Release Overview

extend® Product Suite

Version 8.1

Micro Focus
9920 Pacific Heights Blvd.
San Diego, CA 92121
858.795.1900

E-01-RO-080901-Release Overview-8.1

Contents

Chapter 1: Overview of *extend*® Version 8.1

1 Overview of *extend*® Version 8.1

Key Topics

1.1 Version 8.1 Major Enhancement Summary

Micro Focus is pleased to announce Version 8.1 of ***extend***®, the next release in our Interoperability Series. Version 8.1 provides the ability to leverage Windows XP and Vista control styles. In addition to several other major enhancements, much attention was given to fine tuning many of the Version 8 enhancements, resulting in greater precision, stability, and performance. This section introduces you to the following top Version 8.1 features:

Windows XP and Vista Visual Styles
Windows Server 2008 Support
AcuXUI Enhancements
Conditional Compiling ($IF, $ELSE, $DISPLAY)
New Syntax Support
Instantiating Non-GUI COM Objects
.NET Control Loading Enhancements
Acu4GL Table Searches and Open (DB2 and ODBC)
AcuBench Report Designer and Bitmap Scaling

Windows XP and Vista Visual Styles

A new runtime enhancement enables your applications to match the Windows XP or Vista theme when chosen by the user via the Windows display panel. No recompiling or changes to your source code should be required. You gain a more modern look and feel to applications that will also match the user's preferred Windows setting. This feature is more technically described in section 1.4, "Runtime Enhancements."

Windows Server 2008 Support

The extend product line (as applicable) is now supported on Windows Server 2008. Please note the following when working with AcuConnect and AcuServer:

System Security

With AcuConnect and AcuServer, you have the option to use your operating system security rather than AcuConnect or AcuServer system security. It is recommend that you use the operating system security and is essentially required when using Windows 2008. This is mainly due to a new security layer called User Account Control (UAC). Consult Windows Help for information on how UAC impacts file and program access.

By setting the SECURITY_METHOD configuration variable on both the client and the server, you can override the server access file and use the full range of native security features on files and directories on the server instead. Consult the Configuration section of either the AcuConnect or AcuServer manual for details on setting operating system security.

Accessing Utility Programs

To use various ***extend*** administrator utilities such as the AcuConnect and AcuServer control panels on Windows 2008 where User Access Contol UAC is turned on (as it is by default), any user must choose “Run as Administrator” in order to use the various utilities. UAC can be turned off, in which case the user must merely be a member of the administrators group in order to fully operate AcuServer. Consult Windows 2008 Help for more information on UAC and how it may affect program and file access.

AcuXUI Enhancements

First introduced in Version 8.0, AcuXUI is a multi-platform user interface engine for running graphical ACUCOBOL-GT® programs. With AcuXUI, you execute graphical COBOL programs from a Java command line. This lets you run the same graphical program on any machine that supports the Java Runtime Environment — including UNIX, Linux, Macintosh, and Windows. Programmed accordingly, you can run the same program, without modification, on all of these platforms.

Version 8.1 includes new support for character screens, grid controls, and tab controls. Many other refinements were made to existing supported controls, events, applets, and bitmap handling. Overall, these refinements result in GUIs that look "cleaner" and behave more predictably in the target environment.

AcuXUI.jar is now signed with a digital ID from Verisign. This digital ID means that it is no longer necessary to modify the local "java.policy" file, as was the case in version 8.0.

Conditional Compiling ($IF, $ELSE, $DISPLAY)

The compiler now supports conditional compilation through the use of special constructs in the COBOL source file and by accepting command-line arguments that turn on compiler directives and set constants to values. This enables you to define conditions and whether or not lines/sections of code compile based on those conditions. One potential use of conditional compiling is that you can maintain a single source stream, but compile it differently for different run-time (object code) variations. The same source code can have multiple branches of source code streams. By setting conditions you only include the "relevant" source code in the compilation.

New Syntax Support

Several new COBOL statements and new parameters to existing constructs makes more of the ANSI standard (as well as syntax from other COBOL dialects) available to you. This includes support or enhancements related to the following items:

- EXHIBIT verb
- SORT file descriptor
- Zero-based relative key number (ACTUAL KEY)
- LENGTH OF for returning a single table element

Each of these items are discussed in detail in Section 1.2, "Compiler Enhancements" of this release overview.

Instantiating Non-GUI COM Objects

When running in thin client mode, you can specify that a COM object is to be wholly instantiated on the specified server where the COM object is registered, where all the work is done, and where all resources reside. This can lead to better performance by carrying out operations or functions on the host server rather than the client. It can also mean fewer network transmissions between host and client.

.NET Control Loading Enhancements

Various code improvements were made to the "wrunnet.dll" module. While these changes are invisible to the user, in some circumstances, you may observe improved application performance. In particular, improvements in screen and control load time should be observable in cases where multiple controls are loaded.

Additionally, the error reporting messages for the LoadEntity and GetContainer functions have been improved to include why the failures occurred, which leads to less time spent on troubleshooting.

Acu4GL Table Searches and Open (DB2 and ODBC)

A new and more efficient internal API is now available to use when searching and opening database tables. Using this API can result in faster processing of queries that search and open database tables — the larger the database, the more noticeable the performance gains become.

AcuBench Report Designer and Bitmap Scaling

Enhancements to the report designer includes automatic generation of PERFORM loops via the property stylesheet, as opposed to the COBOL source file. This results in reports that can be entirely generated from the PSF file.

AcuBench now supports the BITMAP-SCALE property introduced in Version 8.0. You can enable/disable automatic bitmap scaling via the property window. AcuBench automatically generates and adds BITMAP-SCALE code to the program's screen section.

1.1.1 Recap of Version 8.0 Enhancements

To better understand the value and significance of Version 8.1 you should note that many internal refinements made in Version 8.1 relate to improving the performance or building upon the many features added in the Version 8.0 release. For those unfamiliar with the Version 8 series of ***extend*** products, a summary of this major release is provided in this section. To obtain detailed information on this release, download the Version 8.0 Release Overview from the Micro Focus website: http://supportline.microfocus.com/productdoc.asp

AcuXUI

This new product enables you to execute graphical COBOL programs from a Java command line. This lets you run the same graphical program on virtually any machine that supports the Java Runtime Environment — including UNIX, Linux, Macintosh, and Windows. Programmed accordingly, you can run the same program, without modification, on all of these platforms.

Boomerang Remote Preprocessing

This new utility provides client and server technologies that enable you to automatically transfer files to a remote server, invoke and perform preprocessing on that server, and 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.

Runtime Performance Enhancements

- ACUCOBOL-GT now uses 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.

Java, ActiveX, .NET, XML Enhancements

- Added support for DRV and OCX files
- Java exception handling
- Support for Nested ActiveX and COM Event Procedures
- Support for .NET assemblies version 1.1 and 2.0
- New C$XML op-codes for parsing Encoding, Stand-alone, and Processing Instructions

Data Access Enhancements

- Compiler options and program settings added to XFD Files
- Large (>2GB) Transaction Log Enhancement for Vision

Supporting Product Enhancements

- AcuBench® — Remote debugging of IBM CICS applications, support for AcuXUI and Boomerang, 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

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 Compiler Enhancements

This section describes the following enhancements:

Conditional Compiling ($IF, $ELSE, $DISPLAY)
Syntax Support Improvements
New XFD DATE Format Specifiers

Conditional Compiling ($IF, $ELSE, $DISPLAY)

The compiler now supports conditional compilation through the use of special constructs in the COBOL source file and by accepting command-line arguments that turn on compiler directives and set constants to values. This enables you to define conditions and whether or not lines/sections of code compile based on those conditions.

Conditional compiling enables you to maintain a single source stream but compile it differently for different runtime (object code) variations. The same source code can have multiple branches of source code streams. By setting conditions, you include only the "relevant" source code in the compilation.

Conditional compilation is controlled by $IF, $ELSE, $END constructs, which behave in a similar way to the COBOL IF construct. Conditional compilation also supplies the $DISPLAY statement, which is used to display a message during compilation or to include a version number in the object

file. The $SET statement is used to define compiler directives for use in $IF statements. Refer to the *ACUCOBOL-GT Reference Manual*, section 2.5 "Conditional Compilation," for syntax and general rules for these constructs.

The "/" compiler option is used to turn on directives and specify constants. See the *ACUCOBOL-GT Users Guide*, section 2.2.15, for details.

Syntax Support Improvements

LABEL, CONFIGURATION SECTION

The compiler has been modified to be more compatible with other COBOL compilers in the following ways:

In a SORT file descriptor (SD), the following phrases will be ignored by the compiler, instead of causing errors:

LABEL
BLOCK
LINAGE
EXTERNAL.

These phrases are considered "cautions"; to see the warnings, compile with "-a".

In the CONFIGURATION SECTION, the SOURCE-COMPUTER, OBJECT-COMPUTER, and SPECIAL-NAMES paragraphs can now occur in any order.

EXHIBIT verb support

The compiler now supports the EXHIBIT verb when a program is compiled in IBM OSVS compatibility mode (-Cv or -Cv=OSVS).

The EXHIBIT verb has the following syntax:

```
EXHIBIT [NAMED] [CHANGED] {literal | variable} ...
```

- If neither NAMED nor CHANGED is used, each literal and variable value is displayed.

- If only NAMED is used, each literal is displayed, and each variable is displayed. Variables are preceded by "*variable-name*=" (where "variable-name" is replaced with the name of the variable in the EXHIBIT statement).

- If only CHANGED is used, each literal is displayed, and each variable is displayed if its value is different from the last time this EXHIBIT verb was executed.

- If both NAMED and CHANGED are used, each literal is displayed, and each variable is displayed if its value is different from the last time this EXHIBIT verb was executed. In addition, each variable is preceded by "*variable-name*=" (where "variable-name" is replaced with the name of the variable in the EXHIBIT statement).

As a compatibility issue, we recommend that you modify your source code to use actual DISPLAY statements, and that you not add new EXHIBIT statements to your COBOL program.

LENGTH OF with single table elements

The LENGTH OF construct works differently in ACUCOBOL-GT than in other COBOL compilers (such as IBM Enterprise COBOL) when the data item used is a table. In this case, ACUCOBOL-GT has always returned the size of the entire table, while other compilers return the size of a single element of the table. For example, consider the following:

```
01  my-data.
    03  my-table occurs 20 times.
        05  my-element-1   pic x(10).
05  my-element-2   pic 99.
MOVE LENGTH OF my-element-1 TO data-size.
MOVE LENGTH OF my-table TO data-size.
MOVE LENGTH OF my-table(1) TO data-size.
```

All compilers treat the first MOVE as MOVE 10 TO data-size, and all known compilers treat the third MOVE as MOVE 12 TO data-size. The second MOVE, however, is treated differently by different compilers. ACUCOBOL-GT treats the second MOVE as MOVE 240 TO data-size, while IBM Enterprise COBOL treats the second MOVE as MOVE 12 TO data-size. (Note that 240 equals 20 * 12.)

When using IBM compatibility mode, the ACUCOBOL-GT compiler now treats LENGTH OF in the same way as IBM Enterprise COBOL. IBM compatibility mode is used by adding "-Cv" to the compile line.

Zero-based Relative Key Number (ACTUAL KEY)

In IBM DOS/VS ("-Cv") and HP3000 ("-cp") compatibility modes, when compiling for object versions 8.1 and greater, the compiler now accepts the ACTUAL KEY feature for relative files opened in RANDOM access mode. For such files, the relative key numbers will be zero-based, rather than one-based, as with RELATIVE KEY. For example, if you have a relative file with a fixed record length of 3 bytes and a relative file with the following contents:

AAABBBCCC

the record keys for the different modes are:

```
     RELATIVE  ACTUAL
AAA:    1         0
BBB:    2         1
CCC:    3         2
```

New XFD DATE Format Specifiers

The XFD DATE directive now has two new format specifiers for users who have worked to get eight digits of date data in six characters.

There are two new date format characters with very specific requirements for use.

Instead of YYYY or YY, you can now specify FY to mean that the first character of the year specifies the decade instead of the "tens" year. The decade can be a character between space (" ") and "I" (inclusive). For the characters "0" through "9" to be treated as decades in the 20th century, the decade characters have the following meaning:

	00	10	20	30	40	50	60	70	80	90
1700					spc	!	"	#	$	%
1800	&	'	(	)	*	+	,	-	.	/
1900	0	1	2	3	4	5	6	7	8	9
2000	:	;	<	=	>	?	@	A	B	C
2100	D	E	F	G	H	I				

This means that a date of ?70210 is converted to 20570210, or February 10, 2057. The range of valid dates is “00101” (Jan 1, 1740) through “I91231” (Dec 31, 2159).

Instead of YYYY or YY, you can also specify RY to mean that the first character of the year specifies the decade and that the entire date is to be 9s complement (to be able to sort dates in reverse order). Note that the entire date and time is treated as a 9s complement number in this case. The decade characters have the following meaning:

	00	10	20	30	40	50	60	70	80	90
1700					I	H	G	F	E	D
1800	C	B	A	@	?	>	=	<	;	:
1900	9	8	7	6	5	4	3	2	1	0
2000	/	.	-	,	+	*	)	(	'	&
2100	%	$	#	"	!	spc				

The date 20570210 is now specified with *29789. The range of valid dates is “I99898” (Jan 1, 1740) through “08768” (Dec 31, 2159), the same valid range as when using F instead of R.

While using F and R before month, day, hour, or any other format specifier does not generate a compile error for the XFD, the results are undefined at runtime.

Note that if you use these format specifiers with Acu4GL, the actual date is written to the database, not the encoded date. That means that the R specifier is not very useful in this scenario (you won't be able to read forward through a file in reverse date order). This format specifier is more useful with AcuXDBC, that is, when the data is Vision.

1.3 Debugging and Utilities Enhancements

Version 8.1 includes enhancements to:

Debugging
AcuSort
cblutil
vutil

Debugging

Debugging with xterm

Some users who want to debug with an xterm don't actually want to debug with the xterm executable, because it doesn't have some of the abilities they need (such as displaying non-ASCII characters). This enhancement allows the user to specify the executable used to show the debugger on UNIX.

There is a new configuration variable, XTERM_PROGRAM. Its default value is "xterm", but it can be set to any compatible program. The runtime executes this program when it tries to create the program for background debugging.

Note that the runtime passes some arguments to this program, so this program must be able to execute with those arguments. These arguments are:

```
-title "title of the window"
-Sccn
-display Xserver-name
```

The "-Sccn" option allows the program to be used as the input and output channel for the runtime, and is absolutely required. Without this option, the program will not know to display data from the runtime.

Debugging with Thin Client and Transaction Servers

When using the thin client to debug programs with transaction servers like Tuxedo if the thin client was not available, the runtime returned COBOL_FATAL_ERROR to the transaction server. This normally required the transaction server to be restarted. The runtime now returns COBOL_NONFATAL_ERROR in this case, which means the transaction server does not have to be restarted.

Compile Options in the ADR

Starting with Version 7.2, the compile options used to build a COBOL object are embedded in that object (see ECN 2944). Those compile options are now included in the ABEND Diagnostic Report for each program in the call stack of each thread. Note that the options may be printed multiple times for a single COBOL program, if that program is active in multiple threads.

Non-debugging Programs

You can now compile a COBOL program so that it will not stop and enter the debugger. This can be useful if you distribute COBOL objects and allow your users to create their own COBOL programs, but you don't want them to be able to debug your programs.

There is a new compile option, "-Gz", which implements this feature. This compile option is not compatible with any other debugger option. If other debug options are used on the command line, the last debugger option will take precedence. In other words, specifying "-Ga -Gz" will result in an object that will never stop in the debugger, while specifying "-Gz -Ga" will result in an object with full debugging symbols.

This option is available only if targeting an 8.1 runtime or later. This is because using this option sets an invalid value for the debugger page of the COBOL object, and if the runtime is not able to detect that invalid value (the 8.1 and later runtimes can), the runtime could fail. Specifying "-Znn" for any *nn* <= 80 will turn off this flag.

The runtime will at times stop and enter the debugger because of program errors, such as reference modification range errors, perform stack overflows, and so on. The only time the runtime will stop and enter the debugger when

executing a COBOL program compiled with “-Gz” is when the runtime encounters a format 2 STOP statement (STOP literal), which is meant to break into the debugger.

AcuSort

Filename Parsing Improved

There is also now a way to specify file names containing whitespace characters: surround the filename containing whitespace characters with double quotes (" "). If the filename contains double-quote characters, specify these by doubling the double-quote characters ("" "").

Examples:

```
filename: Work File
notation: "Work File"

filename: Embedded"Quote
notation: "Embedded""Quote"

filename: Quotes "and" Whitespace
notation: "Quotes ""and"" Whitespace"
```

Additionally, you can now sort/merge a FILE with the same name as a keyword (such as “merge” or “sort”).

cblutil

Improved Native-code Numeric Comparisons

Native code generation for certain types of numeric data has been improved to be more efficient.

vutil

Input File Locking for Better Performance

A new option, "-l", for "**vutil** -unload" places a read lock on the input Vision file. This improves performance, because the records can be read without needing to place and release locks on the individual records.

Option to Skip Duplicate Records

"**vutil** -load" now has a sub-option, "-s", which indicates that duplicate records should be skipped rather than written to a file. When this option is used, any duplicate records found while loading the indexed file are discarded. This new option is incompatible with the "**vutil** -load -r" option.

1.4 Runtime Enhancements

This section describes the following runtime enhancements:

Applying Windows XP and Vista Control Styles
.NET Control Loading Enhancements
.NET Error Handling Improvements
C Function Parameters Increase
Global C Runtime Variable "ERRNO"
Instantiating Non-GUI COM Objects
Key Letters in Character Mode
Linkage Runtime Optimization
Preventing acushare Runtime License Error Message

Applying Windows XP and Vista Control Styles

Controls can have different visual styling or themes on certain Windows operating systems such as XP and Vista. ACUCOBOL-GT applications can employ the latest styling that is set on the workstation (via the Windows Control Panel Display options). You can modify the various styling control

through the new runtime configuration variable WIN32_NATIVECTLS. Refer to the *ACUCOBOL-GT Appendices Guide*, Appendix H, for details on this variable.

Set the WIN32_NATIVECTLS variable as desired. When set to the value of "1" (on, true, yes), the application will display the current control styling available on that operating system, for example, the XP look and feel on the Windows XP O/S or the Vista look and feel on the Windows Vista O/S.

The default setting is "0" (off, false, no) which prevents the runtime from using the latest Windows' control styling, and forces the "gray chiseled" Windows look.

.NET Control Loading Enhancements

Various code improvements were made to the "wrunnet.dll" module. While these changes are invisible to the user, in some circumstances, you may observe improved application performance. In particular, improvements in screen and control load time should be observable in cases where multiple controls are loaded.

.NET Error Handling Improvements

The error reporting messages for the LoadEntity and GetContainer functions have been improved to include why the failures occurred. This has the obvious benefit of reducing the amount of time spent on troubleshooting these errors.

Various code improvements were made to the "wrunnet.dll" module. While these changes are invisible to the user, in some circumstances, you may observe improved application performance.

C Function Parameters Increase

The maximum number of parameters that can be passed to a direct-style C function has been increased from 20 to 30.

Global C Runtime Variable "ERRNO"

It is much easier to call external C system functions without necessarily relinking the runtime. However, it is not so easy to get any error information from those system functions (using the external variable ERRNO), especially in a machine-independent way.

The runtime now exports the ERRNO variable, so that COBOL programs can reference it easily, without requiring a relink.

ERRNO should be defined as:

```
77  ERRNO EXTERNAL SIGNED-INT.
```

Consult your preferred C manual for information on using ERRNO.

Instantiating Non-GUI COM Objects

When running in thin client mode, you can specify that a COM object is to be wholly instantiated on the specified server where the COM object is registered, where all the work is done, and where all resources reside. This can lead to better performance by carrying out operations or functions on the host server rather than the client. It can also mean fewer network transmissions between host and client.

Key Letters in Character Mode

In some situations Key letter behavior is configurable. Some developers prefer to use the KEYSTROKE EDIT=ALT method of requiring users to press an ALT key to activate key letters over the default behavior of moving to a new control without that ALT key.

There is a new configuration variable, NO-BARE-KEY-LETTERS, which, when set to the default value of FALSE, allows users to press key letters of controls when they accept controls that don't allow character input (such as push-buttons and check boxes). When set to TRUE, users are required to press the key set by the KEYSTROKE setting EDIT=ALT before pressing the key letter of a control to move to that control.

Linkage Runtime Optimization

The runtime now performs address optimizations on each Linkage item individually, as opposed to optimizing depending on the group as a whole. The main effect of this enhancement is to improve CALL performance in as many cases/scenarios as possible.

Preventing acushare Runtime License Error Message

This enhancement provides a new configuration variable (LICENSE_ERROR_MESSAGE_BOX) for the runtime to prevent acushare licensing errors from appearing in a message box that requires a response from the user. When the variable is specified, such error messages go to the error output instead (stderr or an error file, if one is specified).

1.4.1 Library Routines — New and Enhanced

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

CBL_SUBSYSTEM (New)
C$XML PARSE-STRING and Pointers
-Q Windows Printing Improvements
WIN$PRINTER — Specifying Colors for Columns and Negative Numbers

CBL_SUBSYSTEM (New)

This new library routine can be used to define a COBOL subsystem that allows cancelling of all COBOL programs in the subsystem with a single statement.

This library routine takes two parameters. The first parameter is an op-code, and the second parameter depends on the op-code used. Refer to the *ACUCOBOL-GT Appendices* guide, Appendix I, for details.

C$XML PARSE-STRING and Pointers

C$XML can now parse strings passed by reference. This improves interoperability, as you can now access XML data in cases where you have only a pointer to the source, such as a pointer returned from some other function or one obtained from a Web site.

Note that strings passed by reference MUST be terminated with low-values so that the runtime can determine the length of the string passed. Not terminating with low-values will result in undefined behavior.

-Q Windows Printing Improvements

The -Q configuration feature has been enhanced with several new options. You may now specify paper format, paper tray, duplex printing, collate printing, color print, using current printer, using windows default printer, and reset printer. There are now many more features to enhance the printing capabilities of your applications so that they more closely resemble "native" Windows programs.

WIN$PRINTER — Specifying Colors for Columns and Negative Numbers

You can now specify a font color for all values appearing in a single printed column. You can also specify the font color of any negative numbers that appear in a column of print. These enhancements enable you to provide special emphasis to certain parts of print jobs through the use of color.

The WINPRINT-COLUMN operation code of WIN$PRINTER has two new members, which are defined in winprint.def. These members are WINPRINT-COL-FONTCOLOR and WINPRINT-COL-FONTCOLOR-NEG. See the WINPRINT-COLUMN documentation in Appendix I of the *ACUCOBOL-GT Appendices* guide for instructions and code samples.

1.5 Acu4GL Enhancements

This section describes enhancements made to the various Acu4GL interfaces.

USER_PATH and additional table ownership options for ODBC and DB2

Several new configuration variables make identifying and accessing tables from different owners or from large databases more efficient.

USER_PATH

This variable takes the following form:

```
USER_PATH user1 [user2] ...
USER_PATH user1[;user2] ...
```

Where the user argument may either be the name of a user on the system, or a period (".") which indicates the files are owned by yourself (This will use the authorization id specified in A_ODBC_LOGIN or A_DB2_LOGIN).

A_ODBC_CATALOG or A_DB2_CATALOG

Use these variables to specify a catalog name. This will constrain queries to the named catalog in the database.

A_ODBC_USE_CATALOG or A_DB2_USE_CATALOG

Not all data sources will support a catalog or will return a catalog name that is not useful. For example, Microsoft Access returns the full path to the database, which is not needed since this information is detailed in the datasource. The default behavior of the runtime is to not use the catalog in the actual SQL queries. These variables enables you to control this behavior.

The default value of these configuration variables is FALSE which will cause the catalog portion of the table name to be treated as blank in SQL queries. Setting this variable to TRUE will cause the value of the catalog returned by SQLTables ODBC function to be used in subsequent SQL queries.

Additionally, if the USER_PATH and CATALOG are used, the form of the SQL statement will be modified from:

```
select COL1, ... from TABLENAME ...
```

to:

```
SELECT COL1, ... FROM [catalog.][username.]TABLENAME ...
```

where “catalog” and “username” will be filled in if provided. This can enable you to access tables having identical names, but different schemas.

A_ODBC_TABLE_TYPES or A_DB2_TABLE_TYPES

Use these variables to specify and constrain queries to a particular table type (TABLE, VIEW, and so forth).

Note that using a combination of these variables can speed up the finding of tables in large databases. For example:

```
A_ODBC_CATALOG
A_ODBC_TABLE_TYPES
USER_PATH
```

Improve speed of table open for large databases

For some large databases, the API function **SQLTable()**, which is called to get a list of the tables and information about them, sometimes took a long time to execute. Acu4GL now has the option to build a test SQL query and use the API function call **SQLDescribeCol()** to determine if the table exists. This may improve performance for large databases.

To turn on this new functionality, you can set one of the following configuration variables:

```
A_ODBC_USE_SQLTABLES FALSE
A_DB2_USE_SQLTABLES FALSE
```

The default value of these variables is TRUE, which will cause the interface to use the **SQLTables()** API function (as was the case in versions prior to 8.1).

Improve speed of table open for large databases (columns)

For some large databases, the API function **SQLColumns()**, which is called to get a description of the columns in a table, sometimes took a long time to execute. We now use an alternate API function call, **SQLDescribeCol()**, instead, which can improve performance for large databases.

To turn on this new functionality, you can set one of the following configuration variables:

```
A_ODBC_USE_SQLCOLUMNS FALSE
A_DB2_USE_SQLCOLUMNS FALSE
```

READ on alternate key with no duplicates optimization

Acu4GL for MSSQL has been enhanced to make dynamic READS faster if the READ is on a key with no duplicates. In this case, a cursor is not created until it is required, just as happens when reading on the primary key.

Expanded error reporting with Oracle

Additional error information now appears in the trace so that the same information is available in versions that use the OCI interface as was available in the non-OCI interface.

1.6 AcuBench Enhancements

Report Designer Automated PERFORM Loops

Enhancements to the report designer includes automatic generation of PERFORM loops via the property stylesheet, as opposed to the COBOL source file. This results in reports that can be entirely generated from the PSF file (project structure file).

The stylesheet contains several new event paragraphs called:

BeforeDoPrint
AfterDoPrint
LoadGridInit
LoadGridNext

These event paraprahs are used to provide the code for performing READs, READ NEXTs, and initializing and resetting print loops. These paragraphs are what populates a report with data.

These events were created so that all of your report generation code will be part of the PSF. This is has two major advantages:

1. All of your report generation code can be generated from your PSF, which means your manual code will not be lost in the regeneration process.
2. You will only need to check in your PSF file into your version control system, as opposed to also having to check in the COBOL source file.

Detailed instructions on using these events are provided in Section 17.7 of the *AcuBench User's Guide*.

Applying Automatic Bitmap Scaling

AcuBench now supports the BITMAP-SCALE property introduced in Version 8.0 via a new property option located on the property window. This property is used to enable automatic resizing of bitmaps to match changes made to the bitmap's interior.

If not set, or set to its default value of "0", the bitmap cannot be resized. This means if the size of the interior of the image is smaller than the image, the image is cut to fit. If the interior is larger than the image size, the image size does not increase to fill the interior.

If set to "1", the bitmap will scale up or down to fit the interior given, and no cutting of the bitmap occurs. On code generation the screen section will contain the BITMAP-SCALE property set to 1.

1.7 AcuConnect Enhancements

acurcl report additions

The information function of **acurcl** has been modified in the following ways:

- The start time of the child process now includes the start date.
- The total number of child processes is now included on the information pane of the graphical control panel.

1.8 AcuServer Enhancements

Reregistering with the master server

With the AcuServer load balancing feature, and in cases where the master server dies unexpectedly, the secondary AcuServer can now be instructed to reregister with its master. This prevents having to shut down the entire system.

Checks for all segments of AcuAccess for increased security

acuserve (and **acurcl**) now check all segments of the AcuAccess file for the ability of someone other than root to write to them.

1.9 AcuSQL Enhancements

64-bit UNIX support

The changes made in this enhancement allow for support of AcuSQL connecting to 64-bit drivers in the UNIX environment.

1.10 AcuXUI Enhancements

Character screens, Grid and Tab controls

Version 8.1 includes new support for character screens, grid controls, and tab controls. Many other refinements were made to existing supported controls, events, applets, and bitmap handling. Overall, these refinements result in GUIs that look “cleaner” and behave more predictably in the target environment.

Verisign digital ID

AcuXUI.jar is now signed with a digital ID from Verisign. This digital ID means that it is no longer necessary to modify the local “java.policy” file, as was the case in version 8.0.

Setting Operating System display themes

There are new Command-line options that enable you to change the default OS display theme for a given AcuXUI deployment. This gives you a choice in OS displays themes for changing the look and feel of an AcuXUI interface.

The new options and corresponding themes include:

--localLAF	local system look and feel
--motifLAF	motif look and feel
--xplatformLAF	cross-platform look and feel
--metalLAF	metal look and feel
--windowsLAF	windows look and feel