Blending BBx and BBj

By Jim Douglas

As a consultant to BASIS, Jim brings more than 15 years of experience developing and supporting Business Basic solutions.

Java™ opens up many new opportunities for BBx® developers and for BBx applications. We're excited about the possibilities for the future, but it's also important to remember that BBj™ is the latest version of the BBx language, and compatibility with the past is just as important. We're working hard to ensure that the new enhancements don't change the basic character of the language. Most applications will require few changes, if any, to run in BBj.

Cross-Platform GUI

The most visible change will be in the GUI implementation - it's no longer tied to Microsoft Windows. There's only one version of BBj, and it supports both GUI mode and traditional character mode on all platforms, including Windows and UNIX. BBj supports the SYSGUI device and all SYSGUI mnemonics and functions using the cross-platform Java Swing GUI library.

The SYSGUI device supports the PRO/5 event queue and associated event loop but also supports a simpler syntax in which the programmer defines event-handler subroutines, which are called automatically by the GUI subsystem.

The BBj version of ResBuilder™ creates Java Swing resources. (The BBj ResBuilder also supports .brc and .arc files from Visual PRO/5.)

Unlimited Numbers

The internal format of BBx numbers is called Business Math (template type B). This format supports up to 16 digits of precision, with numbers as large as +/- 9.9E+64. The internal numeric format of BBj handles any number of digits, with unlimited precision. Of course, meaningful precision is still limited by the numbers you plug into the calculation; any numbers you get from BBx numeric template fields (types B, D, X, Y and F) are limited to no more than 14 or 16 digits of precision.

Another effect of this change is that BBj can work with double precision numbers in templates and in the FBIN() and FDEC() functions up to their inherent limit of 1.79E+308.

International Features

BBx defines the STBL("!CTYPE") and STBL("!DATE") structures as a way to help you internationalize your software. BBj still defines STBL("!DATE"), but this structure is loaded automatically with localized date strings. The CVS() function uses Java functions to convert strings to uppercase or lowercase, or to convert unprintable characters to spaces. This allows the BBj application to adjust itself to different languages automatically.

BBx uses ASCII, but Java works with the Unicode character set. BBj stores strings internally as Java (Unicode) strings, but BBj functions (HTA(), ATH(), etc.) work with 8-bit characters, just like BBx. The 8-bit character set is usually ISO 8859-1 (Latin-1).

Template Enhancements

String templates have been enhanced by removing several limits. Field names are no longer limited to 32 characters. (In fact, all identifiers in BBj, including line labels and variable names, are unlimited in length.) We've also removed the 255-character limit for the user-defined attributes associated with a template field; it can be unlimited length in BBj. Binary data types I and U now allow up to 8 bytes, increased from 6 bytes in PRO/5 and 4 bytes in BBxPROGRESSION/4.

Other Enhancements

In developing BBj, we had an opportunity to enhance all areas of the BBx language in ways that wouldn't be possible with a simple update. Some of these enhancements are minor, like removing the limit of 3 dimensions for arrays or allowing fractional seconds for the TIM= clause. And the change to allow more flexible substring references, like F$=FID(1)(9) to return the file name of a channel, is small but useful. But there are other significant enhancements in BBj.

When an error occurs in a BBj program, the ERR and TCB(10) variables are set as in BBx, but the error message is more precise. For example, a reference to an undefined template field (X.Y) generates a generic !ERROR=47 message in BBx. BBj also sets ERR=47, but the displayed error message is "Undefined template field: Y".

BBj manages the work space in a much more dynamic way. The value returned by DSZ can change from instant to instant as Java reclaims unused memory in the background. Because of BBj's new behind-the-scenes memory management system the number of pages specified in a START command is ignored.

BBj programs can be deployed using a traditional interpreter, or they can be compiled to Java bytecode. Interpreted BBj programs work pretty much the way BBx programs work now. Compiled BBj programs support most BBx features, except that self-modifying code (the EXECUTE and DELETE verbs) isn't allowed. Also, the CPL() and LST() functions aren't supported in compiled code.

This is just a short introduction to the new language features in BBj. You'll learn much more at TechCon99 when we'll have a version in Beta testing and you can see and try out these and other features for yourself.