1,[[@CUSTLEN@VDT]]) would reference the run-time replacement *variable* @CUSTLEN@VDT. This variable would be found on disk, normally maintained by the external application to which GENERAL is linked. This technique can be used to parameterize a dictionary, so that one site, or even one terminal, can use different values in the same dictionary.

Be sure to place quotes around run-time replacements that need to be treated as text. For example:

@FLD=EXPAND("[[Enter Date Code (M,D, or Y)]]","M,D,Y","Month,Day,Year")

Cross-referencing files

Beginning in version 4, there is no concept of a definition type. Instead, the data is given a data type (with the type codes), and an expression describes how the data is constructed.

To create a cross reference, GENERAL provides 5 functions, each taking the same three parameters:

XREF	returns a text value of an exact key match.
MREF	returns a stream of text values from a partial key match.
TREF	returns a number derived from totaling the values of each record in a range defined by a partial key match.
AREF	returns a number, derived from averaging the values as in TREF.
CREF	returns a number, derived from counting the records in the range.

An example of the syntax of a cross-reference function is this:

XREF("AR.CUST", COMPANY+CUSTOMER, NAME)

The first parameter is a file name, in quotes. The file must be defined in GENERAL's dictionary (or in an external dictionary supported by GENERAL). This is the *target file*; that is, this file contains the data to be obtained by the function.

The second parameter is an expression defining the key to be used. This can be any simple expression, including functions, quoted constants, field names, @PF*n* references, and @REC references.

The third parameter is an expression used to derive the data from the external record. Again, the expression can be any combination of functions, constants, field names, and data references. However, this expression references data and fields from the target file. If the function is a TREF or AREF, then the data expression must evaluate to a numeric value.

Using a target file's alternate sorts

Another feature of the dictionary, described in more detail later in this chapter, is the Sort definitions for a file. A Sort definition can describe how a file's secondary keys are used, or how a related "sort file" is used by the application. When two files are related by a Sort definition, then that sort definition can be specified as part of the file name parameter with the **@** sort name operator. **ORD_LINES@ORDER_NUM** for example, would look for a Sort definition called ORDER_NUM in the file ORD_LINES. That sort may define a secondary key that sorts the order lines by order number. When GENERAL is searching for records for a TREF function, for example, it could process the records in the ORD_LINES file by the ORDER_NUM key, returning its result from just those records in a given order number.

Iteration and key suffix ranges

In the key parameter, a series of values can be used as either a prefix or a suffix. As a suffix, a range of values can be used. The iteration delimiters are curly braces within a literal portion of a key parameter, like this:

TREF("SLSHIST","{01,02,03}"+CUST_NO,AMOUNT).

In the above example, a prefix iteration is used to cause GENERAL to search records first with "01"+CUST_NO, then "02"+CUST_NO, and finally, "03"+CUST_NO. This can be useful to consolidate values across several companies, when a company code prefixes each record range.

An example of a suffix range might look like this:

MREF("ORDER_LNS",ORDER_NO+"{A..C}",DATE).

In this example, a suffix range is used to process all records in a range from ORDER_NO+"A" through ORDER_NO+"C".

Only one iteration operation can be used in any key parameter.

Links and Cross References

GENERAL has an additional ability to process data from related files. While cross reference functions are useful for creating specific field definitions, the Link specification for a file can be used to reference any field in a related file at run-time, using special "link notation" for a field.

If an ORDER file has a link specified which provides the key required to access CUSTOMER file data, then in a report from the ORDER file, the data name "CUSTOMER:NAME" would implicitly generate a cross reference lookup to the CUSTOMER file to get the NAME. Links are defined in the Link option of the dictionary, which is described later in this chapter.

Direct reading of other files

Expressions can include statements to open and read other files directly, so that the cross-reference functions don't need to be used. This feature can be used to save memory, as the target file's dictionary doesn't need to be parsed at run-time. It also can be used to access files that are not defined in the dictionary. The functions supported are:

OPEN	is just like the Business Basic OPEN statement. Use a @TEMP numeric variable as a channel number, which is typically set to the UNT (next available unit) if not previously set. If GENERAL has been CALLed, these open channels will not be closed automatically.
READ	Reads data into temporary variables. There are three read options, described below.
READRECORD	reads a whole record into a single temporary variable.
,KEY=	specifies a key on a READ or READRECORD. Note there can be no spaces in the option.
,IND=	specifies an index on a READ or READRECORD.
,ERR=NEXT	drops through if an error occurs on a READ or READRECORD.

Here's an example:

```
IF @TEMPUNT=0 THEN
  {@TEMPUNT=UNT; OPEN(@TEMPUNT)"/u/xyz/file" } ;
  @TEMPY$="";
  READ(@TEMPUNT ,KEY=CUST_NO ,ERR=NEXT)@TEMPX$,@TEMPY$ ;
  @FLD=@TEMPY$
```

External CALLs

External public programs can be CALLed from within an expression. Normally, the arguments would be data field names and temporary variables. Note that data field names are passed to the program by value, not by reference. This means that their values can't be directly modified by the public routine. Instead, the routine should modify a temporary variable, then assign the data field to that temporary value after the CALL.

```
CALL "myprogram",CUST_NAME,@TEMPNAME$;  CUST_NAME=@TEMPNAME$
```

Function Table

Internal Variable	Type	Description
@BREAKn	Text	The current value of the break point of level n.
@DATE	Date	The current date, in internal (Julian) format.
@DAY	Text	The current date, in MM/DD/YY format.
@DTM	Date	The current date and time, in internal format.
@FILECHAIN	Number	The current file chain number. File chains are sequential disk files, delimited by semi-colons, in the dictionary.
@FLD	Varies	The current field.
@KEY	Text	The key of the current record.
@PFn	Text	*Physical field number n. Equates to an IOLIST position, or in the case of a delimited ASCII file, the field number in the record based on the delimiter.*
@PI	Number	3.14159265358979
@REC	Text	The complete record read from the file (READRECORD).
@TEMPname	Number	A temporary numeric variable. May also hold dates.
@TEMPname$	Text	A temporary text variable.
@TTY	Text	The operating system tty device for the terminal.
@USER	Text	The operating system login name.
@VDT	Text	The Business Basic terminal name (FID(0)).
@YMD	Text	The system date in YYMMDD format.

Function	Type	Description
&function(arg-1,arg-2,...)	Number	A user defined numeric function. User defined functions are defined through the system configuration.
&function$(arg-1,arg-2,...)	Text	A user defined text function.
ADPRDB(c,s,l,p,r)	Number	Calculates accumulated depreciation through period p, using the declining balance method. The calculation is based on cost c, salvage value s, life l, and an optional rate r. The rate defaults to 2 (for double declining balance) if not specified.
ADPRSL(c,s,l,p)	Number	Calculates accumulated depreciation through period p, using the straight-line method. The calculation is based on cost c, salvage s, and life l.
ADPRSY(c,s,l,p)	Number	Calculates accumulated depreciation through period p, using the sum-of-years-digits method, using cost c, salvage s, and life l.

Function	Type	Description
AGE(x,y) AGED(x,y)	Number	The age in days from date x to date y.
AGEHR(x,y)	Number	The age in hours from date/time x to date/time y.
AGEM(x,y)	Number	The age in months from date x to date y.
AGEMN(x,y)	Number	The age in minutes from date/time x to date/time y.
AGEY(x,y)	Number	The age in years from date x to date y.
AREF("f",k,exp)	Number	The average of the numeric expression exp, across a series of records in target file f (defined in the GENERAL dictionary or a supported external dictionary), using key expression k. The key expression defines the first portion of the key to the target file. If the target file is specified with a sort (file-name@sort-name) then the sort specified is used when accessing the range of records indicated by the key expression.
AVG(x,y,...,z)	Number	Average of all numbers specified.
BLOCK(txt,size)	Text	Blocks the text stream txt, into word-wrapped lines of the size specified.
CDATE(txt,"map")	Date	Converts the text string txt, into a date, using the format specified by map. The map contains text and place specifiers: DD, DDD, MM, MMM, YY, and YYYY. A date of "03/15/01" would be mapped to "MM/DD/YY". Optionally, the map strings MDY and YMD specify variable length, delimited dates. The date "3-1-01" would be converted to March 1, 2001 by a map of "MDY".
CNT(x,y,...,z)	Number	Count of all elements specified.
CNUM(txt,x)	Number	Converts text string txt to a number, ignoring any punctuation, such as currency formatting or thousands separators. If the decimal character x is specified, then that character, rather than a period (.), is used to denote the decimal point.
CONTAINS(x,y)	True/ False	True if text string x is contained in text string y; false if not.

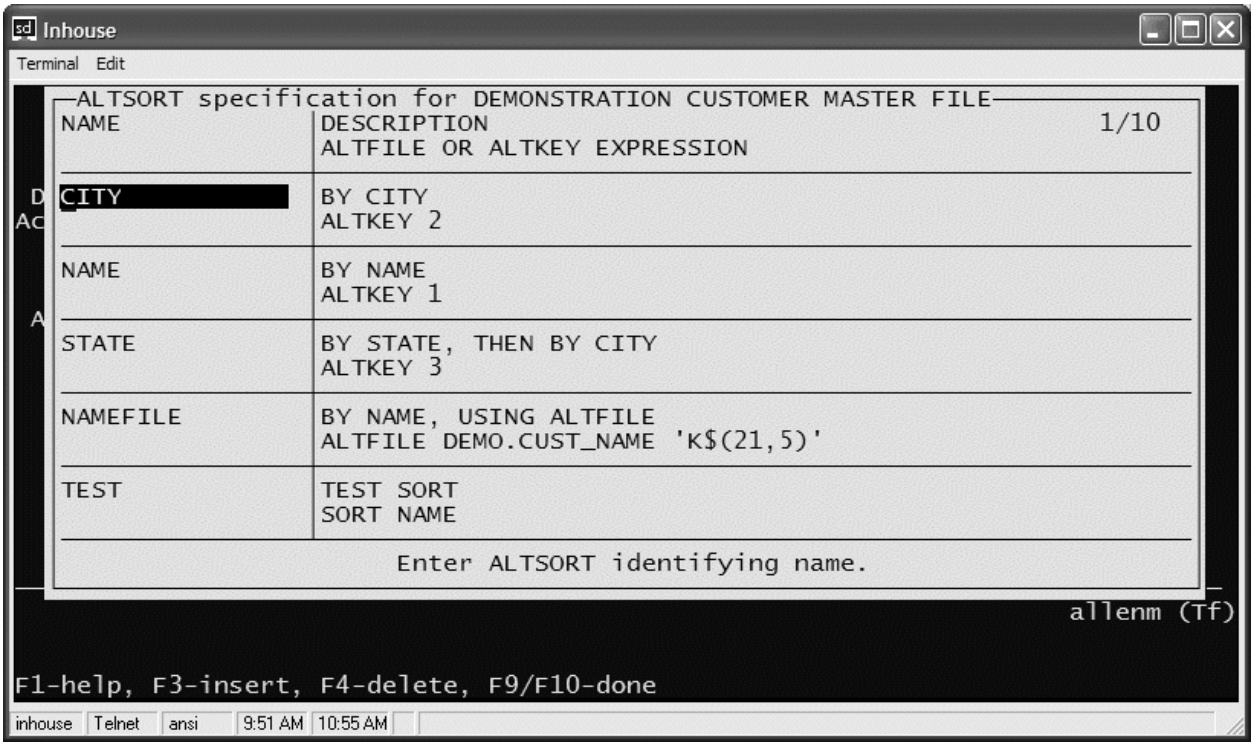
Function	Type	Description
CREF("f",k,exp)	Number	The count of the number of related records in target file f (defined in the GENERAL dictionary or a supported external dictionary), using key expression k. The key expression defines the first portion of the key to the target file. If the target file is specified with a sort (file-name@sort-name) then the sort specified is used when accessing the range of records indicated by the key expression. The expression exp may be any valid data expression.
DAY(x)	Text	The text formatted date of the date value x. The format used is defined in the system configuration, and defaults to MM/DD/YYYY.
DAYTM(x)	Text	The text formatted of the date/time value x. The format used is defined in the system configuration, and defaults to MM/DD/YYYY HH:MM PM.
DD(x)	Text	The day portion of the date x, as two digits.
DDD(x)	Text	The day portion of the date x, as abbreviated (Mon., Tue, Wed, etc.).
DEL(x,y,z)	Text	The modified version of text value x, where the characters beginning with character number y, for a length of z, are deleted from the string.
DIVIDE(x,y)	Number	The results of dividing number x by number y, or 0 if either x or y is 0.
DPRDB(c,s,l,p,r)	Number	Calculates depreciation for the period p, using the declining balance method. The calculation is based on cost c, salvage value s, life l, and an optional rate r. The rate defaults to 2 (for double declining balance) if not specified.
DPRSL(c,s,l,p)	Number	Calculates depreciation for the period p, using the straight-line method. The calculation is based on cost c, salvage s, and life l. The period number is optional, as straight line results in all periods being the same.
DPRSY(c,s,l,p)	Number	Calculates depreciation for the period p, using the sum-of-years-digits method, using cost c, salvage s, and life l.
DT(x)	Date	The internal date format for the text x. x must be in the date entry format as defined by the system configuration, normally "MMDDYYYY". This function is useful to compare an entered date with a fixed date. The year portion of the text may be in 0, 2, or 4-digit format.

Function	Type	Description
DTM(x)	Date	The internal date and time for the text x. x must be in date and time entry format, normally MMDDYYYY HHMM. This function is useful to compare an entered date and time with a fixed date and time.
EXPAND(x,y,z)	Text	Looks for the occurrence of the text string x in the list of values in y, and returns the corresponding element in the list of values in z. If the value of x is not found in y, then the function returns null.
FV(x,y,z)	Number	The future value of a string of payments, assuming x is the payment amount, y is the interest rate per period, and z is the number of periods. The interest rate should be expressed as a fraction: .07 for 7%, and may need to be adjusted to match the rate per period: .07/12 for monthly rate at 7% annual rate.
GAVG(x,y) GCNT(x,y) GMAX(x,y) GMIN(x,y) GSUM(x,y)	Number	The group average, count, maximum, minimum, or sum value of field x, at break level y. Break levels are determined by the order of break points in a LIST command. GSUM(YTD.SLS,1) would return the break level 1 sum of the field YTD.SLS.
IFF(x,y,z)	Text	Tests the condition specified by a relational expression x, and if true, returns text y, else returns text z.
ISNOTNUM(x)	True/ False	True if text value x is NOT a valid number (suitable for processing by the NUM() function).
ISNUM(x)	True/ False	True if the text value x IS a valid number.
LEFT(x,y)	Text	The leftmost y characters of text value x. If x is less than y characters in length, the value is padded to y characters.
LENGTH(x)	Number	The length of text string x, after leading and trailing spaces are removed.
LOWER(x)	Text	The text string x, with all letters in lower case.
MATCH(x,y)	True/ False	True if the text string x matches the regular expression in string y. This function only works in environments that support regular expressions.
MID(x,y,z)	Text	The middle z characters of text string x, starting at position y. If there are not enough characters in x to satisfy the length, the value is padded with spaces.
MM(d)	Text	The 2-digit month number of date d.
MMM(d)	Text	The abbreviated month name of date d.

Function	Type	Description
MREF("f",k,exp)	Text	The values of the expression exp across a series of records in target file f (defined in the GENERAL dictionary or a supported external dictionary), using key expression k. The key expression defines the first portion of the key to the target file. If the target file is specified with a sort (file-name@sort-name) then the sort specified is used when accessing the range of records indicated by the key expression.
PARSE(s,n,d)	Text	Returns the n-th delimited substring field from string s, given field delimiter character d. If s is abc<254>def<254>, parse(s,2,chr(254)) would return "def".
PMT(x,y,z)	Number	The payment amount of an amortization of principal x, at interest rate y, paid over z periods. Interest should be expressed as a fraction (.07=7%) and may need to be adjusted for periods: .07/12 for 7% per month.
PROPER(x)	Text	The text string x, after capitalizing the first letter after spaces and punctuation.
PV(x,y,z)	Number	The present value of a series of payments, where x is the payment amount, y is the interest per period, and z is the number of periods. Interest should be a fraction (.07=7%), and may need to be adjusted for periods (.07/12=7% per month).
RATE(x,y,z)	Number	The interest rate required to fully amortize future value x and present value y over z periods.
RAVG(x) RCNT(x) RMAX(x) RMIN(x) RSUM(x)	Number	The report summary value of field or calculation x. Which function used determines how value is calculated. The calculations are average, count, maximum, minimum, and total, respectively.
RIGHT(x,y)	Text	The rightmost y characters of text string x. If x is not long enough, the value is left padded to y characters.
ROUND(x,y)	Number	The number x, rounded to y decimal places.
STD(x,y,...,z)	Number	The standard deviation of the stream of numbers x..z.
SUBST(x,y,z)	Text	The text string x, after substituting all occurrences of text string y with text string z.
SUM(x,y,...,z)	Number	The sum of all numbers x..z.
SYSVAR("x")	Text	The value of the operating system environment variable x, or null if not found.

Function	Type	Description
TERM(x,y,z)	Number	The number of payments required to amortize future value y and present value z, at interest rate x. Interest should be expressed as a fraction (.07=7%).
TIME(d)	Text	The display format of the time portion of date/time value d.
TREF("f",k,exp)	Number	The total of the numeric expression exp across a series of records in target file f (defined in the GENERAL dictionary or a supported external dictionary), using key expression k. The key expression defines the first portion of the key to the target file. If the target file is specified with a sort (file-name@sort-name) then the sort specified is used when accessing the range of records indicated by the key expression.
TRIM(x)	Text	The text value x, after leading and trailing spaces are removed.
UPPER(x)	Text	The text string x, after converting all letters to upper case.
VAR(x,y,...,z)	Number	The statistical variance of the stream of numbers x..z.
XREF("f",k,exp)	Text	The value of the expression exp in the record in target file f (defined in the GENERAL dictionary or a supported external dictionary), found by using key expression k. The key expression defines the full primary key to the target file.
YY(d)	Text	The 2-digit representation of the year portion of date d.
YYYY(d)	Text	The 4-digit representation of the year portion of date d.

Sort Definitions



A sort definition for a GENERAL dictionary can serve two purposes.

First, when the LIST command is used to generate a report, the ALTSORT keyword can be used to reference one of the sort definitions for a file. When an ALTSORT keyword is used, GENERAL processes the file in a different order than by the file's primary key, which is the default. Instead, GENERAL will use either an alternate key or an alternate file. In addition, when an ALTSORT is used, the BEGIN and END keywords, which are used to process ranges of records, will apply to the alternate sort.

For example, if a LIST command for a customer file uses an ALTSORT STATE, and the STATE sort definition has been defined in this table to access the file by a secondary key sorted in state code order, then a BEGIN "CA" END "CA" would only process records for California.

The second use for a sort definition occurs when a Link definition is set up on a file to which the current file is related, or the current file is the target of a cross reference function. Note that this refers to how other files link to this file, not how this file links to others.

The link definition can be instructed to use the sort order specified here when accessing records in this file. Using the above example with a sort definition for state code, a state code report could reference a list of customers for each state through a cross-reference function like this:

MREF("CUSTOMERS@STATE", ID, NAME)

Name

The name of a sort definition can be from 1 to 15 letters, digits, and underscores. It can't contain spaces, and it should be unique among all the sort definition names for the file.

While in the name field, the following function keys are active:

F3-insert Inserts a blank row for a new definition.

F4-delete Deletes the current row.

F9/F10-done Exits sort definition maintenance and issues the verification prompt.

Description

The description of this sort definition is used when lists of alternate sorts are displayed in GENERAL (in assist mode, for example).

ALTFILE or ALTKEY expression

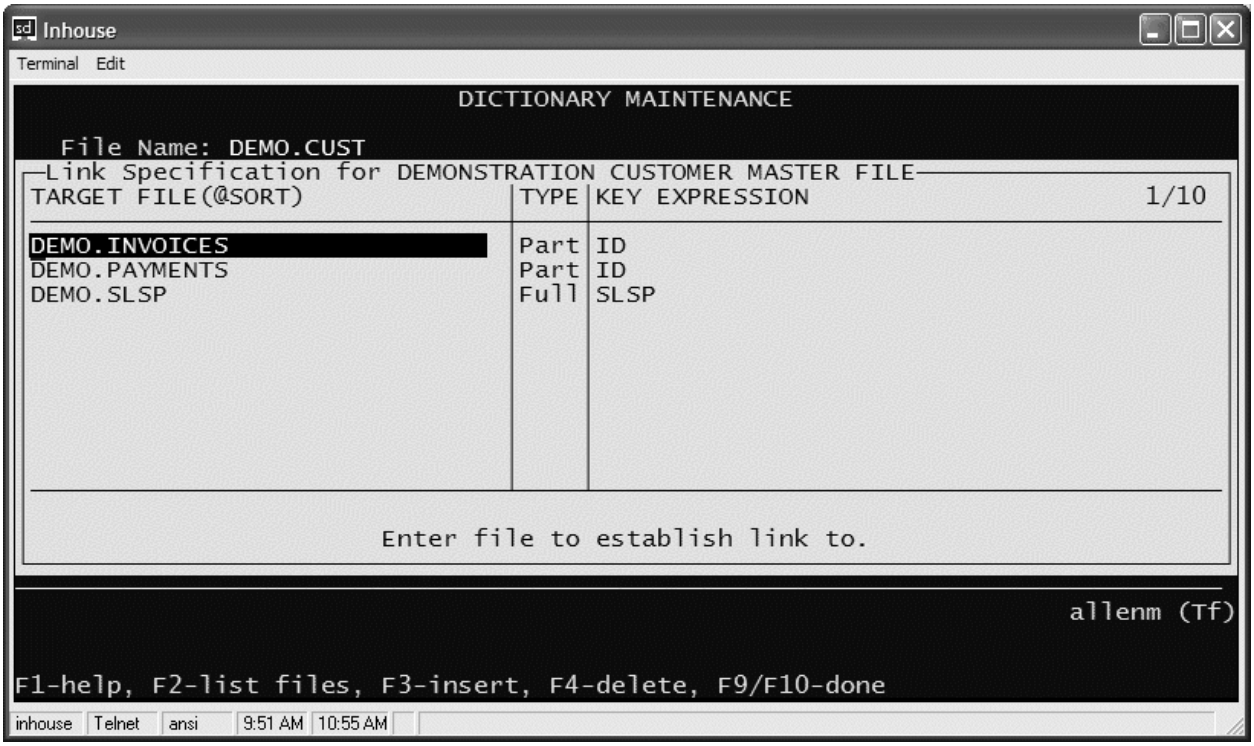
This is where the processing for this sort is defined. GENERAL supports two sort processing methods: ALTFILE and ALTKEY. Note that each of these methods is supported independently as a keyword.

ALTFILE is used to indicate the use of a sort file to drive the processing of the file. Following the ALTFILE keyword is a file name, which should be another file in GENERAL's dictionary or in a supported external dictionary. Following the file name is a specification of how to derive the primary file's key from the key of the alternate file's key. More detail about ALTFILE syntax can be found in the Keyword Reference chapter.

ALTKEY is used to indicate the use of a secondary key in a multi-keyed file. Following the ALTKEY keyword is a number indicating which key number to use.

Both ALTFILE and ALTKEY have abbreviations: AF and AK, respectively.

Link Definitions



Link definitions are used by GENERAL to pre-define relationships between files, so that reports can reference data from related files with the link operator.

A link definition is basically a description of a key, derived from the current file, and used to read one or more records from a related file. Link definitions can use the related file's Sort definitions to access records via a secondary key or a sort file.

File relationships are supported in one-to-one and one-to-many form. The one-to-one form uses a full key to access data in another file. Note that a many-to-one relationship can also be termed a full key relationship, because for each record in the current file, there is just one record in the target file that can be found.

The full key to a customer master file, for example, might be a 2-digit company code plus a 6-digit customer number. If the current file contains the company code and the customer number, then a full key or one-to-one relationship exists between the current file and the customer master file.

The full key to an open order file might be a 2-digit company number plus a 6-digit customer number, plus a 6-digit order number. If the current file contains the company number and the customer number, but not the order number, then a partial key, or one-to-many, relationship exists between the files.

When partial key relationships exist, GENERAL can display each record matching the partial key, or can summarize the records in several ways: total, average, and count.

Target File@Sort

The target file is the name of the file that the current file is related to. The file must be defined in the GENERAL dictionary or in a supported external dictionary. If there is a Sort definition that defines the order in which to access the records in the target file for this relationship, then it can be referenced with the *@sort-name* at the end of the file name.

The target file **DEMO.INVOICES@SLSP** would reference data in the file DEMO.INVOICES, sorted by the Sort specification SLSP.

Function key options at this entry are:

F2-list files List the files, and associated sort definitions, in a selection window.

F3-insert Inserts a new row in the table.

F4-delete Deletes the row.

F9/F10-exit Exits maintenance and issues a verification prompt.

Type

This defines the type of link.

Use **1** to indicate a full key link, where there is a one-to-one or many-to-one relationship between the current file and the target file. The key defined in the Key Expression column will be the full key needed to access a record in the target file.

XREF() functions use a full key link, and a link operator reference, such as CUSTOMER:NAME, will generate an implicit XREF() function if the link defined to the target file CUSTOMER is a full key link.

Use **2** to indicate a partial key link. Partial key links are used by MREF(), TREF(), AREF(), and CREF() functions. The key defined in the Key Expression column will be a partial key that defines the range of records to process when this link is referenced.

Key Expression

This defines the key to be used when accessing records in the target file. Key expressions can range from as simple as the field SLSP to as complex as CUSTOMER+INVOICE, or @PF1(1,2)+"A"+PC_CODE.

EXTERNAL DICTIONARY SUPPORT

In addition to its internal dictionary, General can support several external dictionary formats as well. In releases 4 and 5, General supported the Basis Data Dictionary and Filix data file dictionaries. This support still exists and is discussed in the Appendix, in the Application Integration section. In version 6, additional support has been added to more easily support the Basis Data Dictionary, and also to support two ProvideX data dictionary formats: ini files and providex.ddf files.

To establish support for either of these dictionaries, you must edit the gen6parm.fil text file, and enable the following lines:

extdict=*path*[;*path*]	Sets the external dictionary to a file *path*, which must be one of the following: A Basis TPM file, which contains at a DICTIONARY= line and any parameter lines needed by the dictionary itself. A common parameter name is DATA, though others may be used. A ProvideX .ddf file, used to define normalized data files for the Nomads development environment. A ProvideX .ini file, whose format is described in the Providex ODBC documentation. Note that the "flattening" file specification, using the RECTYPE parameter, is not supported. You may specify both types of ProvideX dictionaries by delimiting multiple paths with semi-colons. General will merge the specifications from both dictionaries.
extdate=*suffix*	The *suffix* is a text value, such as _date, which indicates an assumed date field. When General loads the fields in the external dictionary, any field that ends with this value is presumed to be a date field, whose value will be calculated based on the extdtf setting below.
extdtf=*dateformat*	When a date field assumption is made, use *dateformat* to extract the date. The format can be any of the following: jul for a native julian number aon for an AddonSoftware date ssi for an Aperum (formerly SSI) FACTS software date *map* of mm, dd, yy, and yyyy that defines the date structure. For example: yyyymmdd for dates stored in 8 bytes, with a 4-

	digit year, a 2-digit month, and a 2-digit day.
ext1000=y or n	If set to y, then number fields are formatted with thousands separators, such as 9,999.00.
exthide=*regexpr*	Sets a regular expression pattern that is used to automatically hide fields. For example, "^dummy|^reserved" would assume field names starting with dummy or starting with reserved should be hidden.
extcase=u, l, p, or i	If set to one of these letters, will force field headings to the specified case: u=UPPER CASE l=lower case (this is the letter ell) p=Proper Case i=Initial caps case
extsort=y or n	If set to y, will generate alternate sort specifications based on the secondary keys in the file.
extlink=y or n	If set to y, will generate link specifications for a file based on matching field names in the keys of potential target files. This process can be time consuming, and is also dependent on field name matching. If key field names are not maintained throughout the dictionary, then viable links will be missed. Likewise, if identical field names are given to different data elements, bad links will be generated. Performance depends on the size of the dictionary and the speed of the system. General will cache the links found for any given file during a session, so the scanning overhead is only incurred once per file per session.
extminl=*minlinkcount*	Sets the minimum number of link segments that must be matched before a link is generated when extlink is set to y. The purpose of this is to prevent runaway link building. For example, if all files contain a company code as the first segment of a key, then all files could be linked to all other files. By setting this value to 2, then only files with two or more matching segments would be linked.

Plus Dictionaries

Once an external dictionary is specified, all General reports and file lists can be based on the files and fields defined in that dictionary. There are additional capabilities offered by General's internal dictionary that can be merged with the definitions found externally.

For example, you may choose to not turn on external linking or sorting. If so, you can define sort and link tables within General's dictionary and have them merged so that reports can use the external dictionary for basic data field specification and the internal dictionary for sorting and linking.

In addition, General's data dictionary supports calculation fields. You can add fields that perform simple or complex calculations and have those fields merged with the physical data fields defined in the external dictionary.

The way to create an internal dictionary that will be merged with an external one is to use the "plus" dictionary nomenclature when defining a file in General's dictionary. If a file is called "AR_CUST_MAST" in the external dictionary, you would define the internal dictionary with the name "+AR_CUST_MAST". All sorts, links, and fields that you define in this plus dictionary will merge and replace values in the external dictionary.

KEYWORD REFERENCE

Keywords are used within a List command to control report features, such as placing fields, adding break points and subtotals, delimiting sort or select phrases, and specifying an output device or file. There are dozens of keywords available. May require optional parameters. For example, the ON keyword requires a filename, printer, or special device name to follow it.

When using the Prompt mode to create a report, you don't normally need to worry about keywords, and the List command generated by the report design will contain the keywords necessary.

For users who need to edit List commands directly, the following pages provide a reference for each keyword available.

:(Link Operator)

PURPOSE To generate a cross reference to a related file at run-time, based on the Link definition established in the LISTed file's dictionary.

FORMAT LIST ***... file-name:field-name ...***
LIST ... operator:file-name:field-name ...

SYNONYMS None

USAGE

The link operator is implemented by the LIST command in order to facilitate an easy run-time method of reporting from multiple files. The link operator is not the only means of relating files, but it is the most concise and easiest to use when existing field definitions don't include cross-reference calculations. The combination of the file, link operator, and field is called a *link field definition*.

When GENERAL parses a link field definition, it looks for a link specification between the LIST file and the file-name specified in the link. A *link specification* is a definition of how two files are related, and is defined either in the "from" file's dictionary, or in the LIST command with the LINK keyword. If there is no link specification, the cross-reference cannot be created. If a link specification is defined with the LINK keyword, it must be defined before any link fields are specified that use that definition.

Link specifications describe either a "full-key" relationship, where there is just one record to access in the target file, or a "partial-key" relationship, where there may be multiple records in the target file that relate to the LIST file. If a partial key relationship exists, then GENERAL recognizes three *prefix operators*: @SUM, @AVG, and @CNT (@TOT is a synonym for @SUM). These indicate that the target field is to be totaled, averaged, or counted, respectively. If the link is a partial-key link, and no operator is specified, then all the related records are displayed sequentially on the report.

Internally, GENERAL converts link field definitions to run-time calculations using the cross-reference functions: XREF(), MREF(), TREF(), AREF(), and CREF(). It is possible to enter such calculations manually, with the @CALC keyword, which saves parsing time, but makes for a more complex LIST command. There is no report performance penalty for using either method.

Links can be nested; that is, they can be chained through several files. The only limitation to chaining is that only one "partial key" relationship can exist in any chain. The link field definition **HEADER:CUSTOMER:NAME** will look for link specifications from the current file to the HEADER file, and from there to the CUSTOMER file, to finally retrieve the NAME.

EXAMPLES

This example shows a link field definition to get the salesperson name from the salesperson master file. The link between DEMO.CUST and DEMO.SLSP is defined in DEMO.CUST's dictionary:

LIST DEMO.CUST ID NAME **DEMO.SLSP:NAME**

This example shows a summarized link that will add up the amounts from the invoice file:

LIST DEMO.CUST ID NAME TOTAL **@SUM:DEMO.INVOICES:AMOUNT**

@BREAK*n*
@DATE
@DTM
@LINE
@PAGE
@PAGEDSC
@TIME
@TTY
@USER
@VDT

PURPOSE To print specific values on a report or in a header or footer.

FORMAT LIST **... @BREAK*n* | @DATE | @DTM | @LINE | @PAGE | @PAGEDSC | @TIME | @TTY | @USER | @VDT...**

SYNONYMS **@VDT**: @FID

USAGE

These keywords can be used as regular data fields in the field phrase or header/footer sub-phrases. Some, such as @PAGE or @DTM, are obviously intended for page headers (defined with the HEADER0 keyword).

These keywords return the following values:

@BREAK*n* The break point value for level *n*. Break points are numbered as they occur in the LIST command. Up to 9 levels are supported.

@DATE The system date, formatted as defined by the configuration. The default format is MM/DD/YYYY.

@DTM The system date and time. The default format is MM/DD/YYYY HH:MM PM.

@LINE In headers and footers, prints a blank line. In report lines, prints the line counter (useful for generating "ranking" reports).

@PAGE The current page number, occupying 5 positions, right justified and space filled. The @PAGE variable is now valid in calculation expressions. An example of its use would be to reset the page number after a break point, in a custom footer calculation. Here is a sample report:

LIST DEMO.CUST BREAK-PG SLSP NAME TAB 50.1 TOTAL YTD.SLS SORT SLSP FOOTER1 TAB 50.1 "----------" TAB 50.2 @CALC("N,L12,2", "", GSUM(YTD.SLS,1); **@PAGE=0**)

@PAGEDSC The word "PAGE", followed by the current page number.

@TIME The system time. The default format is HH:MM PM.

@TTY The system tty device for the terminal.

@USER The user's login name (from the operating system, not GENERAL).

@VDT The Business Basic name (or alias) for this terminal. This is the same as the FID(0) function in Business Basic.

EXAMPLES

This example shows the use of several keywords in a header sub-phrase:

LIST DEMO.CUST ID TAB 10 NAME TAB 55 YTD.SLS HEADER0 **@DTM** CENTER "CUSTOMER LIST" TAB 75 **@PAGE** LF **@USER** TAB 1.3 FILL "=" OFF

This example shows the @LINE keyword used to produce a ranking report:

LIST DEMO.CUST NAME **@LINE** DEMO.SLSP:NAME YTD.SLS SORT DSND YTD.SLS STOP 20

@CALC

PURPOSE To define a run-time calculation in a LIST command.

FORMAT LIST **... @CALC*name*(*"type-codes"*,*"heading"*,*expression*) ...**

SYNONYMS None

USAGE

Run-time calculations are functionally just like a data field from a file's dictionary. They may be placed anywhere in a LIST command that a regular field can be placed.

The calculation name can contain up to 15 letters, digits, and underscores (individual underscores only).

@CALC_NAME is a valid name.

@CALC1__2_IN_A_ROW isn't valid because two underscores can't be next to each other.

The type-codes parameter is identical to the type codes used when defining a field in the dictionary. There must be a basic data type and length code, and may be additional formatting codes:

T,L30	30 character text field.
N,L12,2	12 character, precision 2, number field.
D,L10	10 character date field.

See the Fields section of the Dictionary chapter for more information about type codes. Unlike fields in the dictionary, @CALC definitions can be zero-length fields, which disables the printing of any data. Typically, zero-length calculations are used as numeric values in later @CALC definitions within the LIST command.

The heading parameter defines the column heading for the calculation. It can be divided into stacked segments with vertical bars: **CUSTOMER|NAME** would be a two-line column heading. If any segment is longer than the defined length from the type-codes, then GENERAL expands the output to the width of the heading.

The expression parameter is a calculation expression that can reference physical data, other fields, constant values, other calculations (previously occurring in the LIST command), and functions. A detailed explanation of expressions can be found in the Fields section of the Dictionary chapter.

EXAMPLES

This example shows a text calculation:

LIST DEMO.CUST ID TAB 10 NAME TAB 60 YTD.SLS LF TAB 10
CALC_FULL_ADDR("T,L50", "CUSTOMER'S ADDRESS",
TRIM(ADDR1)+"/"+TRIM(ADDR2)+" "+CITY_ST_ZIP)

This example shows two numeric calculations, with the second referencing the first:

LIST DEMO.CUST NAME YTD.SLS YTD.COST
@CALCPROFIT("N,L12,2","PROFIT",YTD.SLS-YTD.COST)
@CALCPCT("N,L6,1","GP %",100*DIVIDE(@CALCPROFIT,YTD.SLS))

ACROSS

PURPOSE To print multiple records side by side, across the report.

FORMAT LIST **... ACROSS** ***number...***

SYNONYMS None

USAGE

Often, when information is desired in columns on a report, there is enough room to place several records across the page. One obvious instance of this is mailing labels, where sheets can have 3 or 4 labels across. The ACROSS keyword instructs GENERAL to place records on WIDTH boundaries across the page. The total page width becomes *width* * *number*, where *width* is specified with the WIDTH keyword. Also, each record will occupy the same number of lines, regardless of NO-BLANK settings.

When printing records across a page, and a break point is encountered, GENERAL will print however many records have been found, and perform the break. The footer will occupy the entire report width (not just the width of each record). Any default footers to be printed will be printed in vertical fashion, as if the VERT-TOTAL keyword is used.

Like footers, the report header will also occupy the full-page width. Column headings, which are built into the report header, however, are duplicated for each ACROSS column.

EXAMPLES

This example shows using the ACROSS keyword to produce 3-up mailing labels:

```
LIST DEMO.CUST TAB 2.2 NAME TAB 2.3 ADDR1 TAB 2.4 ADDR2 TAB 2.5
CITY_ST_ZIP TAB 2.6 "" ACROSS 3 NO-PAGE ON PRINTER
```

Note the use of NO-PAGE to suppress pagination, and the TAB 2.6 "" to force 6 lines on the labels. Another method to force 6 lines per label would be to use the keyword HEIGHT 6.

ALTFILE

PURPOSE To allow the use of an already existing sort file instead of using a sort-phrase, which requires GENERAL to sort the file being LISTed before the report can be produced.

FORMAT LIST **... ALTFILE** ***file-name key-position...***

SYNONYMS AF, PRESORT

USAGE

Business Basic applications often use sort files to cross-reference to existing master files. Examples might be sort files for customers by name or employees by social security number and so on. GENERAL allows these files to be used, thereby saving sorting time during the compilation of a report.

The file-name must be predefined in the dictionary, and the disk file associated with it must be accessible to GENERAL. In addition, the user access level must be at least as high as the file's access level. The file can be any keyed type supported by GENERAL. However, only the key to the file is used by this keyword.

The key position parameter can be entered in one of two formats. The parameter must be quoted in each case (unlike prior versions), and if it contains quotes itself, then single quotes may be used.

Format 1 is a numeric value, which defines, where in the key of the sort file the key to the master file is placed. It may optionally contain two numeric values separated by a comma ("12,6" for example). The second value represents the length of the key.

Format 2 is as an executable string using the variable K$ to represent the data, which allows multiple portions of the key to represent the primary file key.

When using ALTFILE, the BEGIN and END keywords define the ALTFILE key range rather than the master file key range. If, for example, a sort file by name is used, the BEGIN and END keywords would define a name range.

If a sort phrase is present in addition to the ALTFILE keyword, the phrase, rather than the ALTFILE keyword define the sort order. However, record ranges defined by the BEGIN and END keywords apply to the ALTFILE.

See also the ALTSORT keyword.

EXAMPLES

The following example shows the use of an alternate sort file:

LIST DEMO.CUST ID NAME ALTFILE DEMO.CUST_NAME "21,5"

The next example shows an alternate form of key position definition:

LIST DEMO.CUST ID NAME ALTFILE DEMO.CUST_NAME "K$(16,5)"

The last example shows how to use single quotes in cases where the key parameter contains quotes:

LIST *file field-1 field-2* ALTFILE CUST-BY-NAME 'K$(1,2)+"A"+K$(3,5)'

ALTKEY

PURPOSE To allow the use of an alternate key-number instead of using a sort-phrase, which requires GENERAL to sort the file being LISTed before the report can be produced.

FORMAT LIST **... ALTKEY *key-number...***
LIST ... ALTKEY?

SYNONYMS AK

USAGE

Some implementations of Business Basic support the use of multi-keyed files. Such files allow one or more alternate keys to be defined in addition to the primary key to the file. GENERAL supports access to the alternate keys with the keyword ALTKEY.

The key-number is a number that identifies the alternate key-number desired. This is normally a number between 1 and 16. Key-number 0 is the primary key to the file. The key-number selected must be defined as an alternate key-number to the file. GENERAL does not verify the validity of the number, and entry of an invalid key-number may cause an error at run-time.

If a question mark is entered as the key-number, then GENERAL presents a selection list of valid alternate keys to the file being LISTed.

When using ALTKEY, the BEGIN and END keywords define the ALTKEY key range rather than the primary key range.

If a sort phrase is present, in addition to the ALTKEY keyword, the phrase, rather than the ALTKEY keyword define the sort order. However, record ranges defined by the BEGIN and END keywords apply to the ALTKEY.

See also the ALTSORT keyword.

EXAMPLES

The following example shows the use of an alternate key:

LIST DEMO.CUST ID NAME **ALTKEY 1**

ALTSORT

PURPOSE To facilitate the use of both ALTFILE and ALTKEY parameters by name, rather than requiring knowledge of the required parameters of those keywords.

FORMAT LIST **... ALTSORT** ***altsort-name...***
LIST ... ALTSORT?

SYNONYMS AS

USAGE

Most Business Basic applications support alternate sorting options to master files through either alternate sort files or, when the implementation supports multi-keyed files, through alternate keys. GENERAL supports the use of either method through the ALTFILE and ALTKEY keywords. However, those keywords require some technical knowledge, and are best suited to programmers.

Through ALTSORT definitions, maintaining a table of named equivalents can mask the technical nature of the ALTFILE and ALTKEY keywords. This table is maintained in the dictionary Sorts option. Each ALTFILE or ALTKEY definition desired for a particular file can be given a name and a description. Then the ALTSORT keyword can reference the name rather than the ALTFILE or ALTKEY definition.

Also, if the name of the ALTSORT definition is not known, a question mark may be entered, and GENERAL will provide a menu listing of all ALTSORTs defined for the file being LISTed.

EXAMPLES

The following example shows the use of an alternate sort definition:

LIST DEMO.CUST ID NAME **ALTSORT NAME**

This example will produce a menu of all ALTSORT definitions for the CUSTOMER file:

LIST DEMO.CUST ID NAME **ALTSORT ?**

ASCII
EXPORT

PURPOSE To facilitate exporting of data to another application, such as a spreadsheet or word processor. Several standard ASCII text-based formats are supported.

FORMAT LIST **... ASCII...**
LIST ... EXPORT ? ...
LIST **... EXPORT DELIMITED** (or DELIM)
LIST **…EXPORT DELIMITED-H**
LIST ... EXPORT FIXED
LIST ... EXPORT DIF
LIST **... EXPORT WORDPERFECT** (or WP)
LIST ... EXPORT EXCEL

SYNONYMS None

USAGE

The ASCII and EXPORT keywords are used to cause GENERAL to format the output in ASCII text-file format. If the ASCII keyword is used, or the EXPORT ? format, then GENERAL presents a list of valid export formats for selection. Each format is suitable for certain export package targets.

In all cases, GENERAL automatically turns off pagination (like the NO-PAGE keyword does), so the output will not contain any form-feed characters, or page headings. Also, GENERAL automatically adjusts the report width to accommodate all fields.

If an EXPORT format is specified in a NO-DETAIL report, GENERAL will format the sub-total and report-total footers into the export format selected for delimited, fixed position, and DIF formats.

Delimited: All alpha type data is enclosed in quotes, all numeric data is scaled and decimal-formatted without commas, and commas are used to separate each field. Most spreadsheets and databases can import a delimited file. GENERAL can also report from a delimited file, if a dictionary is defined for it.

The delimited-h option is identical to delimited, but adds a single row of column headings to the export file. Many applications, such as Word and Excel, can use this first row to name fields during a merge or import operation.

The **DELIMITER**, **NO-QUOTE**, **PREFIX**, and **SUFFIX** keywords may be used to further customize how GENERAL creates the delimited file.

Fixed
Each field occupies a set position and length on a line. Numbers are formatted without punctuation.

DIF
This format (**D**ata **I**nterchange **F**ormat) is an old export standard. Many spreadsheets and databases can import a DIF file.

WordPerfect:
Use this format to create a file suitable for a WordPerfect merge operation. The file will contain the ^R and ^E terminators that WordPerfect recognizes as field and record terminators.

Excel:
Using either sdOffice (a separate product) or DDE, GENERAL exports data directly to Excel. The DDE version only works on Windows platforms (Visual PRO/5 and ProvideX or WindX on Windows). If you also specify ON *filename*, that file will be created or updated as an Excel workbook (*filename*.xls).

If you are using sdOffice, it may be necessary to specify the sdOffice server machine with an environment variable "SDHOST" or a STBL or GBL table value "$sdhost". The sdOffice mode of the Excel export supports the Title keyword and retains column formatting.

Normally, use of this keyword would coincide with the use of the ON keyword in order to send the output to a file. That file could then be imported by applications, which support file importing.

EXAMPLES

The following example will prompt the user for an export format, then create an export file FILE.PRN.

LIST DEMO.CUST BREAK SLSP DEMO.SLSP:NAME NAME TOTAL YTD.SLS SORT SLSP DESCENDING YTD.SLS **ASCII** ON "FILE.PRN" SELECT YTD.SLS > 999.99

This example will product a WordPerfect merge format file:

LIST DEMO.CUST NAME ADDR1 ADDR2 CITY_ST_ZIP **EXPORT WORDPERFECT** ON FILE

AVERAGE

PURPOSE To cause a numeric field to be averaged, with the average value output at the end of the report, and at break points if specified by the BREAK keyword.

FORMAT LIST **... AVERAGE** ***field-name...***

SYNONYMS AVG

USAGE

The AVERAGE keyword is used to generate the average numeric value of a specified report column. The average generated will be printed at the end of the report and also at each subtotal break point if the BREAK keyword is used.

The field-name specified must be defined as a numeric type field.

EXAMPLES

The following example will produce an average of the extended value of each line on the report. It will print at the end of the report under the YTD.SLS column:

LIST DEMO.CUST ID NAME **AVERAGE YTD.SLS**

BEGIN

PURPOSE To specify the first key value when LISTing only a range of keys.

FORMAT LIST **... BEGIN** ***start-key...***

SYNONYMS FIRST, START

USAGE

The BEGIN keyword is used to specify part of a key range for a report. The key range is the range of record keys in the primary file or ALTFILE or ALTKEY to process. When used in conjunction with the END keyword, a report can be specified which will only process a range of key values out of the primary file. Without this keyword, GENERAL processes keys starting with the first key in the file.

When a range is selected, records outside that range are not processed. This is different than when a select-phrase is used or when skip keys are defined for the file, since in those cases records are processed but not output.

The start-key may be any set of characters. If it includes spaces it must be enclosed in quotes, otherwise quotes are optional. The start-key does not have to exist in the file. If the primary file being LISTed is an indexed file rather than a key-oriented file, then you must specify a numeric constant rather than a start-key. The number represents the first index to process.

When used in conjunction with the ALTFILE keyword or the ALTKEY keyword, the key range is that of the alternate file or key, not the LISTed file or primary key.

You can precede the BEGIN parameter with one of the following keywords to adjust how the parameter is interpreted:

CND converts a literal date to a Julian integer. **CNDI** converts a literal date to a Julian integer, compressed into 4-byte binary form. **CNI** converts a literal number into a 4-byte binary form. These conversions coincide with BBx template sizes and uses.

For example, **BEGIN CND "3/15/95"** would convert 3/15/95 to a Julian number, and use the result (in BBx, 2,449,792) as the BEGIN value. **BEGIN CNDI "3/15/95** integer with the BIN function (BIN(2449792,4)). **BEGIN CNI 99995** would convert 99,995 to a 4-byte integer with the BIN function.

In addition, if the BEGIN parameter is enclosed in single-quotes (‘), the parameter is interpreted as an expression.

EXAMPLES

The following example will process a key range beginning with "00001" and ending with "00099":

LIST DEMO.CUST ID NAME PHONE **BEGIN "00001"** END "00099"

This example shows how BEGIN affects a report when an ALTSORT is used:

LIST DEMO.CUST ID NAME PHONE ALTSORT NAME **BEGIN "GREEN"** END "GREEN"

This example show an expression:

LIST DEMO.CUST ID NAME BEGIN ‘str([[Enter customer number]]:”00000”)’

The expression can use functions defined in the file “userdefn.txt”, if mult-line function capabilities are required.

BREAK, BREAK-PG
SBREAK, SBREAK-PG

PURPOSE To create a report break based on a certain field, and to generate subtotals at that break point when the AVERAGE, COUNT, MAXIMUM, MINIMUM, PCT-TOTAL, or TOTAL keywords are used.

FORMAT LIST **... BREAK *field-name...***
LIST ... BREAK-PG *field-name...*

SYNONYMS **BREAK**: BRK
BREAK-PG: BRK-PG, BRKP
SBREAK: SBRK
SBREAK-PG: SBRK-PG, SBRKP

USAGE

The BREAK keywords are used to denote a break point for a report, where lines are skipped and subtotals are printed. The field-name specified for the break may be a text type field, and usually should be associated with the sorted order of the report. It may also be a date or numeric field, generally prefixed by a conversion keyword such as MMYY or EVEN 1000. Breaks on text fields can be defined to occur when only the first *n* characters change with the CUT keyword.

The field-name specified is considered part of the field-phrase, and will appear on the report, unless the silent (SBREAK) format is used. Whenever the value of the field changes, the report output will break for subtotals, and then will continue. If the BREAK-PG or SBREAK-PG keyword is used, then a page is skipped before the next report line after the break point is printed.

Multiple breaks may be specified for the report. Subtotals will be nested, with the nesting levels defined by the left to right position of the break fields in the LIST command. GENERAL supports up to 9 break levels.

The value of the break point can be printed using the @BREAK*n* keyword. This is typically used in custom group level headers or footers, and in fact is what GENERAL inserts in default footers.

When GENERAL creates default footers, it will first attempt to place all aggregate column calculations directly below their respective column headings. The break point value and descriptions will be to the left of the column calculations. If there are any position conflicts, GENERAL converts the footer to a vertical format, as if the VERT-TOTAL keyword had been used.

To avoid the vertical format, it is sometimes necessary to specifically position aggregate fields further to the right in the report, using the TAB keyword.

Note that if the VERT-TOTAL keyword is used, then all footers are formatted vertically regardless of where column positions are placed.

EXAMPLES

This example will produce a report with subtotals by salesperson:

LIST DEMO.CUST **BREAK SLSP** ID NAME TOTAL YTD.SLS SORT SLSP

This example will produce the same report, but without printing the salesperson on each report line:

LIST DEMO.CUST **SBREAK SLSP** ID NAME TOTAL YTD.SLS SORT SLSP

The last example will produce a page break by salesperson, and a line break in even 5000 increments on the sales value:

LIST DEMO.CUST **SBREAK-PG SLSP** DEMO.SLSP:NAME ID NAME **BREAK EVEN 5000** TOTAL YTD.SLS BY SLSP DSND YTD.SLS

CENTER

PURPOSE To place the next field on the report at the center of the report line. This overrides any tab positioning or automatic positioning of the data.

FORMAT LIST **... CENTER** ***field-name...***

SYNONYMS CTR

USAGE

The CENTER keyword can be placed in front of any data field in a field phrase or header/footer sub-phrase. The position of the field is calculated based on the width of the report and the number of columns the field occupies.

Note that the centering is not calculated based on the actual text length, but instead based on column length, so centering may be only approximate for any given field.

EXAMPLES

The following example will center the sales and cost fields on two rows:

LIST DEMO.CUST ID NAME CENTER YTD.SLS LF CENTER YTD.COST

CONVERT-DATE

PURPOSE To convert a text version of date to an internal date, suitable for use as a GENERAL date type field.

FORMAT LIST **... CONVERT-DATE** ***field-name*** **|** ***"text date"...***

SYNONYMS CND

USAGE

Many older Business Basic applications do not utilize the internal Julian date features of recent versions of the language. Often, dates are stored as text in MMDDYY or YYMMDD format. The CONVERT-DATE keyword provides an ability to convert those fields to date values on the command line.

The CONVERT-DATE keyword converts the text element on the report to a date. It relies on the system date entry format, from the system configuration, when interpreting the field or text. The date entry format defaults to MMDDYYYY (the YYYY portion can be entered as 2 or 4 digits), so data or text such as "031501" would be converted to an internal date for March 15, 2001. If desired, dates can also be entered with delimiters, with the month, day, and year elements placed in the same order as the date entry format. "3/15/01" would be converted the same way as "031501".

If a field-name is used as a parameter, it must be a text field whose display value is the same as the date entry convention.

EXAMPLES

This example converts the literal date entry into an internal date, for proper comparison to the DATE field. Note that if the DATE field were on the left side of the expression, the CONVERT-DATE keyword would be unnecessary, as GENERAL would know to convert the literal text automatically.

LIST DEMO.INVOICES CUSTOMER INVOICE DATE AMOUNT SELECT **CONVERT-DATE** "8/1/01" < DATE

COPIES

PURPOSE To cause a specified number of copies of a report to be output.

FORMAT LIST **... COPIES n...**

SYNONYMS None

USAGE

In GENERAL 4.1 the keyword, COPIES, has been added, which cause GENERAL to send extra copies of a report to the printer or a file. If COPIES is set to a value less than 2, then one report is printed. If COPIES is set to 2 or more, then GENERAL will print copies 2 through *n*, with a page eject between each.

Copies are executed by storing an image of the report to a file, then reading from that file to produce the second and additional copies. This is much faster than re-executing the report multiple times.

GENERAL ignores this keyword if the report is sent to the VDT or PREVIEW device.

EXAMPLES

The following example will print a customer list and provide a count of the number of customers printed:

LIST DEMO.CUST NAME SLSP PHONE **COPIES 3**

COUNT

PURPOSE To cause a count of the lines printed to be output at the end of the report, and at break points if specified by the BREAK keyword.

FORMAT LIST **... COUNT** ***field-name...***

SYNONYMS CNT

USAGE

The COUNT keyword is used to count of the number of report lines printed. The tally will appear at the end of the report and at break points.

The field-name specified is considered part of the field-phrase, and will appear on the report. There is no restriction on field type. However, there must be enough space allocated by the field length to accommodate the count value. Count values are always formatted with thousands separators.

EXAMPLES

The following example will print a customer list and provide a count of the number of customers printed:

LIST DEMO.CUST **COUNT ID** NAME SLSP PHONE

CUT

PURPOSE To trim a text value to the first *n* characters for purposes of generating break points on just a portion of the text value.

FORMAT LIST **... BREAK CUT *length field-name...***

SYNONYMS 1ST

USAGE

The CUT keyword is used as part of a BREAK specification, when a text field break should only reference a beginning portion of the text field.

For example, if an inventory code contains a three-character product group at the beginning of the code, a break point could be generated on just the product group portion with BREAK CUT 3 INVENTORY_CODE.

EXAMPLES

In the example below, a customer list is sorted by name, with a break point for each letter of the alphabet:

LIST DEMO.CUST ID NAME CITY_ST_ZIP PHONE SORT NAME SBREAK **CUT 1** NAME

DELIMITER

PURPOSE To specify a delimiter other than a comma in a delimited export.

FORMAT LIST ... EXPORT DELIMITED **... DELIMITER *"char"*...**

SYNONYMS DELIM

USAGE

Normally, GENERAL will create delimited export files with a comma separating each field. With the DELIMITER keyword, the delimiter can be changed to any other character.

If a non-printable character is required, such as a <tab> character, then GENERAL's ASCII notation can be used inside the quotes, like this: "~009" for ASCII value 9.

EXAMPLES

This example shows how to perform an export with a colon as a delimiter:

LIST DEMO.CUST ID NAME ADDR1 ADDR2 CITY STATE ZIP EXPORT DELIMITED **DELIMITER ":"** ON "/usr/exports/addresses"

DESCENDING

PURPOSE To cause sorting on a field to be in descending sequence.

FORMAT LIST ... SORT **... DESCENDING** ***field-name...***

SYNONYMS DSND

USAGE

This keyword is used to modify the default sorting order for fields specified in a sort-phrase. Normally, fields are sorted in ascending sequence, either by their ASCII values, or their numeric value, depending on the field type. When this keyword precedes a field in a sort-phrase, that field is sorted in descending sequence.

EXAMPLES

The following example will sort first by salesperson, then from highest to lowest monthly sales within salesperson:

LIST DEMO.CUST ID NAME PHONE OPEN.BAL SORT SLSP **DESCENDING MTD.SLS**
SELECT MTD.SLS >= 100 ON LP

DOUBLE-SPC

PURPOSE To force double spacing of a report.

FORMAT LIST **... DOUBLE-SPC...**

SYNONYMS DBLSPC, DSP

USAGE

The DOUBLE-SPC keyword can be used to cause GENERAL to place a blank line between each record on a report, even if a NO-BLANK keyword has been used as well.

A trailing NEWLINE keyword may also be used to double-space a report.

EXAMPLES

This example shows a double spaced report:

```
LIST DEMO.CUST ID TAB 10.1 NAME TAB 10.2 ADDR1 TAB 10.3 ADDR2 TAB 10.4 CITY
STATE ZIP NO-BLANK DOUBLE-SPC
```

END

PURPOSE To specify the last key value when LISTing only a range of keys.

FORMAT LIST **... END** ***end-key...***

SYNONYMS LAST, FINISH

USAGE

This keyword is used to specify part of a key range for the report. The key range is the range of record keys in the primary file or alternate sort to process. If an inventory file includes product group as the first three digits of the item keys, then using END in conjunction with BEGIN would allow a range of product groups to be specified for the report.

When a range is selected, records outside that range are not processed. This is different than when a select-phrase is used or when skip keys are defined for the file, since in those cases records are processed but not output.

The end-key may be any set of characters. If it includes spaces, it must be enclosed in quotes. If it does not, the quotes are optional. It does not need to be an actual key contained in the file. If the file being LISTed is an indexed file rather than a key oriented file, the end-key must be a numeric constant, representing the last index to process in the file.

When used in conjunction with the ALTFILE keyword or the ALTKEY keyword, the key range defined should be for the sort file or alternate key, rather than the primary file or key.

The END parameter can be prefixed by one of the following modifiers to convert the value:

CND converts a literal date to a Julian integer. **CNDI** converts a literal date to a Julian integer, compressed into 4-byte binary form. **CNI** converts a literal number into a 4-byte binary form. These conversions coincide with BBx template sizes and uses.

For example, **END CND "3/15/95"** would convert 3/15/95 to a Julian number, and use the result (in BBx, 2449792) as the BEGIN value. **END CNDI "3/15/95"** would perform the same date conversion, but the value would be further converted to a 4-byte integer with the BIN function (BIN(2449792,4)). **END CNI 99995** would convert 99,995 to a 4-byte integer with the BIN function.

In addition, if the END parameter is enclosed in single-quotes (‘), the parameter is interpreted as an expression.

This example show an expression:
LIST DEMO.CUST ID NAME BEGIN ‘str([[Enter customer number]]:”00000”)’

The expression can use functions defined in the file “userdefn.txt”, if mult-line function capabilities are required.

EXAMPLES

The following example will process records from the beginning of the file until the record keys become greater than "001" ("00100", "00101", etc.):

LIST DEMO.CUST ID NAME PHONE **END 001**

This example show an expression:

LIST DEMO.CUST ID NAME END ‘str([[Enter ending customer number]]:”00000”)’

The expression can use functions defined in the file “userdefn.txt”, if multi-line function capabilities are required.

EVEN

PURPOSE To generate an even numeric value, suitable for a break point to group numbers in the same range together.

FORMAT LIST **... BREAK EVEN** ***number field-name...***

SYNONYMS None

USAGE

The EVEN keyword is used to formulate a number into an even increment value. EVEN requires a *number* parameter, that can be any positive integer value from 1 to 99,999,999 (entered without commas). Each number in the column following the number is then formulated to an even value based on this formula:

number * INT(*value/number*)

For example, if EVEN 1000 were used, any number from 0 to 999.99 would be converted to 0; from 1000 to 1999.99, converted to 1000; from 2000 to 2999.99, converted to 2000, and so on. In a report sorted by a numeric field, a break point could be specified based on even values, to cause a break each time the numeric field reached a certain threshold.

EXAMPLES

This example will produce a report sorted by sales, with breakpoints each 5,000.

LIST DEMO.CUST ID NAME TOTAL YTD.SLS SBREAK **EVEN 5000** YTD.SLS SORT DSND YTD.SLS

EXPORT

See ASCII in this chapter.

FIELD

PURPOSE To indicate that more report fields are to be specified after a sort- or select-phrase has been entered.

FORMAT LIST ... sort- or select-phrase **... FIELD *field-phrase...***

SYNONYMS FLD, SHOW

USAGE

The FIELD keyword allows more flexibility in the placement of data elements in the command line. With it field elements may be entered after a sort- or select-phrase and not be interpreted as being included.

EXAMPLES

The following example shows a sort-phrase followed by the FIELD keyword to allow specification of data fields:

LIST DEMO.CUST ID NAME SORT NAME **FIELD** SLSP PHONE YTD.SLS

FILL

PURPOSE To fill the balance of any line with a specified character.

FORMAT LIST **... FILL "*character*"...**

SYNONYMS None

USAGE

The FILL keyword can be used anywhere in a field phrase (including a header or footer sub-phrase), to print a line of characters from the current column to the right margin.

The TAB keyword can be used to place the start of the line.

The *character* can be any single, printable character, including a space.

EXAMPLES

This example shows two subsequent fills, first dashes starting at column 20, then spaces starting at column 60, to draw a 40 character line centered in an 80 column report:

LIST DEMO.CUST WIDTH 80 ON PRINTER TAB 20 ID NAME PHONE TAB 20.2 **FILL "-"** TAB 60.2 **FILL " "** TAB 25.3 YTD.SLS TAB 40.3 YTD.COST DBLSPC

FOOTER*n*

See HEADER*n* in this chapter.

FULLCASE

PURPOSE To cause a sort phrase field to be sorted in a case-sensitive manner.

FORMAT LIST **... SORT ... FULLCASE** ***field-name...***

SYNONYMS FCASE

USAGE

When GENERAL sorts a file, it normally converts all text fields to upper case, so that multiple spellings of the same value will be printed together. For instance, values of "SMITH" and "Smith" would be sorted together. If the sort order needs to take upper- and lower-case into account, then the FULLCASE keyword can precede the affected field in the sort phrase.

EXAMPLES

This example shows the full case keyword:

LIST DEMO.CUST ID NAME TOTAL YTD.SLS SORT STATE **FULLCASE** CITY

HEADER

PURPOSE To define a column header other than the default, which utilizes the column headings defined for the fields specified.

FORMAT LIST **... HEADER *"header-text"...***

SYNONYMS HEAD, HEADING

USAGE

Normally, the column headings are generated from the Heading element of the field definition for each data element on the report. The heading elements are formatted in the same manner as the data fields, including spacing, tabs, and new lines. If this column heading is not desired the HEADER keyword may be used to set the column headings to some specified text.

The header-text entered must be enclosed in quotes.

Column headings may be turned off with the NO-HEAD keyword. A complete page header, including titles and column headings, may be described with the PAGE-HDR or HEADER0 keywords.

EXAMPLES

The following example will produce the heading specified:

LIST CUSTOMERS ID NAME SLM.NAME PHONE SORT SLM **HEADER "CUSTOMER INFORMATION:"**

HEADER*n*
FOOTER*n*
OFF

PURPOSE To specify custom page and group headers, or custom group or report footers.

FORMAT LIST **... HEADER*n* *header-phrase* OFF**...
LIST **... FOOTER*n* *footer-phrase* OFF**...

Note there are no spaces between the HEADER/FOOTER words and the *n* level indicator.

SYNONYMS None

USAGE

The custom header and footer keywords are toggles that turn on a custom header or footer phrase. In these phrases, any text, data field, calculation, or positioning keyword is legal, and applies to that particular header or footer.

The header or footer level, *n*, applies to the break point level, as defined by the order of BREAK keywords in the report. Report breaks are nested as they are encountered in the LIST command, so the first BREAK becomes level 1, the second is level 2, and so on. Up to 9 levels may be specified.

Level 0 is defined as the "report level". A header of level 0 is considered a definition of the page heading, while a footer of level 0 is considered the report footer, which prints only at the end of the report.

Note that HEADER0 can be used as a replacement for the PAGE-HDR keyword, and in fact, GENERAL internally translates a PAGE-HDR definition into a HEADER0 header phrase.

Header and footer phrases are active until explicitly turned off. They are turned off by:

- Another HEADER*n* or FOOTER*n* phrase.
- The OFF keyword
- The end of the LIST command

EXAMPLES

This example shows a custom report header:

```
LIST DEMO.CUST TAB 10.1 ID TAB 10.2 NAME TAB 30.1 PHONE TAB 30.2
DEMO.SLSP:NAME
```

HEADER0 TAB 1.1 @DATE TAB 1.2 "PAGE" @PAGE CENTER "CUSTOMER PHONE LIST" TAB 1.3 FILL "-" TAB 10.4 "CUSTOMER ID/NAME" TAB 30.4 "PHONE/SALESPERSON" LF FILL "=" OFF

This example shows a custom footer for break level 1, calculating a group level value. Note the use of the group summary function GSUM() to derive level 1 totals for YTD.SLS and YTD.COST.

LIST DEMO.CUST SBREAK SLSP ID NAME TAB 55 TOTAL YTD.SLS **FOOTER1 TAB 55.1 "----------" TAB 10.2 "SALESPERSON:" @BREAK1 TAB 55.2 @CALC1("N,L14,2,P","",GSUM(YTD.SLS,1)) TAB 10.3 "(PROFIT)" TAB 55.3 @CALC2("N,L14,2,P","",@CALC1-GSUM(YTD.COST,1)) OFF**

HEIGHT

PURPOSE To force the number of lines each record occupies on the report, regardless of data positions and the NO-BLANK keyword.

FORMAT LIST **... HEIGHT** ***lines...***

SYNONYMS None

USAGE

When GENERAL produces a report, the number of lines each record occupies is determined by the positions defined in the report. If the last field is placed on line 3, as with a TAB 10.3 keyword, then GENERAL will allocate 3 lines per record. If the NO-BLANK keyword is used, and one of the lines is empty, then that record will only use 2 lines.

If a report requires a fixed number of lines to be allocated, regardless of how many are used, then the HEIGHT keyword can be used.

EXAMPLES

The following example shows how to use the HEIGHT keyword in producing fixed length labels while suppressing blank lines:

LIST DEMO.CUST TAB 2.2 NAME TAB 2.3 ADDR1 TAB 2.4 ADDR2 TAB 2.5
CITY_ST_ZIP NO-BLANK NO-PAGE **HEIGHT 6** ON PRINTER

LBREAK

PURPOSE To generate a line break every *n* lines of a report, so that a blank line is printed between even groups of records.

FORMAT LIST **... LBREAK *n***...

SYNONYMS LBRK

USAGE

The LBREAK keyword is used to generate a blank line every time the specified number of lines has printed without any breaks.

The line break counter is reset at the top of each page and after any break point as defined by a BREAK keyword.

Note that the line count is based on records printed, even if each record prints on multiple physical lines of output.

EXAMPLES

This example will generate a line break every five lines:

LIST DEMO.CUST ID NAME SLSP PHONE **LBREAK 5**

LENGTH

PURPOSE To set the number of printed lines per page.

FORMAT LIST **... LENGTH #-*lines*...**

SYNONYMS LINES, LEN

USAGE

The LENGTH keyword is used for page formatting, allowing specification of the number of printed lines per page. This keyword is only required if the default settings used by GENERAL are incorrect. The default settings are based on the Business Basic knowledge of the device dimensions, typically 66 lines for printers and 24 or 25 lines for VDTs. For VDTs, GENERAL deducts 4 lines for a bottom margin; for printers and files, GENERAL deducts 6 lines.

#-Lines must be an integer and must be greater than 1. Note that this specifies the number of *printed* lines, not the number of *physical* lines.

EXAMPLES

The following example shows the use of the LENGTH keyword for an 88-line page, with a 1" allocation for margins:

LIST DEMO.CUST ON LP ID TAB 10.1 NAME TAB 10.2 ADDR1 TAB 10.3 ADDR2 TAB 10.4 CITY_ST_ZIP DBL-SPC **LENGTH 80**

LINK

PURPOSE To define a new link specification, or override an existing link specification for the LIST file, for use by a link field definition.

FORMAT LIST **... LINK FROM** ***file-name*** **TO** ***file-name*** **FULL** ***key-expression*** **OFF...**
LIST **... LINK FROM** ***file-name*** **TO** ***file-name*** **PART** ***key-expression*** **OFF...**

SYNONYMS None

USAGE

The LINK keyword defines a *link specification* within a LIST command. Following the link specification, link field definitions referencing the files will use the key-expression and full- or partial-key indicator. Note that normally, links are pre-defined in the dictionary, so it is rare to use a link keyword in a report.

The FROM file specifier is optional. If it is not present, then the FROM file will default to the LIST file. If specified, the file must be defined in GENERAL's dictionary or in a supported external dictionary.

The TO file specifier is required, and indicates what file is to be read from. The file-name here can contain a sort specifier *@sort-name* to process the records in the TO file in a non-primary key order. Sort specifications are defined in the TO file's dictionary.

The parameter FULL or PART (or FULLKEY or PARTKEY) is required, and indicates whether the link is a full- or partial-key type of link. Partial-key links produce from 0 to many records, while a full-key link produces one record.

The key-expression parameter is an expression using fields and data in the FROM file to derive the key or partial key needed to access records in the TO file.

The OFF parameter is required, and indicates the end of the key expression, and a return to the field phrase.

More information about link specifications can be found in the Dictionary chapter.

EXAMPLES

This example shows a run-time link definition:

LIST DEMO.INVOICES **LINK TO DEMO.PAYMENTS PART CUSTOMER+INVOICE OFF** CUSTOMER INVOICE TOTAL AMOUNT TOTAL @SUM:PAYMENTS:PAYMENT

If there is already a link defined to DEMO.PAYMENTS in the DEMO.INVOICES dictionary, then the link specified in the LIST command will override the link in the dictionary.

LOWER

See PROPER in this chapter.

MAXIMUM

PURPOSE To cause a numeric or date maximum to be calculated, with the value output at the end of the report, and at break points if specified by the BREAK keyword.

FORMAT LIST **... MAXIMUM** ***field-name...***

SYNONYMS MAX

USAGE

The MAXIMUM keyword is used to calculate the highest numeric or date value of a specified report column. The value so calculated will be printed at the end of the report and also at each subtotal break point if the BREAK keyword is used.

The field-name specified must be defined for the file being LISTed. It also must be defined as a numeric or date type field.

EXAMPLES

The following example will print the highest sales value of each line on the report:

LIST DEMO.CUST ID NAME SLSP **MAXIMUM YTD.SLS**

MINIMUM

PURPOSE To cause a numeric or date minimum to be calculated, with the value output at the end of the report, and at break points if specified by the BREAK keyword.

FORMAT LIST **... MINIMUM** ***field-name...***

SYNONYMS MIN

USAGE

The MINIMUM keyword is used to calculate the lowest numeric or date value of a specified report column. The value so calculated will be printed at the end of the report and also at each subtotal break point if the BREAK keyword is used.

The field-name specified must be defined for the file being LISTed. It also must be defined as a numeric or date type field.

EXAMPLES

The following example will print the smallest sales value of each line on the report:

LIST DEMO.CUST ID NAME SLSP **MINIMUM YTD.SLS**

MONTH
MMMYYYY
MMMYY
MMYY
MMYYYY
YYYYMM
YYMM

PURPOSE To convert the precision of output for a date field to a month and year, rather than a month, day, and year.

FORMAT LIST **... MONTH** ***field-name...***

SYNONYMS None

USAGE

The date conversion keywords are used to convert a date field to display in a month and year format. Which particular keyword is used determines the format.

MONTH, MMMYYYY	Jan/2002
MMMYY	Jan/02
MMYY	01/02
MMYYYY	01/2002
YYYYMM	2002/01
YYMM	02/01

Date conversion can be useful when a break point by month is desired. See also the YYYY keyword.

EXAMPLES

The following example shows the generation of a date sorted report, with subtotals by month:

LIST DEMO.INVOICES CUSTOMER INVOICE DATE TOTAL AMOUNT SBREAK
MMMYY DATE SORT DATE

NEWLINE

PURPOSE To cause the report output to skip to the next line.

FORMAT LIST ... field-name **NEWLINE** field-name...

SYNONYMS LF

USAGE

NEWLINE is a formatting keyword used to force the report output to the left margin of the next line. This feature is useful, especially in combination with the TAB keyword, for formatting stacked lines of output on the report.

Note that the TAB keyword provides both column and row positioning, so the NEWLINE keyword isn't always necessary to formulate multi-line reports.

EXAMPLES

This example shows using NEWLINE to format mailing labels.

LIST DEMO.CUST **NEWLINE** NAME **NEWLINE** ADDR1 **NEWLINE** ADDR2 **NEWLINE** CITY_ST_ZIP **NEWLINE** NO-PAGE ON LP

NO-BLANK

PURPOSE To suppress the printing of blank lines on a report.

FORMAT LIST **... NO-BLANK...**

SYNONYMS NB

USAGE

This keyword is used to suppress the printing of blank lines of output. This can be useful when you are stacking many fields on a report, some of which may be blank.

Note that the DOUBLE-SPC keyword forces a blank line between records, even if the NO-BLANK keyword is used.

EXAMPLES

This example shows the use of the NO-BLANK keyword:

LIST DEMO.CUST ID TAB 10 NAME LF TAB 10 ADDR1 LF TAB 10 ADDR2 LF TAB 10 CITY STATE ZIP DOUBLE-SPC **NO-BLANK**

NO-DETAIL

PURPOSE To cause the printing of a "totals only" report.

FORMAT LIST **... NO-DETAIL...**

SYNONYMS ND

USAGE

This keyword is used to suppress the printing of report detail. Instead, only totals and subtotals are printed.

When the totals are printed, the only fields that will print are those denoted by AVERAGE, BREAK, COUNT, MAXIMUM, MINIMUM, PCT-TOTAL, and TOTAL keywords. All other fields are suppressed.

When this keyword is used, output may be slow and sporadic. This is because the detail is still being processed.

If totals are not printed vertically, then line spacing and underlines are eliminated from the highest break level.

EXAMPLES

This example shows the use of the NO-DETAIL keyword to create a report to count the number of customers per salesperson:

LIST DEMO.CUST BREAK DEMO.SLSP:NAME COUNT ID **NO-DETAIL** SORT DEMO.SLSP:NAME

NO-DUPLICATES

PURPOSE To suppress the printing of duplicate subsequent values in a report column.

FORMAT LIST **... NO-DUPLICATES** ***field-name...***

SYNONYMS NO-DUPS, NO-DUP, NDP, NDU

USAGE

The NO-DUPLICATES keyword provides the ability to enhance the appearance of reports that include repetitive elements.

A typical use of this might involve a transaction file that contains or links to master file information. When LISTing the transaction file, master file information will be repetitive, and the NO-DUPLICATES keyword can precede any master file related columns to suppress the printing of the duplicate elements.

EXAMPLES

The following example will suppress duplicate customer numbers and names in an invoice listing:

LIST DEMO.INVOICES BREAK-PG **NO-DUP CUSTOMER NO-DUP**
DEMO.CUST:NAME DATE TOTAL AMOUNT ON PRINTER

NO-HEAD

PURPOSE To prevent the automatic generation of column headings.

FORMAT LIST **... NO-HEAD...**

SYNONYMS NH

USAGE

This keyword is used to suppress printing of column headings. GENERAL normally creates column headings automatically, or the HEADER, HEADER0, and PAGE-HDR keywords may override them. If they are not overridden and the NO-HEAD keyword appears on the command line, column headings are suppressed.

The report title will be printed and pagination still occurs. To eliminate page breaks and titles, use the NO-PAGE keyword.

EXAMPLES

This example shows the use of the NO-HEAD keyword:

LIST DEMO.CUST ID NAME MAXIMUM MINIMUM YTD.SLS **NO-HEAD**

NO-PAGE

PURPOSE To prevent the automatic generation of page breaks and page headings.

FORMAT LIST **... NO-PAGE...**

SYNONYMS NP

USAGE

This keyword causes GENERAL to suppress the printing of page headings. Normally, page headings consist of the date, title, and page number followed by the column headings. The title and column headings may be created by GENERAL or may be defined by the HEADER and TITLE keywords. If NO-PAGE appears anywhere in the command line, the page headings are suppressed.

In addition to suppressing page headings, this keyword also suppresses pagination. Instead of issuing a page eject at the bottom of a printed page, the report will run continuously from start to finish. This can be useful when producing mailing labels or when producing a report to disk that will be loaded by another application, such as a word processor.

EXAMPLES

This example shows the use of the NO-PAGE keyword:

LIST DEMO.CUST NAME ADDR1 ADDR2 CITY STATE ZIP PHONE WIDTH 200 **NO-PAGE** ON FILE

NO-QUOTE

PURPOSE To disable the normal quoting of text and date values in a delimited export.

FORMAT LIST ... EXPORT DELIMITED **... NO-QUOTE...**

SYNONYMS None

USAGE

The NO-QUOTE keyword suppresses the automatic quoting of certain fields in a delimited export. Normally, both text and date fields are quoted, while numeric fields are not. Each field is then separated by a delimiter, which defaults to a comma but which can be overridden by the DELIMITER keyword.

EXAMPLES

This example shows an export with no quotes around text fields:

LIST DEMO.CUST NAME ADDR1 ADDR2 CITY STATE ZIP EXPORT DELIMITED DELIMITER "~009" **NO-QUOTE** ON FILE

NO-RECAP

PURPOSE To suppress the automatic printing of a recap page for any given report.

FORMAT LIST **... NO-RECAP...**

SYNONYMS NRC

USAGE

GENERAL normally prints a recap page after each report. The recap page displays the LIST command used to generate the report, time and file statistics, run-time replacement substitutions, and errors.

To suppress the printing of this information, include the NO-RECAP keyword in the field phrase of the command.

The NO-PAGE keyword also suppresses recap printing.

EXAMPLES

This example shows the use of the NO-RECAP keyword:

LIST DEMO.CUST BREAK SLSP DEMO.SLSP:NAME COUNT ID **NO-RECAP** SORT DEMO.SLSP:NAME

NO-WKFL

PURPOSE To disable report page buffering when a report is LISTed to the VDT. This differs from older versions of GENERAL.

FORMAT LIST **... NO-WKFL...**

SYNONYMS NW, SCAN

USAGE

When a report is printed to the VDT, GENERAL will normally buffer the output to allow the user to scroll backwards and forwards through the report. This buffering can use a large amount of disk space if the report is large, so including this keyword in the LIST command can turn off buffering.

EXAMPLES

The following example would print to the VDT, but disable page buffering:

LIST DEMO.INVOICES BREAK CUSTOMER INVOICE DATE TOTAL BALANCE **NO-WKFL**

OFF

See HEADER*n* and LINK in this chapter.

ON

PURPOSE To cause output to go to a specified device, such as a printer, rather than to the VDT.

FORMAT LIST **... ON *printer-name...***
LIST ... ON "*file-name*"...
LIST ... ON VDT...
LIST ... ON PRINTER...
LIST ... ON FILE...
LIST ... ON PREVIEW...

SYNONYMS TO

USAGE

This keyword allows reports to be submitted to printers or text files. Unless otherwise specified by this keyword, GENERAL produces all reports on the VDT. If the output is desired on a printer, or if a text file is desired for interface to a spreadsheet or word processing application, then this keyword must be used.

The printer-name may be any valid printer device name (such as LP or P1). The export-file must be a valid quoted file-name for your system, and if it exists, it must be an ASCII text file. Additionally, you may specify the reserved GENERAL device name VDT, which indicates the VDT.

If an export-file is specified, and the file does not exist, GENERAL will create it. If it does exist, GENERAL will prompt to ensure that you want to overwrite it.

Warning: *If the selected export-file exists, running your report will destroy any data in it. Be sure that any file you select does not contain valuable data that should be saved. Some files and/or directories are protected automatically by GENERAL, while others can be protected by site. See the Installation section of the Appendix for more detail.*

GENERAL interprets two special names at run-time to prompt the user for proper values:

ON PRINTER will provide a selection window of configured printers, the VDT, or a file. **ON FILE** will issue a prompt for a filename.

Another special name, **ON PREVIEW**, may be used to specify that the report should be printed to a file in the dimensions of the default printer (or a custom size if WIDTH and/or LENGTH is specified). When the report is complete, GENERAL displays it on screen, with full scrolling capabilities, and allows printing from that point to a printer or disk file.

EXAMPLES

This example shows routing of output to the printer "LP":

LIST DEMO.CUST **ON LP** ID NAME CLASS PHONE

This example shows how to generate a file for interface to a spreadsheet:

LIST DEMO.CUST ID NAME MTD.SLS YTD.SLS SLSP SORT DESCENDING YTD.SLS
SELECT YTD.SLS > 500 EXPORT DELIMITED **ON TSFR.PRN**

PAGE-HDR

PURPOSE To allow definition of a complete page heading for use by the LIST command.

FORMAT LIST **... PAGE-HDR *"heading-definition"...***

SYNONYMS PGHDR, PH
See also HEADER*n*.

USAGE

This keyword provides the ability to completely control the page headings of a report. Normally, GENERAL produces a page heading consisting of a date, title, page number, and column headings. The title and column headings can be controlled with the TITLE and HEADER keywords. The PAGE-HDR keyword supersedes these other items when used.

The heading definition, which must be enclosed in quotes, consists of text intermixed with mnemonic keywords, all of which are enclosed in single quotes ('). The text is presented on the page heading exactly as typed, while the keywords are replaced at run-time with values or formatting actions. Internally, GENERAL converts PAGE-HDR definitions to HEADER0 definitions at run-time.

MNEMONIC KEYWORDS

'BREAK *n*' is replaced with the current level *n* break point value. Levels are counted from the left as BREAK keywords (or BREAK-PG) are found in the LIST command.

Synonym: 'BRK *n*'

'CENTER *text* is replaced by the text defined, centered on the page. The text may be literal or may incorporate any of these keywords: 'DATE', 'TIME', 'PAGE', or 'BREAK'.

Synonym: 'C *text*'

'DATE' is replaced by the date in *mm/dd/yy* format.

'LINE *char*' is replaced by a new line of *char* characters. If no character is specified, a dash (-) is used. The line generated is the full width of the page.

'NEWLINE' causes the page heading information to skip to the next line, just as the NEWLINE keyword does with report line formatting.

Synonym: 'LF'

'PAGE' is replaced by the page number in 5 digit format.

'SPACE *n*' expands to *n* spaces, just as the SPACE keyword does in report line formatting.

Synonyms: 'SPC *n*'

'TAB *n*' causes the next item immediately following the keyword to be printed at column *n*.

'TIME' is replaced by the time in *hh:mm am/pm* format.

EXAMPLES

An example of a PAGE-HDR definition:

LIST DEMO.CUST ID TAB 10 NAME TAB 40 PHONE LF TAB 10 CITY STATE ZIP
PAGE-HDR "'DATE' 'CENTER CUSTOMER MASTER LISTING' 'TAB 70' PAGE 'PAGE''LF''TIME' 'CENTER COMPANY: 'BREAK 1'' 'LINE'CUST 'TAB 10'CUSTOMER 'LF'NO. 'TAB 10'NAME & CITY 'TAB 40'TELEPHONE 'LINE ='"

PCT-TOTAL

PURPOSE To cause a percent-of-total calculation to be performed on a numeric field, with the value(s) printed on each report line, as well as at subtotal break points and at the end of the report.

FORMAT LIST **... PCT-TOTAL** ***field-name...***

SYNONYMS PCT-TOT, %TOTAL, %TOT

USAGE

This keyword is a complement to the TOTAL keyword. Instead of printing the data and producing totals at the end of the report and at subtotal breaks, the PCT-TOTAL keyword causes the percentage of the grand total to be produced on each line at all total and subtotal break points.

GENERAL always uses two decimal points and a "%" in the output (i.e. 55.00%), so field definitions should be at least 7 columns wide to prevent truncation.

The field-name used must be a numeric field definition. In addition, if PCT-TOTAL is used, then AVERAGE, COUNT, MAXIMUM, MINIMUM, and TOTAL are automatically turned off if they are used for the field-name, as PCT-TOTAL renders those calculations meaningless.

Note: *To calculate the percentage, GENERAL must first scan all the records to produce a grand total. Sorting and selecting takes place concurrently if required. If the data used in the calculation changes before the report is output, the percent-of-total calculation may be incorrect.*

EXAMPLES

This example will generate a percent of total column next to a standard total column:

LIST DEMO.CUST ID NAME TOTAL YTD.SLS **PCT-TOTAL YTD.SLS**

PLOT

PURPOSE To generate a cross-tabulation or summary report, which may be displayed, manipulated, printed, and plotted, if the appropriate plotting utilities and devices are available.

FORMAT LIST **... PLOT...**

SYNONYMS GRAPH, TABULATE

USAGE

The PLOT keyword is used to generate a cross tabulation table rather than a report. Once the table is displayed, GENERAL allows manipulation of the results, and printing and/or charting of the results.

The BREAK keyword and any aggregate keywords, such as TOTAL or AVERAGE define the format of the cross tabulation table. Only the first aggregate keyword applied to any field is used in the calculation of values for the table (TOTAL AVERAGE YTD.SLS only uses TOTAL, for example).

If there is one BREAK keyword, then the table is produced as a summary table, with group break points listed down the left side of the table as row labels, and up to twenty aggregate field descriptions listed across the top, as column labels. The values for each aggregate field, calculated as subtotals for the break point, are displayed in the table body.

If there is more than one BREAK keyword, then GENERAL displays a cross tabulation, where the first BREAK value determines group break points for row labels, and the second BREAK keyword determines the value of the column labels. The first aggregate value in the LIST command is used to generate table values.

Manipulation
Once the table is displayed, several function keys can be used to manipulate the table values.

F2-sort cols causes GENERAL to sort the columns in descending order, based on the total value of the columns.

F3-rows causes the sorting of rows based on the value of a specified column. If more than one column exists, GENERAL will present a selection window of the columns.

F4-compress is used to reduce the size of the table. There are three compress options:

Compress Rows/Columns prompts for the number of rows and/or columns that the table should contain. Any rows or columns over this limit are summarized into a value labeled OTHER, which becomes the last row or column in the table. It is often desirable to sort the rows and/or columns before compressing them.

Delete Row prompts for a specific row to delete from the table then redisplays the table without that row.

Delete Column prompts for a specific column to delete then redisplays the table.

F5-reload will recalculate the table based on the work file created by the LIST command.

F6-print will prompt for a printer, and print the values in the table.

EXAMPLES

This example will produce a three column summary table:

LIST DEMO.CUST **BREAK SLSP TOTAL YTD.SLS TOTAL YTD.COST TOTAL YTD.PROFIT PLOT**

This example will produce a cross tabulation, using the first two break points as row and column labels:

LIST DEMO.CUST BREAK SLSP BREAK STATE TOTAL YTD.SLS PLOT

PREFIX
SUFFIX

PURPOSE To add a prefix and/or suffix to each line of a delimited export.

FORMAT LIST ... EXPORT DELIMITED **... PREFIX *"text"...***
LIST ... EXPORT DELIMITED **... SUFFIX *"text"...***

SYNONYMS None

USAGE

In an export, it is sometimes necessary to add characters before or after the records in the export text file. The PREFIX keyword is used to place one or more characters at the beginning of each line; SUFFIX at the end of each line.

To include non-printable characters in the text, use GENERAL's ASCII notation. A carriage return (ASCII 13) would be entered as "~013".

EXAMPLES

The following export adds a <tab> character as both a prefix and a suffix:

LIST DEMO.CUST ID NAME YTD.SLS EXPORT DELIMITED DELIMITER "~009"
PREFIX "~009" SUFFIX "~009" ON FILE

PROPER
UPPER
LOWER

PURPOSE To convert the case of a text field to proper (capitalized words), upper, or lower.

FORMAT LIST **... PROPER** ***field-name...***
LIST **... UPPER** ***field-name...***
LIST **... LOWER** ***field-name...***

SYNONYMS None

USAGE

These case conversion keywords can precede any text or date field on a report. GENERAL will apply the appropriate case conversion to the output.

PROPER conversions are based on an algorithm that looks for spaces and punctuation, and capitalizes the next subsequent letter. All other letters are forced to lower case. In some cases, this may not be what is intended: "JOHN SEYMOUR, MD", for example, would become "John Seymour, MD". The "MD would be properly handled if it was instead "M.D.".

PROPER case conversions will slow down report processing somewhat. UPPER and LOWER will have very little effect on performance.

EXAMPLES

This example shows the use of the PROPER keyword to export converted names for a mail merge:

LIST DEMO.CUST **PROPER NAME PROPER ADDR1 PROPER ADDR2 PROPER CITY STATE ZIP** EXPORT WORDPERFECT ON "/usr/wp/cust.mrg"

This example will print all upper case names:

LIST DEMO.CUST ID **UPPER** NAME SLSP TYPE TOTAL YTD.SLS

RETAIN

PURPOSE To save a work file on disk for possible use by another application. Normally, GENERAL erases work files when a report is complete.

FORMAT LIST **... RETAIN**

SYNONYMS None

USAGE

GENERAL normally erases the work file used when a sort- or select-phrase is present, once the LIST command, which generated it, is complete. However, some applications may require that the work file be retained on the disk for further processing, either by GENERAL or by another application. By including this keyword in the LIST command, the work file will be retained.

Note that the next time the current terminal uses a LIST command that requires the work file, it will be erased and redefined. Work files are terminal-dependent.

EXAMPLES

This command uses the RETAIN keyword:

LIST DEMO.CUST ID NAME DEMO.SLSP:NAME YTD.SLS SORT SLSP **RETAIN**

SBREAK, SBREAK-PG

See BREAK, BREAK-PG in this chapter.

SELECT

PURPOSE To designate the start of a select-phrase.

FORMAT LIST **... SELECT *select-phrase...***
LIST ... **SELECT FROM *Select-phrase***...

SYNONYMS SEL, WITH

USAGE

The SELECT keyword indicates the beginning of a select-phrase. A select-phrase is used to limit the records selected for output based on certain selection criteria. Selection criteria are defined by one or more relational expressions involving data fields, relational operators, and numeric or string constants.

The SELECT phrase also recognizes the four basic arithmetic operators, so relational expressions can utilize simple addition. Relational operators must be taken from this list:

Operator	Relation Implemented
=	Equal to
<> or ><	Not equal to
>	Greater than
<	Less than
>= or =>	Greater than or equal to
<= or =<	Less than or equal to
==	Contains
+=	Contains at Position 1
-=	Does not contain
~=	Approximately equal (regular expression)

Multiple expressions may be connected with logical connectors AND and OR. In addition, multiple expressions may be nested: GENERAL may use parentheses to control the testing of the various expressions.

The "contains" operators are only valid for text fields.

SELECT FROM Usage

The SELECT FROM keyword phrase indicates to GENERAL that the next word in the command is a file name, and all text following until another keyword is a select phrase to be applied to records in the file specified. Any SELECT FROM phrase is independent of any other, applying only to the file specified. If a SELECT FROM phrase specifies a file that isn't used by the report, it is ignored.

Auto Select Phrases

Internally, the criteria are added to the target file's Auto Select phrase during report parsing. In cases where there is already an Auto Select phrase, then this criteria is added as a "group and". For example, if the DEMO.INVOICES file had an Auto Select phrase of SLSP="101", then the result of the above command would be an Auto Select phrase of (SLSP="101") AND (AMOUNT > 500 OR DATE > "1/1/92"). There is no permanent change made to the file's dictionary.

PROMPT Mode Access to SELECT FROM

A PROMPT mode option, called "Linksel", is available from the main PROMPT ring menu. Choosing Linksel will present a entry window where up to 40 files and associated SELECT FROM phrases can be entered.

In the FILE column, you can enter a file name, or choose from one of two lists. The F2-linked file list will present a selection of just those files linked in the dictionary to the primary file for the report. The links shown will only be partial key link types. The F5-all file list allows access to all the files in the data dictionary.

In the CRITERIA column, enter the select phrase you want to associate with the file. Use the F2-list fields key to paste fields from the chosen file into the criteria text.

Parsing Considerations

While SELECT FROM phrases are passed through the command line parser, they are not evaluated there, so keywords such as CONVERT-DATE will not work. Instead, they are evaluated by the expression parser as part of an IF..THEN statement. This means that the functions used in calculations will work, provided they are protected from the command line parser's manipulations by single quotes. Specifically, if a phrase contains functions (like DT(), STR(), JUL(), etc.), it needs to be single-quoted.

SELECT FROM DEMO.INVOICES DATE >= DT("9/1/92") will not be parsed correctly because the DT() function is manipulated by the command line parser before it gets to the expression parser. **SELECT FROM DEMO.INVOICES 'DATE >= DT("9/1/92")'** will work, because the single quotes protect the expression.

Note that the above example would also work without the DT() function, because the expression parser will automatically convert literal dates when found in a pattern of *date-field operator "literal-date"*.

EXAMPLES

The following example shows a command using the SELECT keyword:

LIST DEMO.CUST ID NAME YTD.SLS **SELECT SLSP="101" AND (YTD.SLS>1000 OR YTD.COST > 750)**
This example shows a select phrase that prints invoices dated 30 days or more before a prompted date:

LIST DEMO.INVOICES CUSTOMER INVOICE BALANCE DATE SELECT DATE - 30 <= [["Enter date cutoff",D]]

Here is an example using the SELECT FROM syntax:

LIST DEMO.CUST ID NAME DEMO.INVOICES:INVOICE DEMO.INVOICES:AMOUNT **SELECT FROM DEMO.INVOICES AMOUNT > 500 OR DATE > "1/1/92"**

SORT

PURPOSE To designate the start of a sort-phrase.

FORMAT LIST **... SORT** ***sort-phrase...***

SYNONYMS SRT, BY

USAGE

The SORT keyword indicates the start of a sort-phrase. A sort-phrase is used to control the sorting order of the report output. Typical examples might be sorting by name, product categories, descending dollar values, and so on. Without a sort-phrase, the record key of the file being LISTed sorts a report.

A sort-phrase consists of one or more field-names. The field-names must have been previously defined for the file being LISTed. Each field may optionally be preceded by the DESCENDING keyword to cause that field to be sorted in descending sequence.

Sorts, which resolve to the same value, are then sorted automatically on another level by the key of the List file. This can cause confusion if sorts and breaks are based on related, but not identical, values. Ties in the sort will be resolved by the file's primary key, possibly causing break points to occur out of order.

For a more detailed description of sort-phrases, see the LIST command.

Also see the ALTFILE, ALTKEY, and ALTSORT keywords, which may sometimes be used in place of a sort-phrase, and which allow faster processing of a report.

EXAMPLES

This example shows the use of a sort-phrase:

LIST DEMO.CUST ID NAME BREAK SLSP TOTAL YTD.SLS ON LP **SORT SLSP DESCENDING YTD.SALES**

SPACE

PURPOSE To specify spacing between two fields as other than the default of one space.

FORMAT LIST ... field-name **SPACE** ***spacing*** field-name...

SYNONYMS SPC

USAGE

This keyword is used to specify the spacing between any two data fields on a report. GENERAL will automatically place one space between each fields except when the SPACE, TAB, and NEWLINE keywords are used.

Spacing may be any number from 2 to 999.

EXAMPLES

The following example shows the placement of 20 characters between two fields:

LIST DEMO.CUST ID NAME **SPACE 20** YTD.SLS

STOP

PURPOSE To stop a report printing after *n* lines.

FORMAT LIST **... STOP *n*...**

SYNONYMS None

USAGE

When printing a report, sometimes it is useful to only print a certain number of lines. An example might be to sort customers by descending sales, cutting off the report at the first 10 customers.

EXAMPLES

The following example will print the first 10 lines of a customer report:

LIST DEMO.CUST NAME PHONE CITY_ST_ZIP YTD.SLS **STOP 10** SORT DSND YTD.SLS

SUFFIX

See PREFIX in this chapter.

SUMMARY

PURPOSE To produce a summary report after a normal report.

FORMAT LIST **... SUMMARY...**

SYNONYMS None

USAGE

GENERAL provides a NO-DETAIL keyword to produce summary reports. When that keyword is used, only subtotal and report-total lines are printed. The SUMMARY keyword provides a means of printing such a totals-only report automatically after printing a detailed report.

EXAMPLES

The following example will print first a detailed customer listing by salesperson, then a summary of totals by salesperson:

LIST DEMO.CUST SLSP TAB 20 NAME TAB 60 TOTAL YTD.SLS **SUMMARY** SORT SLSP ON PRINTER

TAB

PURPOSE To specify the column position and optionally the row number of the next field on the report.

FORMAT LIST **... TAB *column*** field-name
LIST ... **TAB *column.row*** field-name

SYNONYMS @

USAGE

The TAB keyword is used to align the next data field at a specific position. If just a column is specified, then the current row is used. If both column and row data is specified, then the next field is placed at that position, and the current row is updated.

The column should be an integer between the current report column and last column position on the report line (see the WIDTH keyword).

EXAMPLES

This example shows the TAB keyword used to produce a name and address list:

LIST DEMO.CUST ID **TAB 20** NAME **TAB 65** TOTAL YTD.SALES NEWLINE **TAB 20** ADDR1 NEWLINE **TAB 20** CITY_ST_ZIP DOUBLE-SPC

This example shows the tab with both column and row positioning to produce the same report:

LIST DEMO.CUST ID **TAB 20.1** NAME **TAB 20.2** ADDR1 **TAB 20.3** CITY_ST_ZIP **TAB 65.1** TOTAL YTD.SLS DOUBLE-SPC

TESTPRINT

PURPOSE To print a specified number of test patterns before beginning to print actual data records on a report.

FORMAT LIST **... TESTPRINT** ***number...***

SYNONYMS None

USAGE

When GENERAL is used to print labels, or to fill in pre-printed forms, it is useful to print several test pattern records at the beginning of a report in order to fine tune the paper alignment before valid data records begin printing. By specifying a TESTPRINT keyword, GENERAL will print the specified *number* of pattern records at the beginning of the report.

The test pattern records are derived from the first record found to print. All characters found in that record are replaced with asterisks, the specified number of patterns are printed, then the actual record is printed and the test pattern is discarded.

A special value for *number* of -1 indicates that a test row should print, and then GENERAL should prompt for another, or allow the report to begin. In order to allow each test row to print, GENERAL must close and re-open the printer. On systems using the print spooler, or with other automatic close logic, a page eject may occur which renders the test pattern useless. GENERAL can do nothing about this, so it is up to the site to provide printer definitions that will not eject the page when the printer is closed.

EXAMPLES

This example shows how to use the TESTPRINT keyword to print 3 test pattern rows of 3 records each:

```
LIST DEMO.CUST TAB 2.2 NAME TAB 2.3 ADDR1 TAB 2.4 ADDR2 TAB 2.5
CITY_STATE_ZIP(1,35) WIDTH 35 HEIGHT 6 ACROSS 3 NO-PAGE ON PRINTER
TESTPRINT 9
```

TITLE

PURPOSE To cause a title other than the default to be printed at the top of each page of output.

FORMAT LIST **... TITLE** ***"title-text"***

SYNONYMS None

USAGE

This keyword is used to define a report title other than the default. GENERAL will automatically produce the title:

LIST 'file-name'

The title is placed between the date and the page number on the first line of each page of the report. If a specific title is desired, the TITLE keyword may be used.

The title-text must be enclosed in quotes. It may optionally contain the special character strings "^L" or "|" to indicate a line split, thereby allowing multiple line titles. Text is split off from the line split and centered. Note that the ^L is a literal "^" and "L", not control-L.

EXAMPLES

The following example shows how to set up a two line title for a report:

LIST CUST.NAME ID NAME SLSP PHONE **TITLE "CUSTOMER PHONE LIST^LSORTED BY CUSTOMER ID"**

TOTAL

PURPOSE To cause a numeric field to be totaled, with the total value output at the end of the report, and at break points if specified by the BREAK keyword.

FORMAT LIST **... TOTAL** ***field-name...***

SYNONYMS TOT

USAGE

The TOTAL keyword is used to generate the total numeric value of a specified report column. The total so generated will be printed at the end of the report and also at each subtotal break point if the BREAK keyword is used.

The field-name specified must be defined for the file being LISTed. It also must be defined as a Numeric type field.

EXAMPLES

The following example will produce the total sales value of each line on the report:

LIST DEMO.CUST ID NAME SLSP **TOTAL YTD.SLS**

UPPER

See PROPER in this chapter.

VERT-TOTAL

PURPOSE To cause subtotals and totals to print in vertical (table) format.

FORMAT LIST **... VERT-TOTAL...**

SYNONYMS VT

USAGE

The VERT-TOTAL keyword is used to cause GENERAL to format totals and subtotals in a table fashion. Each item in the field-phrase, preceded by a BREAK, AVERAGE, COUNT, MAXIMUM, MINIMUM, PCT-TOTAL, or TOTAL will be displayed, along with the heading name of the field affected.

Although this requires more lines of output, it is sometimes preferred over regular totals, since the descriptions help the user's understanding of the report.

If GENERAL can't format the default footers under columns, the VERT-TOTAL format is automatically used.

EXAMPLES

The following example will produce the total sales value of each line on the report:

LIST DEMO.CUST ID NAME BREAK SLSP TOTAL YTD.SLS SORT SLSP **VERT-TOTAL**

WIDTH

PURPOSE To set the number of printed columns per page, if other than the default of 79 for VDT output and 131 for other output.

FORMAT LIST **... WIDTH #-*columns*...**

SYNONYMS WIDE

USAGE

This is a formatting keyword used to override the default settings used by GENERAL for line width for a report. The line length determines the point at which GENERAL will "wrap-around" the output to the next line on the page.

#-Columns must be an integer from 30 to 1000.

EXAMPLES

This example shows adjusting a VDT report to 131 columns:

LIST DEMO.CUST ID NAME DEMO.SLSP:NAME TAB 100 TOTAL YTD.SLS TOTAL YTD.COST **WIDTH 131**

YYYY

PURPOSE To convert the precision of output for a date field to a year, rather than a month, day, and year.

FORMAT LIST **... YYYY** ***field-name...***

SYNONYMS YY

USAGE

The YYYY date conversion keyword is used to convert a date field to display in year format. Years are always displayed as 4-digit values, such as "2001".

Date conversion can be useful when a break point by year is desired. See also the MONTH keyword.

EXAMPLES

The following example shows the generation of a date sorted report, with subtotals by year:

LIST DEMO.INVOICES CUSTOMER INVOICE DATE TOTAL AMOUNT SBREAK **YYYY DATE** SORT DATE

YYYYMM
YYMM

See MONTH in this chapter.

ADMINISTRATION

Configuration

Configuring GENERAL

GENERAL may be configured for site preferences in several ways. Many aspects of configuration don't need to be modified from their defaults, so configuration need not be a difficult task. The following pages describe in detail what each configuration option does.

Here is a brief description:

Display Formats is used to establish system-wide parameters, such as date and currency formatting, and the "exit program", which tells GENERAL what external Basic program to run upon exit.

Terminal Attributes establishes system-wide defaults for terminal display characteristics. Color control, for example, is defined here.

Printer Definitions are used to establish printer drivers, if modes other than those provided by Basic are desired.

Report Heading Defaults allow the establishing of standard "site-specific" heading characteristics for any report that does not have a custom header defined.

VDT Options establish terminal attributes (colors, etc.) for this specific VDT, overriding the system-wide defaults established by the Terminal Attributes option.

User Function Definition is used to maintain user-defined functions that can be used by field definitions in the dictionary.

User Replacement Definition is used to maintain user-defined run-time replacement values. Quite often, these values are maintained automatically by an external application, but may be defined here.

Display Formats

Language Code

Defaults to **ENG** for English.

Date Entry Convention

Defaults to **MDY** (month/day/year) for USA dates. Other formats supported are **DMY** and **YMD**. This value is used when parsing text dates entered by the user at run-time.

Date Display Format

Allows you to choose how dates are formatted by combining certain codes for month, day and year, as follows:

MM - Month number	**MMM** - Month name
DD - Day number	**DDD** - Day name
YY - Year, 2 digit	**YYYY** - Year, 4 digit

You can combine these codes in any way to customize the date display, including spaces and commas (or any printable character), **up to a maximum of sixteen characters in the entry field**.

Time Display Format

Allows you to choose how time is displayed, by combining codes for hour, minutes, and "12 hour time".

HH – Hour **MM** – Minutes **PM** – 12 hour time

Default Century

Allows you to specify the default century for date entry using 2-digit years.

Decimal Character

The decimal character to use for number displays.

Thousands Separator

The character to use for separating thousands in number displays.

Negative Indicator

The character(s) to use for displaying negative numbers when currency format is used, either the **"-"** for a trailing minus sign, or parentheses **"("** to enclose the number in parentheses.

Currency Symbol

The character to use to indicate currency, with an optional underscore to indicate space fills to separate the symbol from the number.

Right or Left Side

Specify whether the currency symbol appears on the Left or Right side, **L** or **R**.

Currency Precision

Decimal precision for currency amounts. Default to **2**.

Exit Program

The name of the program that GENERAL should RUN when exiting. If this is not defined, and GENERAL is accessed via the RUN command, then an exit will return to the operating system.

Terminal Attributes

Background Display

Enter the attribute to use when maintenance screens paint "background" text. Normally, this would be just the "Text Only" attribute. Note that GENERAL always uses an 'SB' Basic attribute for background.

The attributes supported are:

0	Text only
1	Print text in underline
2	Print text in reverse video
3	Print both underline and reverse

Data Display

This attribute is used when foreground data is displayed.

Data Entry

This attribute is used when a field is being entered.

Colors

The color attributes are used when each type of menu or prompt situation is encountered. Each color specification is a color "pair", indicating the color of the characters themselves and the color for the background of each character. The digits that specify the colors are:

1	Black
2	Blue
3	Green
4	Cyan
5	Red
6	Magenta
7	Yellow
8	White

Note that the terminal attributes can also be defined for individual terminals, if these system-wide attributes are inappropriate for any given VDT.

Printer Definitions

Printer definitions in GENERAL can be used to override and enhance the printing capabilities of a printer definition in Basic. Normally, printers in Basic provide a "standard" and "compressed" mode, and not much else. By defining a printer definition in GENERAL, more "modes" can be established for a given printer, and more than one definition can be established for any Basic printer.

When presenting a list of printers at report time, GENERAL looks in the printer definition table defined here. Then, for any printers that are available in Basic, but not defined here, an additional selection entry is provided.

Printer

Enter the name of the printer definition. The name can be up to 6 characters long, without spaces.

Press **F2-list printers** to choose the printer definition from a selection window, or **F10-exit** to return to the configuration menu.

Description

Enter a description of this printer. The description is used in selection lists.

Alias

Enter the name of the physical printer, as Basic knows it. Typically, this is LP or P0.

To choose the alias from a selection list, press **F2-list definitions**.

Width

Enter the default (standard) width in columns for this printer. This is the width assumed when the printer is first opened, and the initialization string is sent to it.

Length

Enter the default (standard) length, in rows, for this printer. This will typically be 66 lines (11 inches by 6 lines per inch).

Form Feed

Enter the desired method for GENERAL to control page ejects:

0 GENERAL doesn't issue form feeds before or after a report.
1 GENERAL issues a form feed before, but not after, each report.

2 GENERAL issues a form feed after, but not before each report.
3 GENERAL issues form feeds before and after each report.

Initialization

Enter a string to send to the printer before any printing takes place. The string can be entered as ASCII text and ASCII codes, or in hexadecimal codes.

To enter ASCII text, enter printable values as text, and non-printable (including spaces) values as ASCII decimal values in angle brackets. An ESCAPE, a letter "k", and a number 1, would be entered as **<27>k1**. Note that spaces are not significant, so **<27> k 1** would be the same. To enter an actual "space" character into the string, enter **<32>** for it.

To enter hexadecimal values, start strings with a dollar sign: **$1B45** would be the entry for an ESCAPE followed by an "E". Spaces can't be used in a hexadecimal entry.

Reset

Enter a string to send to the printer when a report is complete, and the last form-feed, if any, is sent to the printer.

Bold On

Enter a string to turn on **bold** printing.

Bold Off

Enter a string to turn off bold printing.

Underline On

Enter a string to turn on underline printing.

Underline Off

Enter a string to turn off underline printing.

Validation Prompt

Choose the definition filing option, when the maintenance is complete:

Yes to file the definition as displayed.
No to maintain the definition.
Delete to delete the definition, after validation.
Modes to maintain additional printer modes.

What are "Modes"?

Printers typically support many characteristics of output. Often, a printer that normally prints at 10 characters per inch (CPI), for about 80 columns on 8.5" paper, can also print at 12 CPI and at 17 CPI. Also, printers can print in varying quality modes, such as "draft" and "letter quality". In almost all cases, these modes can be turned on and off independently of each other, so that you can combine print characteristics together. GENERAL's print drivers let you create up to four modes under which to print (in addition to the "standard" mode defined by the main printer initialization string.

Name

Enter the name of the mode being defined. Mode names can be up to 10 characters long. Each mode which has enough columns is selectable for any given report, when the ON PRINTER specification is used in the report command. The following function keys are active:

F3-insert	Inserts a new mode line.
F4-delete	Deletes the current mode line.
F9/F10-exit	Pressing either key will exit mode maintenance.

Cols

Enter the number of columns per line this mode provides.

Rows

Enter the number of rows per page this mode provides.

Initialization

Enter the string GENERAL should send to the printer to turn on the mode. This string is sent is sent to the printer after the base initialization string and before any printing will take place. The format of this string is the same as that described in the initialization element, above.

Report Heading Defaults

Report Headings

Whenever a report is generated with the LIST command, a default page heading is created, consisting of a base heading and column heading. The base heading is made up of several information elements, placed at specific positions within a three-line base-heading template. By specifying the placement of these various elements, each site can customize the look of the each report's default heading.

Note that any given report can have a custom page heading, or the page heading can be suppressed.

Placement of text

Each element of text shown in the Report Heading Defaults screen can be turned on or off, and if turned on, can be placed on any of three lines, and positioned at the left, center, or right of the page.

The report title comes from the report itself, specified by the TITLE keyword, or defaulting to **LIST OF "*filename*"**.
The page number will have the word "Page" preceding it at report time.

Text 1 and Text 2 are site specified text elements that can be used for any desired purpose. One common purpose would be to place a company name in each report. If necessary, the text can be a run-time replacement, such as "[[@COMPANY_NAME@VDT]]".

VDT Options for the Current Work Station

Overriding Site Defaults

The configuration elements for Display Formats and Terminal Attributes establish system-wide defaults for how GENERAL behaves. In addition, each terminal can have its own display formats and attributes, overriding the default specified.

There are two elements on the system-wide Display Formats screen that are not available or maintainable on a terminal by terminal basis:

The **Date entry convention** is a system-wide parameter, so that the CONVERT-DATE keyword will work the same for any report that has a literal date reference to convert.

The **Exit Program** is also a system-wide parameter. Every terminal will exit to the same program. If conditional returns are required, then a separate program will have to be created to perform the required returns, and that program made the Exit Program.

User Function Definition

User Defined Functions

GENERAL's dictionary supports the use of user-defined functions that are provided in Business Basic. Function definitions are maintained here, and at run-time, GENERAL inserts them in the report as required.

Name

Enter the name of the function. Function names can be from 1 to 20 characters long, and may contain letters, digits, and underscores. If the function derives a string result, then it must end in a dollar sign.

To choose the function from a selection list, press the **F2-list** key.

To exit to the configuration menu, press the **F10-exit** key.

Description

Enter a description for this function. The description is used in function selection lists.

Function

Enter the arguments and Basic code for the function. This code follows the same conventions as defined in single-line DEF FNx() functions in Basic. Argument variables should not start with TEMP or GEN, as these may conflict with names used by GENERAL itself.

For more detail on user functions, consult a Business Basic manual.

Verification Prompt

At the verification prompt, you may choose:

Yes to file the definition as displayed.
No to maintain the definition.

Delete to delete the function. Any dictionary expression or run-time calculation that references the function will fail.

To abort, without saving the definition, press the **F10-exit** key.

User Replacement Definition

Replacements

Run-time replacements are a key feature of GENERAL that allow the dictionary or report definitions to be parameterized to operate under a variety of conditions. The most common use of replacements is to allow the same dictionary to support the parameters of different companies. For example, one company might have a customer code length of 6, while another might be 8. An expression for the customer code could be defined as:

@PF1(1,[[@CUSTLEN@VDT]])

So that the run-time replacement value defined for @CUSTLEN@VDT would be used for each report.

In the above example, there are several things to note.

- The replacement is delimited by double square brackets. Former versions of GENERAL only required single square brackets. While GENERAL parses the command, the double brackets are detected, and the text inside is evaluated.

- If the replacement text starts with a "@" character, then GENERAL looks in the run-time replacement table for a value. Otherwise, GENERAL issues the replacement text as a prompt to the user, and accepts the user's input as the replacement value.

- Note that another form of replacement is supported to force the entry to be text, numeric, or date data.

- If the keyword @VDT is included in the text, then GENERAL substitutes the terminal's alias in the replacement text before searching the run-time replacement table. This makes it possible to control replacement values on a terminal by terminal basis.

Normally, an external program should maintain run-time replacements, since the parameters that are defined will come from the external application. See the Appendix, Integration Utilities, for more information about how an external application can maintain these.

Name

Enter the name of the run-time replacement. The name can contain from 1 to 30 characters. If it contains the word @VDT, GENERAL will substitute the current terminal alias.

To choose the replacement from a selection list, press the **F2-list** key. To return to the configuration menu, press the **F10-exit** key.

Description

Enter a description for this replacement, if desired. The description is used in selection lists.

Value

Enter the value to use as a replacement when this run-time replacement is referenced by a report.

Import/Export Secondary Dictionary

These options are used to move report and dictionary definitions between systems. A source system can export definitions to a file, and a receiving system can import them from the file.

Secondary dictionary

This is the pathname of the file that stores the exported definitions. When exporting, the file can be created if it doesn't exist.

Export Options

Exporting offers the following options: Prompt, Run, Dictionary, and Clear.

- Prompt and Run provide a selection list of each type of report to export. Selecting one will export that file dictionary to the secondary dictionary file.

- Dictionary provides a selection list of file definitions. Selecting one will export that file dictionary to the secondary dictionary file.
- Clear will remove all records from the secondary dictionary, after a verification prompt.

After all desired report and dictionary definitions have been added to the secondary dictionary, the file can be transferred to another system and imported.

Import Options

The single import option is to control overwriting of dictionary field definitions. If the Overwrite fields prompt is set to Y, then all field names in a given secondary dictionary are imported. If it is set to N, then only new field names are added.

Maintain Report Menus

User menus are defined with this option. There can be any number of menus, each of which can contain up to 36 links to reports (prompt or run) and other menus. There is one default top-level menu name, called "TOP"; the administrator can name all other menus. The TOP menu is available from General's main menu. Any menu, including TOP, can be specified as a user's menu, providing administrators a way of establishing "run-only" users who are only provided a list of valid reports to run. Users who have an associated menu will not have access to General's standard menu.

Menu ID

The menu ID identifies this menu definition. The special ID “TOP” is used to define a menu that can be accessed from General’s main menu. Other names are user-defined.

Title

This is the title of the menu, and appears at the top of the menu window when displayed for a user.

Type

Each menu can have up to 36 items. Each item is given a type:

1 A report defined in PROMPT mode
2 A report defined in RUN mode
3 Another menu

Name and Description

For each type, you enter a name, or press F2 to select from a list of available names. The description of the report or menu is added automatically.

USER MAINTENANCE

What are "Users" in GENERAL?

GENERAL will usually require that a user *login* in order to create and execute a report (see Accessing GENERAL, in the Introduction). The login procedure verifies that the user code is valid, and assigns an access level to the user, so that GENERAL can enforce file security.

Each user code is established by the USER command, which is available from GENERAL's main menu to any user with an access level of 9. Such a user is known as an *administrative level user*. Each user code can also be assigned an associated password, which will also be required by the login procedure. Note that passwords are optional, and may be required by some users and not by others. It would not be uncommon to establish one or more user codes, with low access levels and no passwords, and a single administrative user code, with a password.

System Logins

GENERAL does not utilize operating system user names as user codes. It will test a Basic global string table (STBL() in BBx, for example) called "*GENEXT_USER" for a previously established login name, so that an external Basic menu system can pre-assign the GENERAL user code. That user code must have been established in the GENERAL user code list, so that GENERAL can determine the access level and whether or not to prompt for a password.

Therefore, if the user name is established as a GENERAL user code, without a password, then that code may be placed in the global string table noted above, and the login screen will be skipped.

Code

Enter the user code, which may be any character string, from 1 to 20 characters long, without spaces. If the code entered isn't currently on file, GENERAL verifies that the code is new, and provides an option to copy the new user definition from an existing one.

The **F2-list** key may be used to choose the user from a selection window of defined users.

Press the **F10-exit** key to return to the main menu.

Name

Enter the user name or description associated with this user code. This is used when presenting the user selection window.

Access Level

The access level is a number, from 0 to 9. The higher the number, the more files this user code is allowed to access. When processing a report, GENERAL checks the access level assigned to each file required by the report. The user's level must be at least as high as each file's level.

A user with an access level of 9 is considered an administrative user. An administrative user has access to any file, and also to the CONFIG, DICTIONARY, and USER commands.

First menu

The top level menu that is displayed when the user logs in. User menus are defined in the Configuration section. If no menu is defined, then the standard General main menu is used.

Password

The password is an optional entry for a user code. If this field is blank, then there will be no password required when this user code is used in the GENERAL login screen.

The password is encrypted when the user record is filed at the verification prompt. Once encrypted, GENERAL no longer knows what the password was. For that reason, if a user has a password, and the record is maintained, then the password field will contain a single question mark. **For a user to keep the same password, it is essential that <Enter> is not pressed here.** If <Enter> is pressed, then GENERAL will re-encrypt the new data in the password field (even if it is the "?" shown to indicate that a password is present). Instead, press the **F10-exit** key, and the original encrypted password information will be retained.

Verification Prompt

Yes to file the record as displayed.
No to maintain the record.
Delete to delete the user code, after verification.

Press **F10-exit** to abort filing of the record. Any changes made will not be updated on the user code.

APPENDIX

GENERAL Version 5 Enhancement List

The following information is provided as a list of features added in Version 5, for the benefit of those users upgrading from Version 4.*x* of General. Version 6, of course, features all of these enhancements, plus a GUI interface capability, plus the features documented in the next section.

User Defined Menus

General Version 6 supports a menu structure that can be defined by a site administrator. There can be any number of menus, each of which can contain up to 36 links to reports (prompt or run) and other menus. There is one default top-level menu name, called "TOP"; the administrator can name all other menus. The TOP menu is available from General's main menu. Any menu, including TOP, can be specified as a user's menu, providing administrators a way of establishing "run-only" users who are only provided a list of valid reports to run. Users who have an associated menu will not have access to General's standard menu.

To define a menu, use the Config option from General's main menu, selecting the Maintain Report Menus option (on page 2 of the selection list.) To associate a menu name with a user, use User Maintenance from General's main menu, and enter the desired menu name into the user's record.

Report Security

General has always had file security based on user access levels, and would prevent a user from producing a report from a file with a higher access level than the user. In Version 6, both Prompt and Run reports now have Run-only and Modify access levels. This allows the administrator to grant "run or modify", "run only", or "no" access to a given report based on the user's access level.

Expanded Calculation Space

New expanded windows are available in several calculation entry points. These include @CALC entry in Prompt mode, @CALC entry in assist mode (List command entry), and Header/Footer data calculations in Prompt mode. The expanded windows allow up to 900 characters to be entered. They are opened with a function key identified on the screen.

Expanded Begin with/End with

Like the above calculations, the Begin with and End with entries in Prompt mode also offer expanded space, up to 900 characters.

Begin/End Expressions

On occasion, a file's key may contain non-printable or non-human readable data. This has become more common as software developers make changes to accommodate the year 2000 in dates. To respond to the need to place odd characters into report key ranges, General Version 6 supports the use of Business Basic expressions in the Begin and End keywords. To get General to interpret an entry as an expression, use single quotes around the value. For example:

BEGIN '"01"+HTA(BIN(JUL(12,1,2001),3))'

Run-time prompts can be embedded inside the single quoted expression.

Heading Row in Delimited Exports

A new style of export, "delimited-h" is now available, which adds an initial line to the export file containing the column headings in the same delimited structure as the data. Many productivity tools, such as Microsoft Word or Excel, work better when provided this row of information. In Prompt mode, this option is option 6 (replacing the discontinued Gateway for Windows option.) In List commands, use the DELIMITED-H keyword.

Prompt Mode Delimiter Specification

Older versions of General always assumed a comma-delimited format when a delimited export format was selected. Version 6 allows the delimiter to be changed. You can also specify a non-printable character using General's tilde notation. For example, a "~009" would indicate a tab-delimited file.

Prompt Mode No-Detail Exports

Version 4.1 offered the capability of exporting just totals by combining a NO-DETAIL keyword with the export format keyword(s). However, Prompt mode entry did not provide access to the aggregate value flags (total, average, etc.) when specifying fields, making it impossible to produce a no-detail export from Prompt Mode. In Version 6, this restriction has been removed, so the aggregate flags are accessible in export definitions. Note that for exports *with* detail, the aggregate flags are ignored.

Internal Session Number Tracking

In prior releases, General has used the FID(0) or the operating system task number to create work files. On certain operating systems or combinations of workstations and servers, these values have caused problems with non-unique names or invalid work file names. With Version 6, session IDs are internally assigned and used for unique work file names.

Report Logging

Some General users launch reports from scripts or batch files. A new feature in Version 6 allows the script to specify a log file. Simply specify an environment variable called GEN_LOGFILE, and any errors, plus the report's recap page, will be printed to the file. Care should be exercised when choosing a log filename to avoid damaging programs or data. If no path information is specified (indicated by / or \ in the name), then General will place the file in its own TMP directory. To have General create a log file in its TMP directory, named as *session-ID*.log, set GEN_LOGFILE to "*create".

If you wish to turn on logging when launching General from the Business Basic environment, define a global string (STBL or GBL) "*GEN_LOGFILE" to the filename desired or "*create".

As a debugging feature, a *session-ID*.log file will be created in General's TMP directory when the user presses Escape-Escape-9 (available in BBx only) from General's menu and many other entry points. The session number is displayed in the About General screen available from General's help screens.

Larger Report Designs

Under BBx3, BBx4, and ProvideX, programs are limited to 64K, and complex reports could be designed that exceeded that limit. With the release of BASIS International's PRO/5 and Visual PRO/5 languages, which support program sizes up to 1MB, a barrier to very large report designs was removed. However, with even larger report designs now possible, a new barrier was encountered in the statement number ranges offered by General for certain elements of a report. For example, a limit of 2,000 lines of code could be generated by the field phrases of a report.

Version 6 increases the available number of code lines in many report sections. In addition, Version 6 will issue an error message if a range is exceeded, rather than continuing with unpredictable results. Below is a table indicating the old and new limits:

Code Section	Version 4 limit	Version 6 limit
Field Phrase	2,000	10,000
Select Phrase	1,000	3,000
Cross References	1,000	2,000
Header/Footer Calcs	2,000	4,000

In addition, tabulate reports using the “summary” style now support 20 column calculations, up from 6 in prior versions.

About General Screen

A new screen is available from any General Help (F1-help) screen, which shows version, serial number, session ID, and additional information.

Gateway for Windows Support Discontinued

General 4.1 incorporated a special export format that drove Thoroughbred Software's Gateway for Windows product. This product was never used in production, and the provision for it only generated confusion among users who did not have the Gateway product installed on their system. We have removed support for the Gateway export format.

Excel Export for Windows Environments

For BBj, Visual PRO/5 environments and Windows ProvideX or WindX, there is a new Excel export format that uses DDE to create an Excel worksheet. Simply choose the new Excel export format (Report type 7 in Prompt mode, or EXPORT EXCEL in a List command). If your environment supports it, and if the "excel_path=" line in gen6parm.fil correctly defines the location of excel.exe, then the export is automatic.

Note that for BBj to support DDE, you must add the following alias line to the BBj configuration file (sometimes called the config.bbx file):

```
ALIAS JDDE com.basis.plugin.DDEOpenPlugin
```

When exporting to Excel, you can specify a file name for the workbook. In that workbook, Sheet1 will always be updated, and the workbook will be saved. If you don't want General to save a workbook, then specify VDT output, and an unnamed, unsaved workbook will be created.

When working in a WindX session, naming a workbook file is problematic. File paths must be identical between the host system and the client system. For example, you can specify a path of "/tmp/abc.xls" during the report execution. However, the path "\tmp" must also exist on the client workstation, as that name is passed to Excel unchanged except for conversion of "/" delimiters to "\".

In addition, General can be used to drive sdOffice™, a platform- and language-independent product that automates Microsoft Office applications, from any platform. General will first attempt to use sdOffice when exporting to Excel. If it can do so, it will perform the export and add column formatting to match the field definitions from the dictionary. If it can't use sdOffice, and if it is running in a Windows environment (Visual PRO/5, ProvideX on Windows, or WindX), then it will attempt to use DDE to export the column headings and data.

Run-time Debugging

General has always offered a way to turn on "debug mode" by setting a global string *GENDEBUG before executing General. Debug mode causes General to escape to console mode when exiting the menu and also when a report has just been compiled and is ready to run. To enable easier support, debug mode can now be turned on and off from the menu by pressing Escape-Escape-0 in BBx, or with the command "debug on" and "debug off" executed from a LIST command window (available only for level 9 users).

External User-defined Functions

General now supports multi-line user defined functions through the automatic import of two files, “userdefn.txt” and “userinit.txt”. Code in userinit.txt is executed once just before any report executes. This can be used to initialize variables, most likely TEMPxxx variables, for use by reports or functions. Code in userdefn.txt is intended to contain user-defined functions in single- or multi-line format. This code is inserted into every report and can be used by any dictionary expression, calculation expression, or begin/end expression.

The format of both files is plain text. There can be no line numbers. All line targets must be labels or symbolic labels. Any variables should start with the letters TEMP, and such variables may be referenced in dictionary or report calculation expressions as @TEMP*name*.

ProvideX genptrs.pvx File Enhanced

The file genptrs.pvx, which stores printer names and maximum columns, can now also include a printer description as a third line element. This information will be displayed in printer selection windows that appear during General’s operations, in particular when a report is define with output as ON PRINTER.

Other Enhancements

When in debug mode in the BBx port, the backspace key will now work.

Error messages during reports now show the key of the record that caused the error.

GENERAL Version 6 Enhancement List

In addition to the very obvious enhancement of adding a graphical, Windows-base client for General, there are many features that have been added to the traditional character mode as well.

BBj Compatibility

This new version of BBx from Basis International is gaining in popularity. Internal changes in the language made prior versions of General incompatible with BBj. With Version 6.0, General is now compatible and may be operated in BBj environments.

Report ID Support in Header Maintenance

The Config option in General provides a way to layout default report heading formats. A new field has been added to allow placement of the report name being run in the default heading. This name is taken from the Prompt or Run report being executed. It is, of course, blank when a List command is executed directly or from command history.

Run Date/Time and User in Prompt Reports

Like Run reports, Prompt reports now log the last date and user when the report is executed.

Enhanced Report Lookups

When pressing F2-list in Prompt and Run report name entry, the list has been enhanced to show additional columns: report file and last run date, and also to provide sorting by report ID, title, report file, and last run date. This is especially useful in installations with hundreds or thousands of reports..

Selected Page Printing in Preview Mode

When printing from preview mode, former versions required printing of the entire report. This new feature allows printing of just desired pages.

Integration with Applications

GENERAL is designed to work with existing data files, which are maintained by an existing application. GENERAL works with the other application via the data dictionary, which is described in detail in the Dictionary chapter of this manual. There are several external dictionary formats that GENERAL can use, so in many cases, a GENERAL dictionary will not need to be maintained. Several features of GENERAL also make it possible to develop a more seamless environment between GENERAL and the application with which it runs.

Executing GENERAL from the application

To start GENERAL, the other application must use the RUN or CALL command. The startup program is "gen6":

RUN "gen6" will overlay to GENERAL. This conserves the most memory.

> **CALL "gen6","*command*"** will call GENERAL, preserving the state of the calling application. This method does not conserve memory, so the calling application should have plenty to spare. The *command* is a required parameter, and may be either a complete LIST command, a save-command name, a PROMPT report name, or null ("").

In order to provide fast access to field selection lists, and to generally improve run-time performance of GENERAL, all elements of dictionaries are loaded into memory when accessed. This means, for instance, that when a file is chosen for a PROMPT report, all the field names and their associated descriptions, headers, and expressions are loaded into memory. This works very well except in cases where a large dictionary is loaded into a small memory workspace, which often results in an error 31 (out of workspace memory).

On BBx, GENERAL uses the BBx executables pro5cpl and pro5lst (on BBx4, bbx4cpl and bbx4lst). These executables must be accessible to GENERAL from the same path used to launch bbx. For example, if your application starts BBx as “/usr/lib/pro5/pro5”, GENERAL will expect to find “/usr/lib/pro5/pro5lst”.

Returning to the application

If GENERAL is started by the CALL verb, there is no need to worry about getting back to the CALLing application; if GENERAL is RUN, however, then it must be told how to get back to the application that started it. An "exit program" name is defined by the CONFIG command, under the Display Formats option. When this name is filled out with a valid program name, then GENERAL will RUN that program when exited.

Automatic Report Execution

GENERAL can be told to automatically execute a single command, and exit. The command can be any command that can be typed from the LIST command prompt, including complete LIST commands, saved command names, or PROMPT report definition names. Once the command is complete, GENERAL immediately exits.

There are two methods to accomplish this, depending on whether GENERAL is started with the RUN verb or the CALL verb.

When GENERAL is executed by a RUN verb, assign the variable O$ to the word "EXECUTE", the variable X$ to the command desired, and the variable Y$ to the exit program name before RUNing gen6. Optionally, you can preset two global string tables: *GEN_GO to the command, and *GEN_GOEXIT to the exit program name.

When GENERAL is executed by a CALL verb, then the parameter that it is called with becomes the automatically executed command.

When commands are automatically executed, GENERAL will still prompt for a user login code unless it has been told the user code already, in the global string table *GENEXT_USER. If that element is set to a valid user login code, and that user code doesn't have an associated password, then the user login is bypassed. A passworded-login cannot be bypassed.

Additionally, if the GENERAL logo should be suppressed, when the login is also bypassed, then the global string table *GEN_QUIET may be set to "Y". GENERAL still runs in its own window, so the screen will be cleared on entry into GENERAL, but the logo will not be displayed.

Running GENERAL reports in background

On Unix systems, GENERAL reports can be executed in background from the command line by setting up two environment variables, then starting Business Basic, in background, running the gen6 program. The Unix cron command makes it possible to execute GENERAL reports at specified times and/or on specified days, as well.

In the Unix shell, establish and export these variables:

```
GEN_GO=command-name
        GEN_USER=user-login
        export GEN_GO GEN_USER
```

The *command-name* should be the name of a saved report or a PROMPT name, or it can contain a complete LIST command, if desired. The *user-login* should be set to an unpassworded -login code that has an access level high enough to run the report desired.

Then to start the report in background, using a BBx command line as an example, enter:

```
nohup /usr/basis/pro5/pro5 -m512 -c/usr/basis/pro5/config.bbx gen6 >/dev/null &
```

This command starts a BBx task in background, immune to hang-ups or quits (the initializing user can exit if nohup is used). The task will start gen6, which upon detecting that it is being run in background, will check the above environment variables to determine what to execute. Output that would normally go to the screen is redirected to the null device (in effect, it is thrown away).

Such commands should not require any user prompting and should output either to a file or a specific printer. ON PRINTER, for instance, wouldn't work, because it would cause GENERAL to prompt for a printer.

General 6 adds support for report logging, which can be helpful in background report execution. Set the environment variable GEN_LOGFILE to a path, and errors and the recap page will be printed to that log file.

Filix integration

The BBx version of GENERAL Version 6 can produce reports from Filix databases without a GENERAL dictionary definition for the files. Support for Filix files is determined when GENERAL is started, and requires that the Filix home directory and the directories where any Filix files are located be included in the BBx directory search prefix. Optionally, full pathnames may be used when specifying the Filix file, but the Filix home directory must still be in the prefix list, and any Filix files outside of the prefix path listing will not be identified in file name searches.

When a Filix file is LISTed, GENERAL performs the following conversions:

Money fields are displayed as numerics with currency formatting.

- Keyword and Formatted text fields are displayed as simple text fields. Note that keyword fields are not implicitly broken up into individual keyword segments, but that when the ALTSORT keyword is used with a keyword-based secondary key, the records are sorted by individual keywords.

- Long text fields (memos) are converted to text fields with a "lines" parameter matching the Filix form design for the field. There is, currently, no "variable lines" capability, so some memos will be truncated, others padded.

- If calc-only fields are included in the report, GENERAL performs the appropriate CALLs to calculate the values, but the report will run considerably slower than a similar report without calc-only values.

Note: Do **NOT** set up a GENERAL dictionary for a Filix file. Only those fields defined in GENERAL will be available, and if the Filix file design is updated, the physical layout of the data will probably change, making any field specifications setup in the GENERAL dictionary invalid.

Basis Data Dictionary Integration

The BBx version of GENERAL Version 6 supports the Basis data dictionary, provided that the environment has been initialized before starting GENERAL. Specifically, GENERAL looks for the extended utility set global, "BBEXT", and if found, will use the extended utility program "_ddopen.utl" to determine where the dictionary files arelocated. Once found, the dictionary is read implicitly whenever a file is used that is not present in the GENERAL dictionary or is a Filix file. The Basis dictionary will also be searched when file listings are requested.

The Basis dictionary doesn't provide for calculations, so any calculations must generally be performed at run-time with the @CALC keyword. It is possible, however, to provide an "incremental" dictionary in the General dictionary, by setting up a dictionary named *+file*, where the *file* value is the name of the file in the Basis dictionary. For example, "+CUSTOMERS" would define an incremental extension dictionary to the file CUSTOMERS defined in the Basis dictionary. Use this feature to add calculations and link specifications for use in General reports.

GENERAL will treat text and numeric fields properly based on their field types. Date values are assumed when a field is numeric and the "expand" element in the Basis dictionary "expand" definition starts with one or more site-specified text values, such as "DATE(", or "@(BBEXT)_undat". These values are defined in the parameter file gen6parm.fil, with lines beginning with **taosdate=**. By default, GENERAL provides "taosdate=date(" in that file, so that expand elements such as DATE(%) are interpreted as dates.

In addition, if dates are stored as Julian numbers, but are defined with a numeric offset to reduce the size of the number, that offset value may be specified so that GENERAL will add a specified number of days to the value before considering it a true Julian number. The parameter **taosdateofs=** can be used to specify the offset. For example, **taosdateofs=2000000** would add 2,000,000 to numbers read from the file, like this: X$=DATE(*date-number*+2000000).

Note: Do **NOT** set up a GENERAL dictionary using the same name as that in the Basis dictionary, unless you are willing to define all desired fields and maintain it for future changes in the file. Once the GENERAL dictionary is defined, reports will only look only at that.

Adding run-time replacement values

The GENERAL dictionary supports the use of run-time replacements to help parameterize data elements, such as company codes, output masks, and field lengths. The dictionary can reference such information with the run-time replacement operators: [[@*name*]], where *name* is the name of a parameter, which can optionally contain the word @VDT to make the name terminal-specific. The values associated with a run-time parameter must be defined before the parameter is referenced in a report.

Runtime replacements can be defined manually using the Config command, or can be defined by an external program prior to executing General. They can be defined simply as STBL or GBL values, or can be stored in the dictionary file via an integration program called "gen6-frp".

Using STBL or GBL to define a variable:

To define a replacement as an STBL or GBL (GBL is the function used in ProvideX), simply define the value based on the name after the @. For example, if the variable is referenced in an expression as **[[@FIRMID]]**, then you can define its value as **X$=STBL("FIRMID","01")**. A common practice for defining variables by terminal when they were stored in the dictionary was to add @VDT to the name. In this case, the reference would be **[[@FIRMID@VDT]]**, and the replacement could be defined as **X$=STBL("FIRMID"+FID(0),"01")**.

Using the dictionary to store the variable:

To use the traditional, dictionary-based variable definition, this is the syntax for "gen6-frp":

CALL "gen6-frp", PARAM$, "R", ACTION, NAME$, DESC$, VALUE$, ERRMSG$

For each run-time parameter, the gen6-FRP program should be called. The values and functions of the arguments are as follows:

PARAM$	This value contains GENERAL parameter information. It should be set to null ("") for the first CALL, then the value passed on subsequent CALLs.
"R"	This is a flag that tells GENERAL to maintain replacement values.
ACTION	This value should be 1 to add or update the value for the parameter, or 2 to delete the parameter.
NAME$	This is the name of the parameter, without any @ references. It can be set to a simple name, such as "FIRMID", or to a terminal-specific name, such as "FIRMID"+FID(0).
DESC$	This is just a text description of the value. The CONFIG command will use this when displaying lists of run-time replacement values.
VALUE$	This variable contains the replacement value to use.
ERRMSG$	Contains a message if there was a problem.

An example program fragment follows:

```
0010  PARAM$=""
0020  NAME$="FIRMID"+FID(0), DESC$="COMPANY ID", VALUE$="01"
0030  CALL "gen6-FRP",PARAM$,"R",1,NAME$,DESC$,VALUE$,ERRMSG$; IF
      ERRMSG$>"" GOTO DONE:
0040  NAME$="FIRMNAME"+FID(0), DESC$="COMPANY NAME",
      VALUE$="SDSI"
0050  CALL "gen6-FRP",PARAM$,"R",1,NAME$,DESC$,VALUE$,ERRMSG$; IF
      ERRMSG$>"" GOTO DONE:
```

When GENERAL is exited, a program fragment to remove the same parameters could be:

```
0010  PARAM$=""
0020  CALL "gen6-FRP",PARAM$,"R",2,"FIRMID"+FID(0),"","",ERRMSG$
0030  CALL "gen6-FRP",PARAM$,"R",2,"FIRMNAME"+FID(0),"","",ERRMSG$
```

Adding file- and field-definitions from external programs

The GENERAL dictionary is stored in packed format, so it is not possible to simply write new records to the dictionary as field definitions. In order to allow an application developer to add or update fields to a GENERAL dictionary, a utility has been provided.

There are three steps to updating a dictionary:

First, the dictionary must be opened:

CALL "gen6-upd.utl","OPEN", FILENAME$, DESC$, DISKFL$, ACL$, "", PARAM$, ERRMSG$

"OPEN"	is an action flag.
FILENAME$	is the name of the file to be maintained.
DESC$	if not null (""), will update the file's description in the dictionary.
DISKFL$	if not null, will update the file's disk file name.
ACL$	if not null, will update the file's access level (0-9).
PARAM$	holds information about the dictionary. It should be null ("") before the CALL, and not manipulated subsequently.
ERRMSG$	holds an error message, if an error occurred.

Second, for each field to be added or modified perform the following:

CALL "gen6-upd.utl","FIELD", FIELDNAME$, DESC$, TYPECD$, HEADING$, EXPRESSION$, PARAM$, ERRMSG$

"FIELD"	is an action flag.
FIELDNAME$	is the name of the field to add, update, or delete. It should be a valid field name, according to GENERAL's field naming rules (see the Dictionary chapter).
DESC$	is the field's description. If this is set to the literal text, "<DELETE>", then the field is deleted from the dictionary.
TYPECD$	is the field's type codes.
HEADING$	is the field's column heading.
EXPRESSION$	is the field's expression.
PARAM$	is the parameter variable set by GENERAL in the OPEN call, above.
ERRMSG$	holds an error message, if an error occurred.

Information about the type codes, column heading, and expression can be found in the Dictionary chapter of the manual. No validation is performed by this utility.

Third, once all the fields have been updated, the dictionary should be closed. This will write the changes made to disk.

CALL "gen6-upd.utl","CLOSE","","","","","","",PARAM$,ERRMSG$

"CLOSE"	is an action flag.
PARAM$	is the parameter variable maintained in prior CALLs.
ERRMSG$	holds an error message, if an error occurred.

Sample File Layouts

DEMO.CUST - Customer master file (gensmpl1)

Field	Key	Description
1	0	Customer ID
2	1	Name
3		Address1 (1,30) Address2 (31,30)
4	2, 3.2	City
5	3.1	State
6		Zip code
7		Telephone (10 digit string)
8		Date last sale (1,8) YYYYMMDD Date last pmt (7,8) YYYYMMDD
9		Class code (1,2) Salesperson code (3,3) Terms code (6,2)
10		Credit limit
11		Year to date sales
12		Month to date sales
13		Year to date cost
14		Month to date cost

DEMO.SLSP - Salesperson master file (gensmpl2)

Field	Key	Description
1	0	Salesperson ID
2		Name
3		Year to date sales
4		Month to date sales
5		Year to date cost
6		Month to date cost

DEMO.INVOICES - Invoices by customer (gensmpl3)
DEMO.INVOICE.HIST - Invoice history by customer (gensmpl7)

Accessed by READRECORD only.
DEMO.INVOICE.ALL demonstrates chaining the two files.

Field	Key	Description
1	0.1 0.2, 2.2 1 2.1	Customer (1,5) Invoice number (6,5) Date (11,8) Julian integer Salesperson (19,3) Amount (22,6) binary integer

DEMO.PAYMENTS - Invoice payments by customer (gensmpl4)
Accessed by READRECORD only.

Field	Key	Description
1	0.1 0.2 0.3	Customer (1,5) Invoice number (6,5) Sequence (11,2) Date (13,8) Julian integer Amount (21,6) binary integer

DEMO.CUST_NAME - Customer name sort file (gensmpl5)

Sort file for customers.

Field	Key	Description
Key	0.1 0.2	Customer name (1,20) Customer ID (21,5)

DEMO.SLSP.INV - Invoices by Salesperson sort file (gensmpl6)

Sort file for invoices.

Field	Key	Description
Key	0.1 0.2 0.3	Salesperson Number (1,3) Customer ID (4,5) Invoice Number (9,5)

Index