

The new face of COBOLSM

ACUCORP[®]

Release Overview for Product Suite Version 5.2.2

Acucorp is pleased to release Version 5.2.2 of selected products in our *extend*TM 5 family of technologies. Version 5.2.2 is designed specifically for sites using the HP e3000 environment.

Included in Version 5.2.2 are HP e3000-specific versions of the ACUCOBOLTM-GT Development System and Acucorp's Thin Client technologies, including AcuLaunchTM.

HP e3000 sites may also license our AcuBenchTM integrated development environment, and the ACUCOBOL-GT Development System, for use on Windows[®] systems.

All HP-specific features are documented in the supplement titled *Compatibility with HP COBOL*. This supplement is provided in both printed and on-line format (Hpsup.hlp). Chapter 1 of the supplement provides detailed installation procedures and information on getting started with ACUCOBOL-GT in the MPE/iX and POSIX environments. Chapter 2 explains how ACUCOBOL-GT treats HP COBOL extensions to achieve compatibility, our implementation of the HP COBOL preprocessor, and our support for HP e3000 intrinsics, including the interface to IMAGE and VPLUS.

ACUCOBOL, AcuLaunch, and *extend* are trademarks of Acucorp, Inc. Acucorp is a registered trademark of Acucorp, Inc. "The new face of COBOL" is a service mark of Acucorp, Inc.

Microsoft, Windows, and ActiveX are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Netscape and the Netscape N and Ship's Wheel logos are registered trademarks of Netscape Communications Corporation in the United States and other countries. UNIX is a registered trademark of The Open Group in the United States and other countries. Other brand and product names are trademarks or registered trademarks of their respective holders.

© Copyright Acucorp, Inc. 2002. ALL RIGHTS RESERVED.

R-01-RO-020905-Version 5.2.2

The following table will help you quickly locate documentation on several key topics:

	<u>HP e3000 products</u>	<u>Windows products</u>
Installation and setup	<i>Compatibility with HP COBOL</i> supplement	<i>Getting Started</i> booklet
HP extensions	<i>Compatibility with HP COBOL</i> supplement	<i>Compatibility with HP COBOL</i> supplement
ACUCOBOL-GT	On-line ACUCOBOL-GT manuals, plus this Release Overview	On-line ACUCOBOL-GT manuals, plus this Release Overview
Thin Client (server)	On-line Thin Client book: <i>A Guide to Using Acucorp's Thin Client Technology</i>	N/A
Thin Client (clients)	N/A	On-line Thin Client book: <i>A Guide to Using Acucorp's Thin Client Technology</i>
AcuBench	N/A	On-line AcuBench book

Version 5.2.2 incorporates enhancements from Version 5.2.1 as well as customizations for the HP e3000 user. The remainder of this overview describes enhancements brought forward from Version 5.2.1 that are not documented elsewhere.

ACUCOBOL-GT Compiler

At Version 5.2.1, the ACUCOBOL-GT development system includes several new enhancements. For your convenience, the information presented here is also available in the Release files for the product.

BITMAP Control Property, TRANSPARENT-COLOR

The BITMAP control has been enhanced with a new property, TRANSPARENT-COLOR, which supports transparent regions in a bitmap; instead of a color, the area “underneath” the bitmap is displayed. This is done by reserving one of the colors in the bitmap as the “transparent” color. Any points in the bitmap that contain this color will not be displayed when the bitmap is drawn, allowing the background to show through. This feature can be useful if you want to integrate a logo onto a screen in a way that accommodates the user’s choice of desktop colors.

To designate a transparent region, choose one color in the bitmap to be the “transparent” color. Then, set the TRANSPARENT-COLOR property to the RGB color value of this color. The RGB color value is computed by the following formula:

$$(red * 65536) + (green * 256) + blue$$

where *red*, *green*, and *blue* are values between 0 and 255. You can use a hexadecimal numeric literal to express this value because each component of the color occupies one byte. For example, the following literal expresses an orange color with red at 255, green at 128, and blue at 0:

```
78 orange value x#FF8000.
```

The first byte (x“FF”) is the red value, the second byte (x“80”) is the green value, and the last byte (x“00”) is the blue value.

If you set TRANSPARENT-COLOR to “-1” (the default), the transparency property is turned off and all colors are displayed normally.

Note: In order to avoid flashing, the runtime will not update the region “under” a transparent bitmap once it is in place. One effect of this is that if you animate a bitmap with transparent regions, you must ensure that those regions are the same in each frame of the animation, or the image will blur.

ENTRY Verb in Area A

An ENTRY statement can now begin in Area A. Therefore, the compiler no longer warns you if an ENTRY statement begins there. This change was made to improve compatibility with other compilers that support the ENTRY statement.

Library Routines

Version 5.2.1 of ACUCOBOL-GT includes the following new and enhanced library routines.

C\$LIST-DIRECTORY Routine

This routine lists the contents of a selected directory. It has been enhanced with a new argument, LISTDIR-FILE-INFORMATION, which can be included with the LISTDIR-NEXT operation to receive information about the returned file name. This optional group item is defined in “acucobol.def” and must have the following structure:

```
01  LISTDIR-FILE-INFORMATION.
    03  LISTDIR-FILE-TYPE          PIC X.
    03  LISTDIR-FILE-CREATION-TIME.
        05  LDFC-YEAR             PIC XX COMP-X.
        05  LDFC-MONTH           PIC X COMP-X.
```

```

05  LDFC-DAY                PIC X COMP-X.
05  LDFC-HOUR               PIC X COMP-X.
05  LDFC-MINUTE            PIC X COMP-X.
05  LDFC-SECOND            PIC X COMP-X.
05  LDFC-HUNDREDTHS       PIC X COMP-X.
03  LISTDIR-FILE-LAST-ACCESS-TIME.
05  LDFLA-YEAR             PIC XX COMP-X.
05  LDFLA-MONTH           PIC X COMP-X.
05  LDFLA-DAY             PIC X COMP-X.
05  LDFLA-HOUR            PIC X COMP-X.
05  LDFLA-MINUTE          PIC X COMP-X.
05  LDFLA-SECOND          PIC X COMP-X.
05  LDFLA-HUNDREDTHS     PIC X COMP-X.
03  LISTDIR-FILE-LAST-MODIFICATION-TIME.
05  LDFLM-YEAR            PIC XX COMP-X.
05  LDFLM-MONTH           PIC X COMP-X.
05  LDFLM-DAY             PIC X COMP-X.
05  LDFLM-HOUR            PIC X COMP-X.
05  LDFLM-MINUTE          PIC X COMP-X.
05  LDFLM-SECOND          PIC X COMP-X.
05  LDFLM-HUNDREDTHS     PIC X COMP-X.
03  LISTDIR-FILE-SIZE     PIC X(8) COMP-X.

```

The data items are defined below.

Note: Because the supported file types vary by operating system, the following data items have slightly different meanings depending on your operating system. Even on *operating* systems that support these values, some *file* systems may not. Some versions of the UNIX® operating system may change these values when permissions are changed. Refer to your operating system documentation for specific definitions.

LISTDIR-FILE-TYPE is one of the following:

B = block device

C = character device

D = directory

F = regular file

P = pipe (FIFO)

S = socket

U = unknown

LISTDIR-FILE-CREATION-TIME is reported in the manner defined by your operating system. In general, the “creation time” is the date (and time) that the file was originally created. Some versions of UNIX may change this time when permissions are changed.

LISTDIR-FILE-LAST-ACCESS-TIME is reported in the manner defined by your operating system. The “last access” time is the date (and time) that the file was last accessed by some application (usually when the file was queried in some way). Some versions of UNIX may change this time when permissions are changed.

LISTDIR-FILE-LAST-MODIFICATION-TIME is reported in the manner defined by your operating system. The “last modification time” is the date (and time) the file was last written to. Some versions of UNIX may change this time when permissions are changed.

LISTDIR-FILE-SIZE contains the size of the file in bytes.

W\$FLUSH Routine

The W\$FLUSH library routine refreshes the display so that the user sees the most current screen and cursor, even if an ACCEPT has not been performed. The runtime system calls this routine in a variety of circumstances. Doing so generally eliminates the need for COBOL programs to call the routine directly.

Sometimes there can be a performance penalty for these runtime calls, especially during file processing where the screen is updated after each record. The W\$FLUSH routine has been enhanced so that you can choose to inhibit calls to W\$FLUSH if you do not want the screen updated after each change. This has been done with the addition of two new parameters:

256 inhibits any future calls to W\$FLUSH, including those made internally by the runtime system. Only a call with parameter “257” will be honored. This parameter must be turned off to restore user interaction.

257 cancels the inhibited state caused by parameter “256”. This parameter does not cause a flush itself; it just allows future calls to W\$FLUSH to function. You must call this parameter before interacting with the user.

For example:

```
78  INHIBIT-FLUSH  VALUE 256.
78  ALLOW-FLUSH   VALUE 257.

CALL "W$FLUSH" USING INHIBIT-FLUSH
PERFORM LARGE-ACTIVEX-UPDATE
CALL "W$FLUSH" USING ALLOW-FLUSH
```

One possible use for this feature is to simulate a “mass update” functionality for ActiveX[®] controls that do not have this capability built in.

Refer to Appendix I of the ACUCOBOL-GT manual set for more information about the W\$FLUSH routine.

END PROGRAM Extended Syntax

The END PROGRAM syntax has been extended to support syntax that is common to other COBOLs. END PROGRAM now allows:

```
END PROGRAM string-literal.
```

Getting the Value of an Environment or Configuration Variable

New syntax supports a two-step process for getting the value of an environment or configuration variable. This syntax is compatible with syntax that is common in other COBOLs. The syntax works as follows:

Step 1: Set a special register to the name of the environment variable for which you want the value. This is done with the DISPLAY verb, as follows:

```
DISPLAY name UPON ENVIRONMENT-NAME
```

where *name* is a Working-Storage item of alphanumeric type and is the name of the environment variable or configuration variable of interest.

Step 2: After the special register is set, you can query the value of the environment variable by executing:

```
ACCEPT value FROM ENVIRONMENT-VALUE
```

where *value* is a Working-Storage item of the type needed to hold the value of the configuration or environment variable.

This pair of statements is equivalent to:

```
ACCEPT dest-item FROM ENVIRONMENT env-name  
(ACCEPT statement, Format 5)
```

To compile the new syntax, you must include the "--521" compile flag on the compiler command line (this requirement will be removed in a later release). Programs compiled in this way cannot be run with a pre-5.2.1 runtime.

REDEFINES Phrase Referencing a Definition

The REDEFINES phrase has been enhanced to allow it to reference an item that is itself a redefinition of an area. The ANSI Standard requires that a REDEFINES phrase reference the data item that originally defined the memory area. However, several other COBOL compilers allow a REDEFINES phrase to reference items that are themselves a redefinition of an area. This enhancement adds that extended referencing to ACUCOBOL-GT.

For example, prior to Version 5.2.1, the following code would generate a compilation error:

```
01  ITEM-1                                PIC 9(10) .  
01  ITEM-2 REDEFINES ITEM-1              PIC X(10) .  
01  ITEM-3 REDEFINES ITEM-2 .  
    03  ITEM-3A                            PIC X(5) .  
    03  ITEM-3B                            PIC X(5) .
```

The code generated the error because ITEM-3 references ITEM-2, which is itself a redefinition. Starting with Version 5.2.1, this code is accepted by the compiler.

ADDRESS OF Phrase in Expressions

ACUCOBOL-GT now accepts the ADDRESS OF construct in expressions. In an arithmetic expression, you may use the address of a data item anywhere you could normally place a numeric data item. The address is treated as an unsigned integer with sufficient range to express any address in the host machine's address space. For example, the following statement is now accepted by the compiler:

```
IF ADDRESS OF LINKAGE-ITEM-1 = NULL
  SET ADDRESS OF LINKAGE-ITEM-1 TO WS-ITEM-1
```

To create an address, you would use the following phrase:

ADDRESS OF *data-item-1*

where *data-item-1* is a data item.

For example, the following statement produces a pointer that points one character position past the beginning of ITEM-1:

```
COMPUTE PTR-1 = ADDRESS OF ITEM-1 + 1
```

LENGTH OF Phrase, Subscripting for Table Items

The LENGTH OF form of numeric literals has been enhanced to allow subscripts on the specified data name if the data name refers to a table item. This makes it possible to use the LENGTH OF phrase to determine the size of a single table element. For example, the following is now accepted by the compiler:

```

01  TABLE-1 .
    03  ITEM-1 OCCURS 10 TIMES PIC X(5) .

77  SIZE-1
                                     PIC 9(5) .

MOVE LENGTH OF ITEM-1(1) TO SIZE-1

```

This moves “5” to SIZE-1.

Windows Print Spooler, Printing Multiple Jobs Simultaneously

It is now possible to print multiple print jobs simultaneously using the ACUCOBOL-GT runtime. This means that you can open multiple file descriptors that point to “-P SPOOLER” or “-P SPOOLER-DIRECT” at the same time. (More information about the Windows operating system print spooler is found in the *Getting Started* book.) For example, you may have two simultaneous print jobs:

```

SELECT FIRST-FILE
    ASSIGN TO PRINTER "-P SPOOLER".

SELECT SECOND-FILE
    ASSIGN TO PRINTER "-P SPOOLER".

..

PROCEDURE DIVISION.

..

    OPEN OUTPUT FIRST-FILE.
    OPEN OUTPUT SECOND-FILE.

```

and both will print to the default Windows printer without interfering with each other. You can call WIN\$PRINTER USING WINPRINT-SETUP before one or both of the OPEN statements. Each file may have individual file status variables or may refer to a common file status variable.

This does not mean that you can open a single file descriptor multiple times. For example, the following returns a file status indicating that the file is already opened:

```
SELECT FIRST-FILE
      ASSIGN TO PRINTER "-P SPOOLER".

..

PROCEDURE DIVISION.

..

      OPEN OUTPUT FIRST-FILE.
      OPEN OUTPUT FIRST-FILE.
```

This is normal behavior and is consistent with the way file handling is implemented in COBOL and in other programming languages.

If you are using only the verbs OPEN, CLOSE, and WRITE, no further changes to your code are needed. If you are using WIN\$PRINTER functionality (other than WINPRINT-SETUP), you must specify which print job is targeted. (The WIN\$PRINTER library routine is documented in Appendix I of the ACUCOBOL-GT manual set.) This can be done in two ways:

1. The simplest way is to execute the WIN\$PRINTER operation immediately after an OPEN or WRITE statement on the intended job. Every execution of OPEN and WRITE sets the current job as the default so that subsequent activity using WIN\$PRINTER is automatically directed to the job that was last accessed with an OPEN or WRITE statement.

In this situation, if you have multiple jobs running, and you close one of them, the runtime switches to the next job in the list. For example, if you are printing jobs 1, 2, and 3, and you close job 2, the close command sets the current job to 3. If there is no job 3, the runtime attempts to set to the job that preceded the closed job (which in this case would have been job 1). If this is the only job, it is initialized.

2. The other method is to use a new WIN\$PRINTER operation, WINPRINT-SET-JOB.

This operation has the following syntax:

```
CALL "WIN$PRINTER"  
    USING WINPRINT-SET-JOB JOB-ID  
    GIVING PRINT-JOB.
```

If you set JOB-ID to "0", WINPRINT-SET-JOB returns the identifier of the job that is currently spooling into the printer (PRINT-JOB). You can then use that number to tell WIN\$PRINTER operations which print job is the target. This is your only way to obtain the ID of a job. (This number is compatible with, and may be used in conjunction with, the operations WINPRINT-SET-JOB-STATUS and WINPRINT-GET-JOB-STATUS.) The call should be issued immediately after the opening of a job. For example:

```
OPEN OUTPUT FIRST-FILE.  
CALL "WIN$PRINTER" USING WINPRINT-SET-JOB JOB-ID GIVING FIRST-  
ID.
```

where FIRST-ID is a variable declared signed integer, such as:

```
77 FIRST-ID USAGE SIGNED-INT.
```

Subsequent calls to WIN\$PRINTER may use FIRST-ID to identify the target for the next action, as follows:

```
OPEN    OUTPUT    FIRST-FILE.  
CALL    "WIN$PRINTER" USING  
        WINPRINT-SET-JOB 0  
        GIVING    FIRST-ID.  
  
OPEN    OUTPUT    SECOND-FILE. | Is now current.  
  
*Initialize the print record for the first print job.  
MOVE    "This is job 1, printed with MS Sans Serif." TO  
        RECORD-FILE-1.  
  
*Initialize the print record for the second print job.  
MOVE    "This is job 2, printed with Script." TO  
        RECORD-FILE-2.
```

```

*Set active job to the first print job.
CALL      "WIN$PRINTER"  USING
          WINPRINT-SET-JOB FIRST-ID.

*Set the preferred font for the first print job.
INITIALIZE          WINPRINT-DATA.
MOVE      FIRST-FONT      TO WPRTDATA-FONT.
CALL      "WIN$PRINTER"  USING
          WINPRINT-SET-FONT
          WINPRINT-DATA.

```

If you try to perform an operation that requires an active print job and there is none, an error status is returned. This series of calls can be used with all WIN\$PRINTER functions, with the following exceptions:

- The status of a printer cannot be determined via WINPRINT-GET-PRINTER-STATUS or WINPRINT-GET-JOB-STATUS unless the print job has already started. This is because the port monitor must both detect a print error and report it to the printer queue before it can be recognized by WIN\$PRINTER functions.
- Due to a limitation in the Microsoft® Windows API, computers that run Windows 9x (Windows 95, Windows 98, and Windows ME) do not return the spooler job ID when opening a print job. This means that you cannot use the WINPRINT-GET-JOB-STATUS and WINPRINT-SET-JOB-STATUS operations of the WIN\$PRINTER library routine on these machines. (These operations are used to check and modify the status of a particular printer.)

Note: When you are printing multiple jobs simultaneously, you should not set a printer font before the print job has been opened because the font could be applied to the wrong job. Once the print job is opened, you may set the font, using the JOB-ID of the target printer.

However, if you need to change the printer settings for a subsequent job on a different printer, you should set JOB-ID to “-1” before setting WINPRINT-SET-SETTINGS or WINPRINT-SET-PRINTER(-EX). This causes WINPRINT-SET-JOB to return the ID number of the next job in the queue (after the current job). This should be done just prior to calling an OPEN statement. When JOB-ID is set to “-1”, the runtime executes the next

WIN\$PRINTER operation as if no current job were printing. This does not affect existing jobs, but it affects the status of subsequent jobs, unless it is an OPEN, WRITE, or CLOSE statement.

ACUCOBOL-GT Runtime

CALLing ENTRY Points in an Object Library

Prior to Version 5.2.1, when the runtime loaded an object library (“-y”), objects had to be CALLED by the program before the runtime would recognize ENTRY points. Now, any ENTRY points within the objects contained in the library are automatically loaded at startup, and can be CALLED by their ENTRY point names. To use this feature, simply put your program in an object library file, and include the “-y” option in the runtime command.

File Access to Windows Shared Files

On Windows systems, the speed of file access to Vision files located on a Windows shared drive can be variable. The cause of this variability is attributed to an imperfect interaction between the way the Windows operating system handles file locks and the way it caches shared files on the local system. In Version 4.3.1, to improve performance, the method used to access Vision files on Windows shared drives was changed. Subsequently, although most users experienced good performance and no problems, some users experienced sporadic file corruption problems (error 98, 90). To minimize the possibility of these errors, in Version 5.2.1 the runtime has reverted to the pre-4.3.1 method of opening files. Although this method can result in somewhat slower performance, it is safer. The faster method can still be used and is configurable via a Windows registry entry.

CAUTION: Changing the Windows registry incorrectly can cause serious problems that may necessitate a complete reinstallation of the operating system. Changes to the Windows registry should be made only by a qualified Windows system administrator. Changes to the Windows registry are not recommended.

To change the runtime to use the faster, but riskier method of opening files on Windows shared drives, in the registry key:

```
HKEY_LOCAL_MACHINE\Software\Acucorp, Inc.\ACUCOBOL-GT
```

manually set a registry value of type DWORD:

```
GetUniqueId Uses CreateFile
```

to the value “0”. When this entry is set to “1” (the default), the safer method is used.

Performance Tip: File OPENS are relatively expensive operations, especially on MS-DOS® and Windows systems. Eliminating non-essential file OPENS could significantly improve application performance.

File Trace Timestamps

A new timestamp option has been added to file trace output. When you’re debugging a program, it’s sometimes helpful to know the exact time of each file operation. File trace debugging output now offers this capability. When you enable the timestamp option, a timestamp is placed at the beginning of every line in the trace file. The format of the timestamp is HH:MM:SS.mmmmmm, where “mmmmmm” is the finest resolution that ACUCOBOL-GT can obtain from the system.

Notes:

- Timestamp information is included only when file trace information is directed to a file. Timestamp information is never included in file trace output that is sent to the screen.
 - Enabling the timestamp option can add significant overhead and may have a noticeable impact on performance.
-

There are two ways to enable timestamp information in the file trace.

1. Before starting the program, set the runtime configuration variable `FILE_TRACE_TIMESTAMP` to “1” (on, true, yes). The default value is “0” (off, false, no).
2. In the debugger, use the “t timestamp” command. To obtain a timestamp, you must direct file trace information to a file.

This function is also available for AcuLaunch. For this product, set the `FILE_TRACE_TIMESTAMP` variable in the server configuration file.

File Handling Performance Improvement

The logic used to commit data to disk has been improved. In prior versions, *all* open files were written to disk. ACUCOBOL-GT now keeps track of which files have been updated, and the runtime writes only those files to disk. Flushing files to disk can be an expensive operation and is especially so on DOS/Windows systems. This optimization can improve the performance of programs that use a large number of files that are opened and closed frequently.

The Address of Unassigned Memory

In prior versions, referring to the ADDRESS OF a data item that had never been given an address would generate a runtime error. Starting with Version 5.2.1, these addresses are now treated as if the owning group item has an address of zero. The rule is: if a data item has not been given an address by the program, the data item’s address acts as if its uppermost parent item (or itself if it has no parent) had an address of zero. This rule is used only for ADDRESS OF calculations.

This makes it possible to test for a NULL address without generating a runtime exception. For example, if LINK-ITEM-1 was a LINKAGE data item that had not been passed into the program, the following code would cause execution to terminate in versions prior to 5.2.1:

```
SET POINTER-1 TO ADDRESS OF LINK-ITEM-1
IF POINTER-1 = NULL
    SET ADDRESS OF LINK-ITEM-1 TO WS-ITEM-1
```

Starting with Version 5.2.1, LINK-ITEM-1 is treated as if it has an address of zero.

Debugger Enhancements: Step Into (S) and Auto Step (SA)

The debugger's Step Into (S) command has been changed so that new threads are followed as they are created. (The original behavior was to track only the original thread.) This change also affects the Auto Step (SA) command in the same way. If you want to step through a program following only the original thread, use the Step Over (P) command instead.

New Runtime Configuration Variables

SHARED_LIBRARY_EXTENSION: This configuration variable allows you to define the file name extension to be used with UNIX shared libraries. The default value is “.so”. This variable has meaning only on systems that support UNIX shared libraries.

TC_CONTROL_SYNC_LEVEL: This variable determines which VALUE data items in a Screen section are updated when a BEFORE, AFTER, or EXCEPTION procedure executes under the Thin Client. (This variable affects only BEFORE, AFTER, and EXCEPTION procedures. The values of all variables are made current anytime an ACCEPT terminates.) The setting of this variable can affect performance. The possible values for TC_CONTROL_SYNC_LEVEL are:

- 1 (default) only the VALUE data item associated with the current field is updated when its AFTER or EXCEPTION procedure executes.
- 2 only the VALUE data item associated with the current field is updated when its BEFORE, AFTER, or EXCEPTION procedure executes.

- 3 all VALUE data items are updated when any BEFORE, AFTER, or EXCEPTION procedure is executed.

For best performance, Acucorp recommends leaving this variable at its default setting of “1” unless that causes your program to perform incorrectly. In that case, you can increase the setting of TC_CONTROL_SYNC_LEVEL to “2” or “3” to adjust for problems in the application behavior.

Note: Alternatively, you can directly INQUIRE the value of a control in an embedded procedure. This allows you to tune application performance more precisely than TC_CONTROL_SYNC_LEVEL allows.

WINPRINT_NAMES_ONLY: This variable allows you to generate a list of the names of printers installed on a Windows PC. It does this by altering the behavior of some of the operations of the WIN\$PRINTER library routine. When WINPRINT_NAMES_ONLY is set to a value of “1” (on, true, yes), the WIN\$PRINTER operations that retrieve printer information return only the names of installed printers, rather than the real-time status of all available printer capabilities.

This variable can be set in the configuration file or directly in your program with the following code:

```
SET ENVIRONMENT "WINPRINT-NAMES-ONLY" TO "1".
```

When this variable is turned on, the following operations of the WIN\$PRINTER library routine are affected:

- WINPRINT-GET-PRINTER-INFO
- WINPRINT-GET-PRINTER-INFO-EX
- WINPRINT-GET-CURRENT-INFO
- WINPRINT-GET-CURRENT-INFO-EX

Instead of returning detailed information about the capabilities of each printer (duplex, orientation, etc.), the routine returns only the name of the printer. This can provide a significant performance improvement, particularly with networked printers.

If you are using the default printer settings, set the WINPRINT_NAMES_ONLY variable to "1", generate a list of printer names using the WINPRINTER-GET-PRINTER-INFO-EX operation, and select the desired printer.

If you want to modify the printer settings, such as the number of copies or the paper orientation, you should perform the steps described above, and then reset WINPRINT_NAMES_ONLY back to the default of "0" (off, false, no). You may then use WINPRINT-GET-PRINTER-INFO-EX to obtain detailed information about the capabilities of the selected printer.

See "Windows Print Spooler: Printing Multiple Jobs Simultaneously," earlier in this document, for additional information about WIN\$PRINTER enhancements. Refer to Appendix I and Appendix H in Book 4 of the compiler documentation for information on library routines and configuration variables, respectively, in an ACUCOBOL-GT environment. The sample program "prndemox.cbl" may also be helpful.

Defining Windows Function Key Combinations

A number of function key combinations can now be defined by Windows users. (See the list below.) These keys have no specified default action, and the key combinations are recognized by the Windows keyboard driver only if they are assigned a definition. Functions are defined with the "KEYSTROKE" configuration variable. For example, the following line in a program would assign "Shift-Ctrl-F1" to return an exception value of "201":

```
set environment "keystroke" to "exception=201 S1"
```

Key	Key Code
Alt-F1	a1
Alt-F2	a2
Alt-F3	a3

Key	Key Code
Alt-F5	a5
Alt-F7	a7
Alt-F8	a8
Alt-F9	a9
Alt-F10	a0
Alt-F11	U7
Alt-F12	U8
Shift-Ctrl-F1	S1
Shift-Ctrl-F2	S2
Shift-Ctrl-F3	S3
Shift-Ctrl-F4	S4
Shift-Ctrl-F5	S5
Shift-Ctrl-F6	S6
Shift-Ctrl-F7	S7
Shift-Ctrl-F8	S8
Shift-Ctrl-F9	S9
Shift-Ctrl-F10	S0
Shift-Ctrl-F11	U9
Shift-Ctrl-F12	U0

Notes:

- Alt-F4 and Alt-F6 are reserved for use by Windows and are not included in the table.
 - Shift-Ctrl-F10 may be used only if the configuration option F10_IS_MENU is set to “false”. When F10-IS-MENU is set to the default of “true”, Shift-Ctrl-F10 activates context menus (such as a control’s pull-down menu).
-

The Web Runtime Control

Version 5.2.1 introduces the ACUCOBOL-GT Web Runtime, an ActiveX control, that enables you to run ACUCOBOL-GT applications over the Internet. The Web Runtime is designed for Internet Explorer Versions 5.0 and above. You can use it as an alternative for browsers that no longer support Netscape® style plug-ins. The Web Plug-in, available in earlier versions of ACUCOBOL-GT, is also supported in Version 5.2.1. Both the Web Plug-in and the Web Runtime are included with the ACUCOBOL-GT runtime.

Note that the Web Runtime works only with Internet Explorer Versions 5.0 and above. The Web Plug-in control works only with Netscape and with Internet Explorer Versions 5.5, SP1 and below. Starting with Version 5.5, SP2, Internet Explorer does not support plug-ins. For complete information on the new Web Runtime control, please see Chapter 5 in the Version 5.2.1 *Programmer's Guide to the Internet*.

Acucorp's Thin Client Technology With AcuLaunch

Version 5.2.1 of Acucorp's Thin Client technology contains the enhancements described in the following paragraphs. This version also includes several performance enhancements and corrections, which are described in the ECN list located in the Support area of the Acucorp Web site, <http://www.acucorp.com>. Please refer to the AcuLaunch section of the ECN list for several of these ECNs. Other Thin Client ECNs appear in the ACUCOBOL-GT section of the list.

Executing Desktop Programs Using C\$SYSTEM

The ACUCOBOL-GT Thin Client is enhanced in Version 5.2.1 to allow the server program to start desktop applications via the C\$SYSTEM library routine. Previously, only ActiveX controls could be executed on the client machine. For information regarding the use of C\$SYSTEM, refer to section 3.6 in *A Guide to Using Acucorp's Thin Client Technology*.

New Configuration Variables

FILE_TRACE_TIMESTAMP: This configuration variable lets you turn on trace file timestamping. For more information, refer to the "File Trace Timestamps" section in this Release Overview.

TC_CONTROL_SYNC_LEVEL: This variable lets you control which VALUE data items are updated after a BEFORE, AFTER, or EXCEPTION procedure is executed. For more information, refer to section 3.3.1 in *A Guide to Using Acucorp's Thin Client Technology*.

MSG-SB-THUMBTRACK Messages

In order to minimize network traffic and optimize Thin Client performance, some control messages that are considered expensive are suppressed. MSG-SB-THUMBTRACK, sent when the user is interacting with the scroll bar, is one such message. With Version 5.2.1, the Thin Client sends a single MSG-SB-THUMBTRACK message immediately before the MSG-SB-THUMB message, and the scroll bar is automatically updated. Note that this behavior occurs only with scroll bars having the TRACK-THUMB style.

Technical Support

If we can help you in any way, please contact Acucorp Technical Support. We can be reached at the following numbers:

Acucorp Technical Support in the United States:

Phone: +1 (858) 689-4500
1-800-262-6585 (within the U.S.)

Fax: +1 (858) 689-4552

E-mail: support@acucorp.com

Customers outside of the United States, please contact your regional Technical Support office or your local distributor:

In Benelux:

Phone: +31 (0)73 623 01 95

Fax: +31 (0)73 623 01 99

E-mail: nlsupport@acucorp.com

In France:

Phone: +33 (0)1 53 34 9005

Fax: +33 (0)1 53 34 9001

E-mail: frsupport@acucorp.com

In Germany:

Phone: +49 (0)61 75 9331 33

Fax: +49 (0)61 75 1429

E-mail: desupport@acucorp.com

In Scandinavia and the United Kingdom:

Phone: +44 (0)20 8843 7101

Fax: +44 (0)20 8744 9401

E-mail: uksupport@acucorp.com