Release Overview

extend® Product Suite

Version 8.0

Acucorp, Inc.
8515 Miralani Drive
San Diego, CA 92126
858.689.4500

E-01-RO-070924-Release Overview-8.0

Contents

Chapter 1: Overview of *extend*® Version 8.0

1 Overview of *extend*® Version 8.0

Key Topics

1.1 Introduction

Acucorp is pleased to announce Version 8.0 of ***extend***®, the next release in our Interoperability Series. Version 8.0 introduces both new products and enhancements that increase interoperability with other languages and technologies. Additional enhancements provide added compatibility with other COBOL, and faster performance for math-intensive applications. Version 8.0 introduces the following top features:

AcuXUI™
This new product is a cross-platform user interface engine that allows a graphical ACUCOBOL-GT program to display its user interface on UNIX and Linux platforms as well as Windows platforms. With AcuXUI, you do not directly execute your COBOL program using the runtime. Rather you run your graphical application by issuing a Java command on the server command line indicating the resources to use to run the COBOL program. Because of its platform independence, AcuXUI lets you deliver one graphical program on multiple platforms with predictable results. AcuXUI is documented in Chapter 11 of the *ACUCOBOL-GT User Interface Programming Guide.*

Version 8.0 also features numerous enhancements to improve your application's overall performance:

- ACUCOBOL-GT now uses a binary math operations by default. This results in faster performance for many programs
- Vision has been enhanced to efficiently WRITE or READ NEXT indexed files while a lock is held on the file. This can lead to a significant improvement in runtime performance.
- The IS NUMERIC conditional operation on alphanumeric or unsigned USAGE DISPLAY data items now operate much faster than before.
- The compiler now produces faster code for comparison operations involving reference modified data items. The improvement tends to be more pronounced when native code is generated.

- Many operations that involve reference modification containing arithmetic expressions are now faster. Again, this improvement is more pronounced when you are generating native code.

- The compiler now produces more efficient code for the GO TO DEPENDING ON statement.

Version 8.0 also includes several enhancements that are focused on IBM compatibility. This includes support for XML PARSE and XML GENERATE and the addition of separate options to the compiler switch "-Cv" for OSVS and VSC2.

A new remote preprocessing utility called **Boomerang** provides client and server technologies that enable you to automatically transfer files to a remote server, invoke and perform preprocessing on that server, then return the preprocessed files to your client machine where additional compiling can occur. Many proprietary or third party preprocessors have machine-specific functions that require preprocessing to occur in their native environments. **Boomerang** makes accessing these types of preprocessors easier and more efficient. Boomerang is documented in Chapter 3 of the *ACUCOBOL-GT User's Guide.*

Finally, nearly all of Acucorp's products have been enhanced to provide greater flexibility, to simplify development efforts, and to increase your application's functionality. For example:

- Bringing AcuBench® into the forefront of enterprise application development by allowing remote debugging of IBM CICS applications, support for AcuXUI, and Boomerang, modernized toolbar and menu icons, HTML report improvements, and new styles and configuration options.

- Porting of ACUCOBOL-GT, AcuConnect®, AcuSQL®, and Acu4GL® to run on the Windows x64 platform.

- Reengineering of AcuSQL and Acu4GL for Microsoft SQL Server to access Microsoft SQL Server using the latest Microsoft communication technology.

In addition to highlighting enhancements by solution category, this *Release Overview* highlights new features in Version 8.0 for the following technologies:

- ACUCOBOL-GT Compiler
- ACUCOBOL-GT COBOL Virtual Machine™ (runtime)
- AcuBench
- AcuConnect
- AcuXDBC™
- AcuSQL
- Acu4GL

Unless otherwise indicated in this overview, please refer to their respective manuals for complete details. See each product's online *Release Notes* for information not included in the manuals.

1.2 Interoperability Enhancements

This section describes enhancements related to working with other technologies such as Java, .NET, and ActiveX controls.

Added support for DRV and OCX files

Prior to Version 8.0, you could not call on modules with extensions ".drv" or ".ocx" even though these were dynamic linked libraries just as those with the extensions "dll". Starting with Version 8.0, the runtime supports calls to all of these file types. For information on calling DLLs, refer to *A Guide to Interoperating with ACUCOBOL-GT*, Chapter 3, section 3.3.

Enhancements to Java Interoperability (C$JAVA)

When COBOL calls Java from the runtime and a Java exception occurs, it is now possible to get a stack trace of this information. The exception information will be written to stderr or the "-le" error file.

Version 8.0 also adds functionality for handling Java exceptions that might be thrown when the C$JAVA library routine calls Java from COBOL. There are two new codes in samples/def/java.def to handle these exceptions: CJAVA-EXCEPTIONOCCURRED and CJAVA-GETEXCEPTIONOBJECT.

Another new op-code, CJAVA-SETSYSTEMPROPERTY, lets you change a system property from a C$JAVA call. For example, you can set the CLASSPATH environment variable this way.Other changes simplify the use of C$JAVA. Namely, you no longer have to pass a method signature for a default constructor when using CJAVA-NEW with C$JAVA and you no longer have to pass the name of a class when using CJAVA-CALL or CJAVA-CALLNONVIRTUAL. These behaviors are still supported for backward compatibility.

Another op-code, CJAVA-CALLJAVAMAIN, has been added as a convenience for the purpose of simplifying calls to a Java main method. It used to require four calls to C$JAVA to call a Java main function and possibly some MOVE statements in order to populate the string array. With the addition of CJAVA-CALLJAVAMAIN. The same thing can be accomplished with one call to C$JAVA.

In addition, C$JAVA has been enhanced to prevent the most common configuration problems that arise when calling Java from COBOL, specifically the inability to find the Java classpath and library path settings. Starting with Version 8.0.0, the runtime checks what is being set using JAVA_OPTIONS. If it is not java.class.path, then the runtime sets CLASSPATH from the configuration variable. If it is not java.library.path, then it looks first for LD_LIBRARY_PATH and sets java.library.path using that. Otherwise it looks for LIBPATH and PATH in the environment in that order and sets java.library.path using either of those. This causes the runtime to behave more consistently regardless of platform.

For information on calling Java from COBOL, refer to *A Guide to Interoperating with ACUCOBOL-GT*, Chapter 2, section 2.3. The C$JAVA routine is also documented in *ACUCOBOL-GT Appendices*, Appendix I.

ActiveX Enhancements

Support for Nested ActiveX and COM Event Procedures has been added to Version 8.0. When an application dialog contains an ActiveX control or COM object that is assigned an event procedure, the event handler sometimes triggers additional ActiveX or COM events. By default the event procedure is nested, however starting with Version 8.0, you can use the NESTED_AX_EVENTS configuration variable to specify whether or not the event procedure should be nested. For more information on ActiveX events, refer to *A Guide to Interoperating with ACUCOBOL-GT*, Chapter 4, section 4.4.

.NET Enhancements

Previous versions of ACUCOBOL-GT supported .NET assemblies version 1.0. As of Version 8.0, ACUCOBOL-GT supports .NET assemblies version 1.1 and 2.0. The NETDEFGEN utility has been enhanced to recognize both versions of assemblies, and the compiler has been enhanced to compile both versions of assemblies.

In Version 8.0, NETDEFGEN's interface has been simplified and a Settings dialog has been added to eliminate the hand-generated options file, help the implementation of multiple .NET versions, and add persistence to your directory selections between invocations.

In addition, there have also been large amounts of code refactoring, optimizations, and improved annotations as well as the elimination of two source files.

The COPY files generated by NETDEFGEN have been enhanced to improve their readability. Namely:

- Eliminated the recording of some annotations when they weren't relevant.

- Formatted enumerations so that they are easier to read.
- Added processing so that instead of object members (methods, properties, events, etc.) being recorded as they're found in the supplied array (a property here, and event there, another property, etc.), they are grouped together into sections.
- All generated IL source files have had their formatting improved as well.
- Updated the utility's help file.
- The comment ahead of each method/property in these files now includes a reference to the current class to provide context.

For information on working with .NET assemblies from COBOL, refer to *A Guide to Interoperating with ACUCOBOL-GT*, Chapter 5.

C$XML Enhancements

Comments

Comments in C$XML are handled slightly different starting with Version 8.0.

1. They no longer include the expression:

   ```
   '.* - generated by ACUCOBOL-GT v.*'
   ```

   If you depend on having those comments, you will need to rework your application in some way.

2. They are now separated by a single byte of low-values as they are in the XML file.

New C$XML Op-codes for Encoding, Stand-alone, and Processing Instructions

Starting with Version 8.0, any stylesheet (or other processing instructions) can now be retrieved from a parsed XML file and set when creating an XML file. (Processing instructions are any non-comment lines after the first ?xml tag, and before the first top-level element.)

Further, the version, encoding, and stand-alone attributes are retrieved from the file, and are available to the COBOL program. The COBOL program can also set new values for these attributes.

Finally, the DOCTYPE information can be retrieved and set.

All of this information is included in the file when it is written. If the COBOL program does not modify it, then it will be written just as it was read when the file was parsed.

There are nine new op-codes for C$XML to enable these functions. All have been added to "acucobol.def". These include:

CXML-GET-PROC-INSTR-COUNT (OP-CODE 36)
CXML-GET-PROC-INSTR (OP-CODE 37)
CXML-GET-VERSION (OP-CODE 39)
CXML-SET-VERSION (OP-CODE 40)
CXML-GET-ENCODING (OP-CODE 41)
CXML-SET-ENCODING (OP-CODE 42)
CXML-GET-STANDALONE (OP-CODE 43)
CXML-SET-STANDALONE (OP-CODE 44)
CXML-GET-RAW-DOCTYPE-LEN (OP-CODE 45)
CXML-GET-RAW-DOCTYPE (OP-CODE 46)
CXML-SET-RAW-DOCTYPE (OP-CODE 47)

Two new error codes can now be returned:

CXML-INVALID-PROC-INSTR-NUMBER (value 17)
CXML-NO-PROCESSING-INSTRUCTIONS (value 18)

For information on using the C$XML routine, refer to *A Guide to Interoperating with ACUCOBOL-GT*, Chapter 11, section 11.2.6. This library routine and its op-codes are also documented in *ACUCOBOL-GT Appendices*, Appendix I.

Internet Enhancements

The maximum size of input to a CGI program when REQUEST_METHOD=POST has been raised from 32K to 2GB.

1.3 Data Access Enhancements

In addition to the data access enhancements described in this section, Version 8.0 introduces several enhancements to ACUCOBOL-GT utilities related to data manipulation such as **AcuSort** and **vutil**. See the section of this *Release Overview* for details on these enhancements.

Enhanced Data Format Information for XFD Files

Additional information is added to the Identification Section of Version 6 XFD files that indicates compiler options or program settings, which may have been applied to the XFD. This includes fields that identify the following information:

- Sign compatibility (specified by the various "-Dc" compiler options)
- Maximum numeric digits (set by the "-Dd31" compiler option)
- Program period (decimal value of the character used as the program period)
- Program comma (decimal value of the character used as the program comma)

XFDs are documented in Chapter 5 of the *ACUCOBOL-GT User's Guide*.

Large (>2GB) Transaction Log Enhancement for Vision

Vision (and **logutil**) have been enhanced to allow transaction log files to be larger than 2GB. Note that the host operating system must allow such large files and that on UNIX the "USE_LARGE_FILE_API" configuration variable must be set.

1.4 Compatibility With Other COBOLs

ACUCOBOL-GT is enhanced to make it more compatible with other COBOL systems. In addition to the following sections, please refer to the section titled “New and Enhanced Configuration Variables” for information regarding new configuration variables for HP COBOL and Micro Focus compatibility.

++INCLUDE Syntax Support

The compiler now supports INCLUDE syntax supported by other COBOL vendors. The compiler understands ++INCLUDE to be a synonym to COPY. Filenames referenced by ++INCLUDE are searched for in the same way as COPY files.

Added Support for IBM COBOL XML Verbs

ACUCOBOL-GT now supports the IBM Enterprise COBOL XML GENERATE and XML PARSE statements. This simplifies IBM migrations and gives you an alternate way to read and write XML data from COBOL.

XML PARSE gives you a way to parse XML data and process it in a COBOL program, associating processing procedures with the exception and non-exception cases that can result from the parse. XML GENERATE gives you a way to translate COBOL data into XML.

As before, ACUCOBOL-GT also includes AcuXML, a runtime-resident interface that transparently converts XML data to sequential files for COBOL processing, and C$XML, a library routine that gives you precise control over which elements or attributes of the data to parse. A developer utility called **xml2fd** creates File Descriptors (FDs) and SELECT statements from existing XML files to support AcuXML.

All three of these approaches can be used to parse records-based XML files; however, only C$XML and XML PARSE can be used to parse non-records-based XML files.

For information on...	**See...**
AcuXML and xml2fd	Sections 11.2.2 through 11.2.5 of *A Guide to Interoperating with ACUCOBOL-GT*
C$XML library routine	Section 11.2.6 of *A Guide to Interoperating with ACUCOBOL-GT*, and Appendix I in *ACUCOBOL-GT Appendices*
XML GENERATE and XML PARSE statements	*ACUCOBOL-GT Reference Manual*, Chapter 6, "Procedure Division"

COPY/REPLACING and REPLACE Syntax Extensions

ACUCOBOL-GT now supports the use of colons, parentheses, and asterisks as delimiters for substring replacing.

FILE STATUS Extensions

The compiler now allows a PIC 99 data item to be used as a FILE STATUS item without issuing a warning. In addition, the compiler now supports the specification of a second FILE STATUS item, which is treated as commentary. Both of these changes are for IBM/COBOL compatibility.

The FILE STATUS phrase of the File-Control paragraph now supports the following format:

```
[ File STATUS Is status-variable [status-variable-2] ]
```

New Conditions for EVALUATE WHEN Statement

Several new conditions are now allowed for the EVALUATE WHEN statement, as shown below:

```
WHEN <
WHEN IS LESS THAN
WHEN <=
WHEN IS LESS THAN OR EQUAL TO
WHEN =
WHEN IS EQUAL TO
WHEN EQUALS
WHEN >
WHEN IS GREATER THAN
WHEN EXCEEDS
WHEN >=
WHEN IS GREATER THAN OR EQUAL TO
WHEN <>
WHEN IS UNEQUAL TO
```

In these condition statements, IS, THAN, and TO are optional words.

New IBM Compatibility Mode

The "-Cv" compiler option now takes an optional secondary argument to distinguish between OSVS compatibility and VSC2 compatibility. By itself, "-Cv" implies OSVS compatibility.

The two modes are very similar, except that in VSC2 compatibility mode, the following words are not reserved:

CURRENT-DATE
EXAMINE
TIME-OF-DAY
TRANSFORM

Note that CURRENT-DATE is a valid function in any compatibility mode.

NEXT SENTENCE Allowed in READ Statement

The NEXT SENTENCE phrase may now appear in a READ statement.

PERFORM Enhancements

An in-line PERFORM statement may now end with a period (“.”). An END-PERFORM is no longer required.

Converting Non-USAGE DISPLAY Numeric Items to USAGE DISPLAY

With a new “-Vd” video compiler option, non-USAGE DISPLAY numeric items are converted to USAGE DISPLAY before the screen display occurs. For compatibility with other COBOLs, this option is used in conjunction with the “-Ca” switch.

--FpRounding Compiler Option

A new “--FpRounding” compiler option simulates the behavior of other COBOL systems with regard to implied rounding when floating point is used in a math statement. For more information about this option, please refer to the “**New and Enhanced Compiler Switches**” section in this *Release Overview.*

SORT Elements of a Working-storage Table

The SORT statement can now be used to sort elements of a working-storage table. The syntax is:

```
SORT data-name-2[ON ASCENDING/DESCENDING KEY data-name-1 ... ]
   [ WITH DUPLICATES IN ORDER ]
   [ COLLATING SEQUENCE IS alphabet-name ]
```

SORT-MESSAGE Special Register

When compiling for IBM compatibility ("-Cv"), the following special register has been added: SORT-MESSAGE. Special Registers are variables that recognized without having been explicitly declared in the program. The implied declaration is:

```
01  SORT-MESSAGE PIC X(8) EXTERNAL.
```

1.5 Performance

Version 8.0 introduces a variety of performance enhancements to both the compiler and runtime, including improved computational performance for ACUCOBOL-GT arithmetic operations.

Improved Computational Performance

Historically, ACUCOBOL-GT performed the majority of its arithmetic operations using a 40-digit decimal format (68 digits if using the "-Dd31" compiler option). The advantages of this format are its high precision, compact code size, and ease of porting to many machine architectures. It also has a low conversion cost when working with USAGE DISPLAY or PACKED-DECIMAL (COMP-3) values.

Starting with Version 8.0, ACUCOBOL-GT uses binary math operations as its default for arithmetic operations. The primary purpose for this change is to improve overall application performance. The decimal operations remains in place to handle certain high-precision cases and to maintain compatibility with existing programs. ACUCOBOL-GT users can choose which operation they use: the binary operation for enhanced performance or the decimal format for strict compatibility with prior compilers.

Associated Compiler Options

You can control which math operations your program uses via compiler switches. The following two flags are new to Version 8.0:

--binaryMath	Forces the use of the binary math package if the target runtime version can support it. This option overrides any other options (can be abbreviated as "--bin").
--decimalMath	Forces the use of the decimal math package regardless of any other flags (can be abbreviated as "--dec").

Function Description

The Binary math affects the operation of the following verbs:

ADD
SUBTRACT
MULTIPLY
DIVIDE
COMPUTE

It also has a role in comparison operations with equations or in other situations where equations can appear, such as reference modification.

In general, intermediate results up to 9 digits can be stored in 32 bits and up to 18 digits in 64 bits. Many traditional COBOL math statements "fit" this model, with the key exceptions being when large numbers are multiplied together or an equation contains a division whose result is further used by the equation. The latter case may require carrying a large number of digits after the decimal point.

The binary math itself performs all of its operations on integer data. The compiler does scaling as needed to produce the proper arithmetic result.

The **cblutil** utility recognizes the new binary instructions and generally produces a direct CPU implementation of them. When it cannot produce a direct CPU implementation, **cblutil** calls small runtime helper functions. When decimal math is used, native code calls the runtime to perform the operation.

Application Performance

Generally, binary math is more efficient and can be faster than decimal math operations. This is particularly true if you compile for native code ("-n").

Binary math performs best when working on data that is already in binary format. Typically, the time for converting non-binary data items (typically DISPLAY or PACKED-DECIMAL) to binary exceeds the time for the actual math operation. For this reason, USAGE BINARY (or similar) data items are preferred for computational intensive aspects of a program. For example, the following data types are efficient with binary math operations:

BINARY
COMP
COMP-4
COMP-5
COMP-N
COMP-X
COMP-NS
COMP-XS
SIGNED-SHORT
UNSIGNED-SHORT

Note that COMP is treated as a decimal format under some compile options, so its efficiency is closer to USAGE DISPLAY than it is to USAGE BINARY. These options are "-D2" (treat COMP as COMP-2) and "-Cr" (RM/COBOL compatibility, which implies "-D2"). You should also note that SIGNED-INT, UNSIGNED-INT, SIGNED-LONG, UNSIGNED-LONG, and POINTER are variable size data items that are not treated as native binary items in the current implementation.

Operation results

For all well-defined cases, the computed results of the binary and decimal math operations should be the same. However, different results can be produced for undefined cases.

In COBOL, if the result of a math operation exceeds the storage of the destination item, the result is a "size error" and is undefined unless the SIZE ERROR phrase is specified. If the SIZE ERROR phrase is specified, the case is well-defined, and the two operations behave the same.

If the SIZE ERROR phrase is absent, the case is undefined and the two operations can produce different results. The differences are caused by what happens when the operation's internal variables overflow. Consider the following code:

```
77  DATA-1       PIC 99 BINARY, VALUE 95.
COMPUTE DATA-1 = DATA-1 + 10
```

Both the decimal and binary math operations would give the result of this computation as "105". However, "105" does not fit into a PIC 99 data item, so the result is truncated on the left and DATA-1 contains "05". This result is "undefined" under COBOL rules, but the practical effect of the implementation is that the program computes the remainder of the result after dividing by 100.

Now let's look at another case:

```
77  DATA-1       PIC S9(9) BINARY, VALUE 999999999.
COMPUTE DATA-1 = DATA-1 * 3
```

When run through the decimal operation, the binary DATA-1 is converted to an internal decimal format, the multiplication is performed, and the results are converted back to binary. The arithmetically correct result is "2999999997". However, because this result does not fit into a S9(9) item, the leading digit is truncated and DATA-1 receives "999999997" as its final result.

When the operation is performed by a 32-bit binary operation, the result is still "2999999997", but that value is x"B2D05DFD" in binary. This value is indistinguishable from the negative value "-29497299", which is stored in DATA-1. In this case, the overflow caused the resulting number to go negative instead of being truncated.

Both answers are "correct" under COBOL rules, because the case is undefined.

Cases that produce different results have a key characteristic. A result can be different if any of the computed intermediate values or the final value exceeds the capacity of the operation's internal storage. That capacity is either 32 or 64 bits for the binary operation or 40 digits for the decimal operation.

While most programs probably do not rely on the particular behavior of overflow results, a program may unintentionally do so. For this reason, users upgrading to Version 7.3 or later may wish to use the "--decimalMath" (or "--dec") compile option to avoid any unwanted effects in their programs.

AcuBench Support

Direct support for the two new math operations compiler options ("--bin" and "--dec") is available in the AcuBench Project Settings dialog Compiler tab, in the Miscellaneous catalog.

1.6 ACUCOBOL-GT Compiler

This section provides information on the following:

- **New and Enhanced Compiler Switches**
- **ACUCOBOL-GT Utilities**
- **New and Enhanced Library Routines**
- **Code Generation**

New and Enhanced Compiler Switches

Version 8.0 introduces the following compiler switches. Compiler options are documented in Chapter 1 of the *ACUCOBOL-GT User's Guide*.

--acceptrefresh	This option takes the most recent value of what was entered on the screen and then uses that value in subsequent ACCEPT statements. If you enter something in a screen section that goes to a variable, then MOVE something to that variable, the --acceptrefresh option will allow the subsequently MOVED value to be the basis of what is in the next ACCEPT statement.

--binaryMath	Causes the compiler to use binary math operations to handle arithmetic as long as the target runtime supports these operations. This can be abbreviated "--bin". Refer to the "**Improved Computational Performance**" section of this *Release Overview* for more details.
-Ce	The "-Ce" compiler option is enhanced. It currently allows you to specify a default source name extension that is used for both the main source file and its COPY files. In previous versions, ACUCOBOL-GT automatically adjusted the COPY file extension specified in "-Ce" for upper or lower case to match the case of the name it was being appended to. It then searched for the COPY file. In Version 8.0, if this first search fails, ACUCOBOL-GT matches the case of the extension specified in the "-Ce" compiler option to perform a second search for the COPY file. Note that this affects only operating systems with a case-dependent file system, such as UNIX.

-Cv	This option, which sets the compiler to its IBM DOS/VS compatibility mode has been enhanced. Since there are slight differences between IBM COBOL versions, "-Cv" now takes the following optional arguments: "-Cv=OSVS" specifies OSVS compatibility. "-Cv=VSC2" specifies VSC2 compatibility. "-Cv" by itself defaults to OSVS mode. The two modes are very similar, except that in VSC2 compatibility mode, the following words are not reserved: CURRENT-DATE EXAMINE TIME-OF-DAY TRANSFORM Note that CURRENT-DATE is a valid function in any compatibility mode.
--decimalMath	Causes the compiler to use decimal math operations to handle arithmetic, overriding other compiler flags. This can be abbreviated "--dec". Refer to the "**Improved Computational Performance**" section of this *Release Overview* for more details.

-e	The "-e" option has been enhanced so that when specifying a runtime error file name, you can use the following format specifiers: **"%p"** If the name contains the string **"**%p**"**, that string is replaced with the process ID (PID) of the runtime. **"%d"** If the name contains the string **"**%d**"**, that string is replaced with the current date in the form YYYYMMDD where YYYY is the year, MM month, and DD day. **"%t"** If the name contains the string **"**%t**"**, that string is replaced with the current time in the form HHMMSSTTT where HH is the hour, MM minute, SS second, and TTT milliseconds. **"%u"** If the name contains the string **"**%u**"**, that string is replaced with the username. **"%h"** If the name contains the string **"**%h**"**, that string is replaced with the hostname. Note that these specifiers may also be used in the file names configured with the ACU_MON_FILE and ACU_DUMP_FILE configuration variables.
--FpRounding	Simulates the behavior of other COBOL systems with regard to implied rounding when floating point is used in a math statement. This case-insensitive option is followed by an equals sign ("=") and one of the following: **OSVS** means that any math statement containing a floating-point data item as a sending item has "ROUNDED" implied for every receiving item. **VSC2** means that any math statement containing a floating-point data item as either a sending item or a receiving item has "ROUNDED" implied for every receiving item.

--noTrunc	All binary data types ignore their PICTURE when determining the largest value they can hold. However, the PICTURE is used when moving data from a binary number to a nonnumeric data item. The name of this option is similar to the name used by some other COBOL systems that behave this way.
--power	Produces code that is compatible with POWER and POWER2 processors, as well as PowerPC and later POWER series processors. This option allows you to use a wide range of machines, but it may affect performance. The existing "--ppc" option produces 32-bit PowerPC code that is compatible with POWER3, POWER4, and POWER5 machines. This code does not run correctly on POWER- or POWER2-based machines. These options are also available for the ACUCOBOL-GT **cblutil** utility.
-Vd	Causes the conversion of non-USAGE DISPLAY numeric items to USAGE DISPLAY before the screen display occurs.

-W1	Generates the following 01-level item warning: `USING parameter <name> is not an 01-level item` The ANSI COBOL standard requires that parameters passed to subprograms be 01-level items. ACUCOBOL-GT does not restrict them as such; however, there are valid reasons for restricting their use. For example, starting in version 7.0.0, the compiler can generate better code for certain moves and comparisons, based on the alignment of the underlying data types. When those data types are in LINKAGE, the alignment rules that the compiler assumes may not be valid. Making all of the passed parameters 01-level items ensures that the compiler's assumptions about alignment of the data items are valid. Note that the compiler can generate incorrect code when the assumptions are invalid. In such situations, it is possible to get a MAV at runtime.
-Wa	Generates the following alignment warning: `USING parameter <name> not aligned and may cause problems in the called subprogram` To be less restrictive, the compiler also includes an alignment warning. This is generated whenever a passed parameter is a group or is binary, and its alignment is not an even multiple of the alignment specified by the "-Da#" option.

ACUCOBOL-GT Utilities

Version 8.0 introduces a new utility program for performing remote preprocessing as well as enhancements to several existing utlities such as the runtime debugger, **vutil,** and **AcuSort**.

New Remote Preprocessing Utility - Boomerang

Version 8.0 introduces the new remote preprocessing utility called **Boomerang**. The **Boomerang** utility program includes client and server technologies that enable you to automatically transfer files to a remote server, invoke and perform preprocessing on that server, then return the preprocessed files to your client machine where additional compiling can occur. Many proprietary or third party preprocessors have machine-specific functions that require preprocessing to occur in their native environments. **Boomerang** makes accessing these types of preprocessors easier and more efficient.

With **Boomerang** you can:

- Send source files, COPY files, and user INCLUDE files from a Windows or UNIX/Linux client to a UNIX/Linux server.
- Invoke and run popular third-party preprocessors such as Pro*COBOL or invoke custom-built preprocessors.
- Have preprocessed output files, error files, and status returned to your client machine.
- Use **Boomerang** with the ACUCOBOL-GT compiler's “-Pg” option to perform multiple preprocessing steps.

See Chapter 3 of the *ACUCOBOL-GT User's Guide* for detailed instructions on **Boomerang** setup and usage.

Acucorp's Indexed File Record Editor (alfred)

As of Version 8.0, Acucorp's Indexed File Record Editor (**alfred**) is provided as a sample program and is located in the "sample" folder under "AcuGT". You can download detailed information on using **alfred** in PDF format from our Web site at the following address: http://www.acucorp.com/support/public/sample_programs/index.php.

.NET Definition Generator (NETDEFGEN)

The NETDEFGEN utility has been enhanced to recognize .NET assemblies version 1.1 and 2.0. NETDEFGEN's interface has been simplified and a Settings dialog has been added. The COPY files generated by NETDEFGEN have been enhanced to improve their readability.

Runtime Debugger

A qualifier is no longer needed in the debugger for multiple fields having the same name. In previous versions, when the display ("d") command was used to display the value of a Working-Storage data item in the debugger, and more than one data item with that name had been defined, a qualifier was required.

For example, if field-1 is defined in two groups in Working-Storage, as shown here:

```
01 group-1.
   05 field-1 pic xx.
01 group-2.
   05 field-1 pic xx.
```

The debugger will display the value of all occurrences of the field. The Runtime Debugger is documented in Chapter 3 of the *ACUCOBOL-GT User's Guide.*

Vision File Utility — vutil

The **vutil** "-load" and "-unload" commands now buffer the input file according to the A_SEQ_DEFAULT_BLOCK_SIZE configuration variable's value when that variable is set in the environment. Please refer to

the “New and Enhanced Configuration Variables” section in this *Release Overview* for more information. The **vutil** utility is documented in Chapter 3 of the *ACUCOBOL-GT User’s Guide.*

External Sort Utility — AcuSort

To improve performance, the file buffer layer of **AcuSort** has been enhanced to better manage its hash of buffer blocks and to add read-ahead capability.

Two new keywords have been added to the “take” file syntax: CHAR-ASCII and SIGN-ASCII. CHAR-ASCII puts **AcuSort** into ASCII character mode, and SIGN-ASCII puts **AcuSort** into ASCII sign mode. These two new keywords mirror the operation of the CHAR-EBCDIC and SIGN-EBCDIC and make it possible to switch back and forth between the different modes. This means you can put multiple SORT/MERGE operations in a single “take” file that uses different character sets or sign modes. **AcuSort** still defaults to CHAR-ASCII and SIGN-ASCII mode. The restriction that the SIGN and CHAR mode specifications must occur prior to all SORT/MERGE instructions no longer applies.

A new tracing bit has also been added to display the various modes. **AcuSort** is documented in Chapter 3 of the *ACUCOBOL-GT User’s Guide.*

New and Enhanced Library Routines

This section briefly describes new and enhanced library routines. Detailed descriptions of all library routines appear in Chapter I of the *ACUCOBOL-GT Appendices Manual.*

C$COPY

This routine has been enhanced to copy files remotely as text files instead of copying only in binary mode. This functionality applies when copying between UNIX and Windows systems, where text files have different line terminator characters.

C$EXITINFO

You can call a new C$EXITINFO library routine from an END procedure in the declaratives to return information when an END procedure is specified with the USE AT PROGRAM END statement (Format 6). Syntax information can be found in Appendix I, in Book 4, *ACUCOBOL-GT Appendices*.

C$FILEINFO

The "@[DISPLAY:]" notation is now allowed for the FILE-NAME parameter to C$FILEINFO.

C$GETERRORFILE and C$SETERRORFILE (New)

These routines return the name of the runtime error file as specified with the runtime "-e" command line option, or with a call to the C$SETERRORFILE routine. These routines provide a means for applications to embed identifying information in the runtime error file name. When specifying a runtime error file name, you can add identifying information to that name by using name format specifiers. For example, you can have information such as the process ID, date, time, username, or hostname automatically inserted in place of the corresponding format specifier you choose.

C$GETPID

This function, previously supported only on UNIX, is now supported on Windows and provides the same functionality as on UNIX.

C$JAVA

See **Section 1.2, "Interoperability Enhancements"** for a detailed description of the C$JAVA enhancements.

C$SETVARIANT

This library routine is now supported in thin client environments and includes all server operating systems supported by the Thin Client. Note that C$GETVARIANT is not supported at this time.

C$SOCKET

This routine has been enhanced with the addition of several new operation codes:

AGS-GETHOSTADDR - This is used to get the address of the local host in dotted notation (192.15.4.32) when C$SOCKET returns.

AGS-GETSOCKETPORT - This is useful when passing “0” as the port number when creating a socket, since a socket is created on some unknown port in that case. This function can then be used to determine the actual port being used.

C$XML

See **Section 1.2, “Interoperability Enhancements”** for a detailed description of the C$XML enhancements.

CBL_*_FILE (New)

Six new functions have been added to the runtime for reading and writing files that are compatible with other COBOL's routines. These routines include the following:

CBL_CLOSE_FILE
CBL_CREATE_FILE
CBL_FLUSH_FILE
CBL_OPEN_FILE
CBL_READ_FILE
CBL_WRITE_FILE

WIN$PRINTER

WINPRINT-UPDATE-PRINTERS

This new feature enables programmers to force the runtime to reload the internal printerlist so that any changes to that list (e.g., new printer added) will be detected by the COBOL program.

WINPRINT-SET-BKMODE

This new feature enables programmers to set the background mode for their printing. This is useful for adding watermark effects to prints.

WINPRINT-COLUMN-ALIGN-VERT

This operation code enables your COBOL application to support printing requests that contain fonts of alternate heights. This op-code tells the runtime to find the tallest font and align the print so that all the print on one line comes out without any overlap of another line of print.

WINPRINT-SETUP-EX

This operation code enables you to invoke the Microsoft SDK PrintDlgEX printer dialog. This printer dialog is strictly used for selecting a printer. This operation code is also supported in thin client environments.

W$PROGRESSDIALOG (New)

W$PROGRESSDIALOG provides access to the features of the Windows progress dialog box. This routine can be used to create a modal or modeless window containing a progress dialog and to set its title, animation, text lines, progress, and cancel message. The progress dialog can be configured to automatically estimate and display the time remaining until the operation completes.

Progress dialog windows are typically used when deleting, copying, moving, uploading, or downloading large files or a large number of files. They can also be used when performing a time-consuming operation that you want to allow the user to cancel at any time.

Code Generation

Segmented Program Code No Longer Produced

With Version 8.0, ACUCOBOL-GT moves to a “flat memory” model, so segmented program code is no longer supported. Programs may now have up to 16 MB of Procedure Division code per compilation unit for non-native code (an increase from 1 MB). Native code may exceed this amount as long as the non-native equivalent does not exceed 16 MB.

IF statements in programs with more than 64 KB of code may be more efficient in non-native code due to simpler address handling.

Note that the “-Zg” compiler option is ignored for Version 8.0 or later objects. You may need to remove “-Zg” from your compile scripts if you receive a new warning from the compiler.

Segmented Data Space No Longer Created

Beginning with Version 8.0, all data is stored in a single, 32-bit addressable data space, simplifying internal memory management. With this change, all program data is contiguous, which may be expected by non-COBOL subroutines or some external tools. As a result, the maximum data allowed per program is increased to 2 GB, as is the maximum data item size. The actual limit will be somewhat less, as the runtime needs some addressing space to run.

POWER versus PowerPC Code Generation

In previous versions, **cblutil** produced instructions that ran under both PowerPC and POWER CPU architectures when generating PowerPC native code. As of Version 8.0, this is no longer the case. Version 8.0 provides a new compiler option and enhances an existing option to generate code that is compatible with PowerPC and POWER CPU architectures. These options are available for the ACUCOBOL-GT **cblutil** utility.

A new “--power” option produces code that is compatible with POWER and POWER2 processors, as well as PowerPC and later POWER series processors. This option allows you to use a wide range of machines, but it can affect performance. For example, if you decide to use the “--power”

option in order to run on older AIX machines, you will not get some of the performance benefits if you also execute on new AIX machines on a POWER3, -4, or -5 architecture.

The existing "--ppc" option produces 32-bit PowerPC code that is compatible with POWER3, -4, or -5 machines. This code does not run correctly on POWER- or POWER2-based machines.

1.7 ACUCOBOL-GT Runtime

This section describes runtime enhancements as they relate to the following categories:

- **New and Enhanced Configuration Variables**
- **Graphical Technology Enhancements**

New and Enhanced Configuration Variables

In addition to the descriptions in this *Release Overview*, all ACUCOBOL-GT runtime configuration variables are described in complete detail in Chapter H of the *ACUCOBOL-GT Appendices Manual*.

A_SEQ_DEFAULT_BLOCK_SIZE

This configuration variable determines the size of the buffer to use when accessing a sequential file whose definition has no BLOCK CONTAINS clause. When set, A_SEQ_DEFAULT_BLOCK_SIZE specifies the size of the buffer in characters, rounded up to the nearest power of 2 that is greater than or equal to that value. The default value is "0", which sets the block size to 1 record. Note that this variable does not apply to print files or to files with names that start with a hyphen followed by "D" or "P".

You can set A_SEQ_DEFAULT_BLOCK_SIZE in the environment to allow the **vutil** "-load" command to buffer the input file according to the variable's value. The maximum buffer size is 1 GB. If this variable is not set, the default buffer block size is 4096 bytes. If it is set to "0", **vutil** "-load" performs record-based I/O on a sequential file.

ACCEPT_AUTO

This configuration variable applies only when running in HP COBOL compatibility mode (with the "-Cp" compiler option). The ACCEPT_AUTO configuration variable causes the runtime to treat all Format 1 ACCEPT statements as if the AUTO phrase is used, whether or not AUTO appears in the statement. Set this variable to "1" (on, true, yes) to enable this behavior. The default value is "0" (off, false, no).

ACU_USER_DIR

In previous versions, the ACUCOBOL configuration variable was used to specify the default location of a user debugger settings file. Because this variable may be needed for other reasons, a new ACU_USER_DIR variable is created for this purpose. When set, ACU_USER_DIR specifies the directory for the user's debugger settings (".adb") file. The default value is "NULL", which retains current runtime behavior.

ALLOW_FS_OVERRIDE

The runtime has been enhanced to enable the user to determine if the actual EXTFH return status will be returned or if the return status should be translated by the runtime. Previously, the runtime always translated the EXTFH return status. Set the new runtime configuration variable, ALLOW_FS_OVERRIDE. The default setting is "True" or "1" and will cause the actual EXTFH return status to be returned to the user. Setting this variable to "False" or "0" will cause the EXTFH return status to be translated by the runtime.

ANSI_OUTPUT_IN_DEBUG

This variable prevents a COBOL program that uses ANSI-style DISPLAY statements from interfering with the runtime debugger window. This variable accepts two possible values: "CANVAS" or "TERMINAL".

When set to "CANVAS" (the default setting) the runtime constructs a default canvas on which to place the ANSI output. This prevents the ANSI output from interfering with the debugger window. Note that if your COBOL program sends escape sequences to the terminal, this mode will cause those escape sequences to not have the intended result.

When set to "TERMINAL", the runtime will send ANSI output to the terminal, possibly interfering with the view of the debugger window. This is how the runtime behaved before the implementation of this new feature.

Note that this configuration variable must be set before the runtime initializes the terminal manager, which means you cannot set this variable from a COBOL program.

BTRV_USE_REPEAT_DUPS

There is a new configuration variable for use with the Btrieve interface that controls behavior related to duplicate keys. Previously, the Btrieve interface always created duplicate keys as LINKED duplicates (the default for BTRIEVE files). Set BTRV_USE_REPEAT_DUPS as desired. When set to the default value of “FALSE”, the Btrieve interface will create all duplicate keys as LINKED duplicates. When set to “TRUE”, the Btrieve interface will create all duplicate keys as REPEATING duplicates. See the Pervasive documentation for information on REPEATING duplicates and reasons for using them.

COBLPFORM

Previously, the compiler and runtime allowed printing only to C01 (channel 1), which was always assigned the value of line 1. With this enhancement, a new configuration variable called “COBLPFORM” enables you to define and print to all 12 print channels.

CGI_CLEARS_MISSING_VALUES

This configuration variable controls whether an ACCEPT sets the value of numeric data items to zero and nonnumeric items to spaces if the CGI variable is empty or does not exist in CGI input data. Set the CGI_CLEARS_MISSING_VALUES variable to “0” (off, false, no) to prohibit ACCEPT from clearing these data item values in this situation. The default value is “1” (on, true, yes).

EXTFH_KEEP_TRAILING_SPACES

A new EXTFH_KEEP_TRAILING_SPACES configuration variable allows you to preserve trailing spaces in line-sequential file records when using our EXTFH module with EXTSM. Set this variable to “1” to retain the trailing spaces, which is the runtime’s default behavior. With a default value of “0”, trailing spaces are removed.

GRID_NO_CELL_DRAG

A new style property called “NO-CELL-DRAG” enables you to prevent the user from dragging a cell in a GRID control. A new configuration variable called “GRID_NO_CELL_DRAG” makes NO-CELL-DRAG the default style for all grid controls, as opposed to specifying the NO-CELL-DRAG style individually for each grid control. To configure the NO-CELL-DRAG style as the default setting for all grid controls, set the GRID-NO-CELL-DRAG configuration variable to “1” (on, true, yes). The default value is “0” (off, false, no) and will enable the user to drag a cell in a GRID control unless you specify NO-CELL-DRAG style for that particular grid.

NESTED_AX_EVENTS

When an application dialog contains an ActiveX control that is assigned an event procedure, the event handler sometimes triggers additional ActiveX events. This variable determines whether or not the event procedure will be nested.

Set this variable to “1” (on, true, yes) if you want the event procedure to be nested. (This is the default). When NESTED_AX_EVENTS is set to “1”, the runtime allows events to trigger while it is processing other events. It is your responsibility to know when the event procedure is busy and reject events when this is the case, or to look for specific events and properly handle them.

For example, consider this code:

```
AX-EVENT.
MOVE 1 TO MY-LOOP.
PERFORM UNTIL MY-LOOP = 10
* Do some stuff
ADD 1 TO MY-LOOP
END-PERFORM
```

When NESTED-AX-EVENTS is set to “1”, it is possible that when your code is inside the event, possibly executing the loop for the fifth time, a new event triggers, setting MY-LOOP back to “1”. The perform loop could execute simultaneously in two threads on the same data, and the runtime

could crash. When you do not have reentrant events, MY-LOOP can only become “1” one time. This is the case when NESTED-AX-EVENTS are set to “0”.

Set NESTED_AX_EVENTs to “0” (off, false, no) if you do not want the event procedure to be nested. Be aware, however, that this option may cause you to lose certain events (typically events triggered by modifications made in the event procedure).

When NESTED_AX_EVENTS is set to “0”, once a program has entered an ActiveX control’s event procedure, new events are ignored. This prevents the runtime from executing the same code, at the same time. However, events that are imperative for the code execution may be refused.

Note: NESTED_AX_EVENTS applies only to the local runtime and has no effect in thin client scenarios.

PAGED_LIST_SCROLL_BAR

This variable controls the appearance of a scroll bar on paged list box controls in text-mode environments. PAGED_LIST_SCROLL_BAR can be set to “-1”, “0”, or “1”. The default value is “-1”. When set to “-1”, the vertical scroll bar is displayed to the right of a paged list box if the user interface configuration supports a mouse. Otherwise, the right border appears just like the left border.

PARAGRAPH_TRACE & SCREEN_TRACE

Two new runtime configuration variables have been introduced to allow for more tracing options. Set PARAGRAPH_TRACE to “1” (on, true, yes) to turn on paragraph tracing from within the configuration file or the COBOL program. This is equivalent to the debugger “tp” command. The COBOL program must be compiled with symbols (“-Gs”, or anything that implies that option) to get any error output.

Set SCREEN_TRACE to “1” (on, true, yes) to turn on screen tracing from within the configuration file or the COBOL program. This is equivalent to the debugger “ts” command.

PROFILE_TYPE

This new variable provides an optional method of profiling ACUCOBOL-GT on Windows called “COUNTER”. The counter method uses the debugger to perform counting and appears to provide the most accurate results in Windows environments. Set the PROFILE_TYPE configuration variable to either “ASYNCH” or “COUNTER”. When set to the default value of “ASYNCH”, the runtime performs profiling the way it historically has. When set to the value “COUNTER”, the runtime uses the new method of profiling. Note that your COBOL programs must be compiled with “-Gd” as well as “-Gs” options in order to take advantage of this new method.

This new method is also available on UNIX and can be used if profiling your COBOL results in a message similar to “profile timer expired”. This new method doesn't completely solve that problem, but it does substantially mitigate it.

QUIT_ON_FATAL_ERROR

This configuration variable applies only when running in HP COBOL compatibility mode (with the “-Cp” compiler option). A new QUIT_ON_FATAL_ERROR configuration variable causes the runtime to call the MPE QUIT intrinsic when an error occurs. The MPE job control word (JCW) is then set, and the MPE environment can determine if the program terminated with a fatal error. When set to “1” (on, true, yes), QUIT_ON_FATAL_ERROR calls the MPE QUIT intrinsic. The default setting is “0” (off, false, no).

USE_MPE_REDIRECTION

This configuration variable applies only when running in HP COBOL compatibility mode (with the “-Cp” compiler option). With the use of the USE_MPE_REDIRECTION configuration variable, input for an ACCEPT statement is read from the file specified by STDIN=, and output from a DISPLAY statement is written to the file specified by STDLIST= on the RUN command line. To enable this behavior, set USE_MPE_REDIRECTION to “1” (on, true, yes). The default value is “0” (off, false, no).

USE_SYSTEM_QSORT

This new variable instructs the runtime SORT routine to use the system qsort() function, rather than the built-in sort function. Set USE_SYSTEM_QSORT to "1" if you want to use the system qsort() function. The default value is "0" and results in the use of the built-in sort function.

Some systems have qsort() functions that perform better than the built-in function. Consider experimenting with this variable's settings to determine if this option yields better performance on your system. Pay particular attention to the number of comparisons done during the sort, which can be seen in the runtime trace output.

USE_WINSYSFILES

This variable specifies whether the runtime should recognize calls to modules with the extensions ".drv" and ".ocx" as well as those with the extension "dll". By default, it is set to "1" (on, true, yes).

For backwards compatibility, you can turn this feature off by setting it to "0" (off, false, no). Then, only calls to ".dll" files are supported.

Graphical Technology Enhancements

This section briefly describes new functionality for graphical user interface programming. You can find detailed programming instructions on these enhancements in the *ACUCOBOL-GT User Interface Programming* guide.

New NO-CELL-DRAG Style Property

A new style property called NO-CELL-DRAG enables you to prevent the user from dragging a cell in a GRID control. A new configuration variable called GRID_NO_CELL_DRAG makes NO-CELL-DRAG the default style for all grid controls, as opposed to specifying the NO-CELL-DRAG style individually for each grid control.

Wheelmouse Support for Paged Grid and Paged List Box

ACUCOBOL-GT now supports wheelmouse events for paged grids and paged list boxes. The new events (defined in acugui.def) are as follows:

- Paged GRID
- MSG-PAGED-NEXT-WHEEL, MSG-PAGED-PREV-WHEEL
- Paged LISTBOX
- NTF-PL-NEXT-WHEEL, NTF-PL-PREV-WHEEL

These are new exceptions/events and are handled in a similar way as the paged events for list boxes and grids. For details on using these events, refer to the *ACUCOBOL-GT User Interface Programming guide*, Chapters 3 and 6. Additionally, a sample program called "WheelEvent.cbl" demonstrates the use of these events and is included in the ACUCOBOL-GT samples directory installed as part of the development system.

Ability to Scale Bitmaps

New bitmap control properties enable you to apply automatic resizing of bitmaps to fit the area on a form where the image is to be displayed. Refer to the ACUCOBOL-GT User Interface Programming manual, Chapter 5, for coding instructions related to these new properties.

Auto-scroll Window for Tabbing to Controls Beyond Visible Bounds

In cases involving graphical windows with scroll bars, users can tab to a control outside of the visible area; and the runtime will scroll the window in order to make that control visible.

1.8 AcuBench

This section describes the major enhancements made to AcuBench. Noteworthy enhancements include support for AcuXUI, Boomerang, updated toolbar and menu icons, HTML report improvements, and new styles and configuration options.

NUM_COL_HEADINGS

The AcuBench Property sheet for the grid control now includes a Num Column Headings option to generate a value for the NUM_COL_HEADINGS property in the Screen Section.

The grid control's NUM_COL_HEADINGS property is documented in Book 2, section 5.11.2 of the ACUCOBOL-GT manual set.

Screen Designer Enhancements

Logo Screen Display Time

If you use the Screen Designer to design screens for a program, AcuBench gives you the option in the Program Properties interface to set one of those screens as a logo screen. In previous versions, this screen displayed for two seconds before the main program screen was displayed; there was no way to configure the amount of time the logo screen displayed.

Now, when you choose to display a logo screen, Version 8 gives you the option to determine how long the screen will be displayed before the main application window appears. You can choose a time between 1 and 300 seconds.

Reset Controls' Tab Order Improvement

In the AcuBench Screen Designer, it is possible to view and rearrange the tab order of screen elements by opening the Reset Controls' Tab Order interface. This screen presents a list of controls by tab number. In previous versions, when controls in the list had very long names, it was sometimes difficult to distinguish between controls with similar names.

In version 8.0, a horizontal scroll bar has been added to the control list in the Reset Controls' Tab Order dialog box. The scroll bar displays whenever control names too long to fit in the list box appear.

Parameter Entry

In previous versions, AcuBench users entered parameters in an entry field of the Execute with Parameters dialog box. In version 8.0, that entry field has been replaced with a combo box that can be expanded to display a list of up to twenty previous command execution parameters.

Drag and Drop

Control Labels

When you use the Drag and Drop interface to create certain types of controls (such as entry fields, check boxes, radio buttons, and so on), the control is associated with a label or given a title that matches the name of the data item associated with the control. For example, if you use Drag and Drop to create an entry field associated with the "client-id" field in an FD, a label with the title "client-id" is drawn next to the entry field.

Version 8.0 makes it possible to automate the process of creating more descriptive labels or titles for these controls. If you use the XFD tab of the File Designer to assign a Name directive to a field, you can elect to have Drag and Drop use that name, rather than the field name, when creating titles and labels. When this option is enabled, these assigned names will also appear in the Drag and Drop and Autoload interfaces, even for control types (such as list boxes) that are not generally associated with a label control or a title property.

Autoload Combo Boxes

In version 8.0, the Drag and Drop Autoload Dialog functionality has been added to combo boxes for character screens.

Screen-Level Property Sheet

The AcuBench screen-level property sheet now includes the option “No Close”, which adds the NO-CLOSE property to generated code used to DISPLAY the screen.

HTML Report Improvement

In previous versions, when you used the AcuBench Report Composer to create an HTML report, embedded spaces in titles or values for many controls were collapsed at run time. Label controls, for example, supported embedded spaces in literal values, but not variables. The entry field “Display Type” property allowed some options for preserving spaces in variables, but not affect literal values.

In version 8.0, a new option, “Collapse Spaces,” has been added to the Tools/Options/Report Writer/General interface. When this option is enabled (the default), the behavior described above is maintained. When this option is disabled, embedded spaces are preserved in the HTML code for the report and appear in the report output.

For entry fields, if the Tools/Options “Collapse Spaces” check box is marked, the Display Type property is used. If the check box is not marked, the values in the entry field are treated as though the “Keep Spaces” option were selected, regardless of the actual value set for the Display Type property.

Precompiling with Boomerang

You can take advantage of an AcuBench interface to the Boomerang client utility. The Boomerang utility sends source files to a remote server for preprocessing, then returns the precompiled source to the local machine for compiling.

For detailed information about the Boomerang utility, including server setup and configuration, see Chapter 3 of the *ACUCOBOL-GT User's Guide.*

AcuXUI Support

Version 8.0 provides support for programs that use AcuXUI technology through the following enhancements:

New Environment Variables

- The environment variable CLASSPATHDIR has been added to the Environment tab of the Project Settings interface. You should set this variable to the directory that contains the AcuXUI.jar file. It has a default value of %acudir%/bin.
- The environment variable XUIJAR has been added to the Project Setting interface. You should set this variable to the name of the jar file. It has a default value of AcuXUI.jar.
- The environment variable XUIPARAMS has been added to the Project Settings interface. This allows you to pass Java parameters as part of the Java command line.

USE AcuXUI Command

The "Use AcuXUI" command has been added to the AcuBench Build menu. When you select this option, AcuBench uses the AcuXUI environment variables set in the Project Settings interface to execute programs. This command operates very much like the Build menu's "Use Thin Client" option.

Note that you can invoke both the "Use AcuXUI" and "Use Thin Client" commands to run a remote program using AcuXUI.

If you are running locally, the command line that AcuBench uses to execute a program with AcuXUI is as follows:

```
javaw com.acucorp.acuxui.AcuXUI $XUIPARAMS --acucobolgt
   "Runtime Path" <runtime options> "Program Path"
```

If you are using the AcuConnect Thin Client, the command line that AcuBench uses is as follows:

```
javaw com.acucorp.acuxui.AcuXUI $XUIPARAMS -s <server>
   -p <port> -r <runtime options> Alias-name
```

New Navigation Option

In previous versions, if an AcuBench user had several Code Editor or graphical design windows open, it was sometimes difficult to navigate between the open documents or to discern quickly which documents were open, especially when the designer/editor windows were maximized.

In Version 8.0, each time you open a document, a tab associated with that document is added to a tab list at the top and bottom of the design area of the IDE. You can bring an open designer/editor window immediately to the foreground by clicking the tab associated with the document. This is similar to the design used by Microsoft Visual Studio .NET.

Print Improvement

Version 8.0 allows you to print the contents of the output window. Right-click anywhere inside the output window and select Print from the pop-up menu to send the contents of the selected output window tab to the default printer.

New “Delete From Disk” Functionality

In the Structural, File, and Data views of the Workspace window, the right-click pop-up menu includes a “Delete From Disk” option. In previous versions, when you chose this option, the selected file was deleted from the hard disk. Now, when you choose “Delete From Disk”, the corresponding file is sent to the Windows Recycle Bin.

New Thin Client Flags

Two Version 7.2 thin client runtime flags, "--wait" and "--restart", have been added to the AcuBench Project Settings interface. This integrates background debugging capability (as it relates to Transaction Processing systems) into AcuBench. See the ACUCOBOL-GT Interoperability Guide, section 9.8.2.3 for details on background debugging. An AcuBench dialog enables users to start and keep track of thin client processes that are running locally in --wait mode. Thin client processes can be started and stopped from this dialog.

cblutil Enhancements

Two Version 7.2 enhancements to **cblutil** have been added to the graphical cblutl32 interface, accessed through the AcuBench Tools menu. The utility's Library tab now includes the option to include a comment in the library (the "-c" flag); the Information tab includes an option to show extended library information (the "-x" flag).

Toolbar and Menu Icons Updated

The AcuBench toolbar buttons and menu icons, as well as other icons used throughout the interface, have received a facelift for Version 8. It's the functionality you're used to with a clean, contemporary look.

Compiler Warnings

Two new compile options ("-Wl" and "-Wa") have been added to the Compiler tab of the Project Settings interface. If "-Wl" is set, the compiler issues a warning when parameters passed to a called program are not 01-level items. If "-Wa" is set, the compiler issues an alignment warning when a binary or group data item is passed whose alignment is not an even multiple of the alignment specified by the "-Da#" option.

Wheelmouse Support

Version 8.0 includes wheelmouse support for the paged grid and list box controls. The Event tab of the AcuBench Property window has been enhanced to add code insertion points for paged-grid and list-box wheel events.

In addition, the automatically generated code for autoload paged list boxes has been updated to handle wheel events.

New Style and Configuration Variable

A new style and configuration variable were created to affect grid behavior at run time. AcuBench now includes support for these enhancements. When working with a grid control in the Screen Designer, you can specify the NO-CELL-DRAG style in the Property window. To set this style as the default, add the GRID-NO-CELL-DRAG configuration variable to your runtime configuration file.

New Compiler Options

Compiler options “-Pe” and “-Pw” have been added to Version 8.0.

1.9 AcuConnect

This section describes several Version 8.0 enhancements to AcuConnect and the Thin Client.

Thin Client: Redirecting Error Output

When using "-ee" to specify an error file, the runtime reopens stderr into the new file. This can be useful when you link routines that use stderr or start new child processes that write to stderr. **acurcl** now accepts "-ee" (and preserves that state) when the user specifies the command line to use. When a user specifies "-ee errorfile" as a runtime option, the runtime reopens stderr (instead of using its own file) and all image errors go to the runtime error file. No shell redirection is required.

Thin Client: New TEXT 36 Message Number

Message number "36" has been added to the TEXT configuration variable. This message number enables you to customize the message text that is returned when Thin Client does not receive a response from the server.

Thin Client: LABEL Titles Larger Than 1024 Characters

The length of a LABEL title under the thin client was limited to 1024 characters. This limit has been lifted.

Thin Client: Mouse Support for Character Screen Section Items

Thin client now includes support for the MOUSE configuration variables, selecting a character Screen Section item with the mouse, and changing the mouse shape when it passes over fields.

Thin Client: ACCEPT FROM TERMINAL-INFO Includes Client User ID

A new variable called CLIENT-USER-ID has been added to the TERMINAL-ABILITIES group item in the acucobol.def file. When IS-REMOTE is true, the CLIENT-USER-ID field is filled in with the client-side login name of the current user. This is the name the user enters when logging in to the client that is running acuthin. If it is not set, acuthin looks for the environment variable "USERNAME". If that is not set, then the literal "USER" is placed in this field.

Thin Client: @[DISPLAY:] Notation Now Allowed With C$FILEINFO

The "@[DISPLAY:]" notation is now allowed for the FILE-NAME parameter to C$FILEINFO.

acurcl Support for -k and -r Runtime Options

acurcl now supports the use of "-k" and "-r" runtime options in the alias file's command line. With the "-k" option, the script file named is searched first on the client machine and then on the server (as described in the W$KEYBUF library call).

With the "-r" option, the script file named is searched only on the server.

COBOL Virtual Machine to Link Data Remotely BY REFERENCE

With the implementation of the new COBOL Virtual Machine API, the runtime can execute remote COBOL objects without requiring a COBOL object to be executing on the client. For example, you might have an application developed in C, C++, Java, Delphi, or Visual Basic running on the client and using the CVM. AcuConnect can execute a COBOL object remotely and share data with the COBOL Virtual Machine.

1.10 AcuXDBC and AcuXDBC Server

Version 7.4 introduced the new AcuXDBC, which replaced AcuODBC. See the *AcuXDBC User's Guide* for detailed information on the functionality added to AcuXDBC including comparisons between the previous AcuODBC product line. Since Version 7.4 is a relatively new release with many new features, Version 8.0 enhancements focused on two main improvements that are described in this section.

Enhanced Data Format Information for XFD Files

Additional information is added to the Identification Section of Version 6 XFD files that indicates compiler options or program settings that may have been applied to the XFD. This includes fields that identify the following information:

- Sign compatibility (specified by the various "-Dc" compiler options)
- Maximum numeric digits (set by the "-Dd31" compiler option)
- Program period (decimal value of the character used as the program period)
- Program comma (decimal value of the character used as the program comma)

New SUBTABLE Directive

A new directive called SUBTABLE has been added to the compiler and XFD parsing routines. This directive modifies the way fields that appear in an OCCURS clause are processed. This directive instructs the XFD parsing routines not to append the occurrence number to the field name, but instead store just the base name along with the name of the subtable as written in the XFD file.

Note: AcuXDBC is the only product that currently makes use of this directive. Other products that use XFDs such as Acu4GL do not support this directive.

1.11 AcuSQL

Version 8.0 of AcuSQL received extensive enhancements to improve performance and increase functionality. This section describes these major enhancements.

ASQL_SQLSTATE_2000_ON_EOD Configuration Variable

Some embedded SQL products set SQLSTATE to the value of "02000" when end of data is reached on FETCH (SQLCODE = 100). A new configuration variable, ASQL_SQLSTATE_2000_ON_EOD, enables you to duplicate this behavior.

Level-78 Items

Level-78 items are now allowed in the pictures of alpha host variables. However, Level-78 items are still not allowed in the pictures of numeric variables.

ACUSQL_ODBC_CURSORS Configuration Variable

The Microsoft cursor library was removing the "FOR UPDATE" clause from SELECT statements, resulting in records not being locked. To prevent this, a new configuration variable, ACUSQL_ODBC_CURSORS, blocks the library from being loaded.

Automatic Line Breaks

In certain situations, AcuSQL now adds line breaks before and after various statements and following code generation to improve readability.

SQL Statement Names

To enable you to cache SQL statements at some later time, the pre-compiler now generates SQL statement names that previously had the form SQLISTM as SQLISTM#### where "####" is a sequential numeric value unique to a pre-compiled program.

Process SQL Within COPY Files

AcuSQL has been enhanced to enable the processing of ESQL located within COPY files. In addition, AcuSQL has been significantly modified through the addition and deletion of several precompiler and compiler options. Refer to section 3.2.2 of the *AcuSQL User's Guide* for details on preprocessing options.

Rewrite of AcuSQL for SQL Server

The AcuSQL runtime for SQL Server has been rewritten to remove the dependency on the proprietary DBLIB library from Microsoft. Instead, the runtime uses ODBC to communicate with SQL Server.

Support for INSERT Command on Group Items

In previous versions, it has been possible to SELECT a group item, but not to INSERT a group item. In Version 8, the pre-compiler now supports the use of the INSERT command on a group item.

Brackets in Column Names

Some SQL engines allow brackets around column names as part of the column, for example [TestDate]. In Version 8.0, the pre-compiler has been modified to accept brackets in column names.

FETCH Position can be a Host Variable

You can now specify a row position through a host variable when fetching from a cursor that uses the ABSOLUTE or RELATIVE extensions. In previous versions, the specified row position had to be a constant.

Support for Rowset Functions

The AcuSQL preprocessor has been enhanced to allow for rowset functions. Microsoft SQL Server includes several built-in rowset functions, and also allows users to define their own rowset functions.

1.12 Acu4GL

This section describes enhancements made to the various Acu4GL interfaces.

Configuration Option to set the Oracle Sort Order to Binary

The order of returned records can be changed depending on the native language's sort order if the correct Oracle configuration options are not set. A new configuration variable called A_ORA_NLS_SORT is used to force the sort order of the returned records to be in binary order.

Set this new variable, A_ORA_NLS_SORT to "1" to turn off the default binary order and use the native language sort instead. The default is "0", which means that the sort order will not be the NLS sort order and will, instead, be binary.

Note that A_ORA_NLS_SORT must be set before the first database file is accessed.

COBOL Triggers

There is a new XFD directive for instructing Acu4GL that a COBOL trigger is to be executed. The directive is called COBOL-TRIGGER. Please see Chapter 4 in the *ACUCOBOL-GT User's Guide* for detailed programming instructions. Triggers are programs that can be called to execute before and after every READ, WRITE, REWRITE and DELETE operation.

DB_MAP, USEDIR_LEVEL Configuration Variables

Acu4GL has new configuration variables for mapping directory specifications to databases. These variables allow users to use FILE-PREFIX as a list of directories, each of which maps to a different database. There is also a new variable for mapping filenames (with directory

information) to table names. This allows you to use files of the same name, which would normally go into separate directories, as separate names in a single database.

Currently, these variables are available with the MSSQL and Sybase Acu4GL interfaces.

Rewrite of Acu4GL for Microsoft SQL Server

Acu4GL for Microsoft SQL Server (MSSQL) has been rewritten to remove the dependency on the proprietary DBLIB library from Microsoft. Instead, the runtime uses ODBC to communicate with MSSQL Server.

In the process of the rewrite, several configuration variables that are no longer needed were removed. The interface should be a “drop-in” replacement for the old version, and performance should be improved.

Configuration variables that are no longer supported are the following:

A_MSSQL_CHECK_DELETE_SP
A_MSSQL_CHECK_INSERT_SP
A_MSSQL_CHECK_READ_SP
A_MSSQL_CHECK_UPDATE_SP
A_MSSQL_EXTRA_PROC
A_MSSQL_FORCED_INDEX
A_MSSQL_NO_DBCLOSE

New configuration variables are the following:

A_MSSQL_APPROLE_NAME
A_MSSQL_APPROLE_PASSWD

These variables can be used to have Acu4GL for MSSQL use approles. Before connecting to a database, set these variables to the name of the role and to the password for that role. The runtime will then be set to use that approle. For more information on approles, please see the MSSQL Server documentation.

Enhanced transaction processing for Acu4GL/ODBC and Acu4GL/DB2

A new set of configuration variables have been added to allow Acu4GL/ODBC and Acu4GL/DB2 to function more like Acu4GL/Oracle when working with the configuration variable 4GL_COMMIT_COUNT.

These variables are:

A_DB2_ALTERNATE_COMMIT_LOGIC
A_ODBC_ALTERNATE_COMMIT_LOGIC

The primary purpose of the variable 4GL_COMMIT_COUNT is to provide for applications that must communicate with a transaction-oriented database, but have not coded transactions into their COBOL application.

Now, when determining whether or not to issue a commit, the current transaction status is not checked. For example, if the configuration file has the setting:

4GL_COMMIT_COUNT=1

every WRITE is immediately committed. This prevents an application from rolling back any of the WRITE operations.

When A_ODBC_ALTERNATE_COMMIT_LOGIC=1 is set for Acu4gl/ODBC or A_DB2_ALTERNATE_COMMIT_LOGIC is set for Acu4GL/DB2, the value of 4GL_COMMIT_COUNT is checked after each WRITE, REWRITE, DELETE, or UNLOCK operation. A commit caused by the setting of 4GL_COMMIT_COUNT is issued only if the runtime is not currently in a transaction. This functionality more closely matches the behavior found in Acu4GL/Oracle.

Configurable VARCHAR handling for Acu4GL Oracle/OCI

A new configuration variable, A_ORA_VARCHAR_MODE, allows you to include or leave out spaces in fields that are passed to Oracle variable-length fields. This configuration variable is particularly useful if your application

interfaces with international databases and contains variable-length character fields in which a character is composed of multiple bytes with trailing spaces significant to the formation of a character.

See Section C.4 of the *Acu4GL User's Guide* for details on using and setting this variable.

1.13 Technical Services

You can reach Acucorp Technical Services in the United States Monday through Friday from 6:00 a.m. to 5:00 p.m. Pacific time, excluding holidays. You can also raise and manage product issues online and follow the progress of the issue or post additional information directly through the website. Following is our contact information:

Phone:	+1 858.689.4502
Phone:	800.399.7220 (in the USA and Canada)
Fax:	+1 858.689.4552
E-mail:	**support@microfocus.com**
Online:	**http://supportline.microfocus.com**

For worldwide technical support information, please visit **http://supportline.microfocus.com**.