

BASIS International Advantage®

Number 1 ■ Volume 11 ■ 2007



Bxcellerate your Application

Document Management Solutions with UnForm®

Production › Delivery › Archiving › Scanning

,80,5,10,1,c
,"Date",univ
,"Invoice",u
,"Page #",un
s from old p
,"cut (61,5,8
,"cut (71,5,7
,"cut (79,5,2



UnForm is a powerful enterprise document management software solution that seamlessly integrates with any application. The UnForm suite includes laser form and electronic document production, document delivery via email and fax, document archiving and management, and document imaging/scanning. UnForm is a platform independent client server application for Windows®, Unix®, and Linux.

UnForm Laser Forms

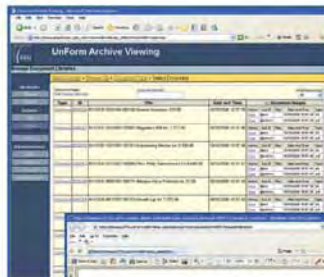
UnForm seamlessly integrates with any software application.

- Windows-based graphical design environment
- Eliminate pre-printed forms with laser printer output
- Produce presentation quality reports
- Create e-Documents in Adobe® Acrobat PDF format
- Email e-Documents automatically
- Print bar codes in most symbologies
- Create laser checks with MICR encoding
- Dynamic image conversion and scaling capability
- Database access via ODBC
- Microsoft® Fax Server support
- PCL 5 and Postscript® printer support
- Extensive programmability

Document Archiving and Management

The UnForm Document Archiving and Management component provides the ability to capture, store and retrieve paper-based and electronic documents.

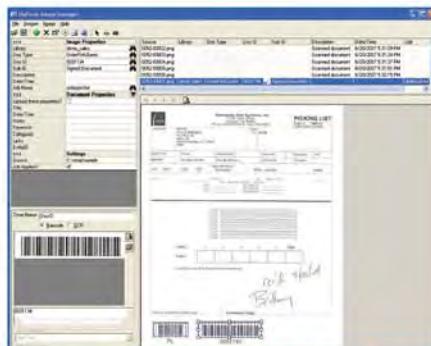
- Rules-based document archiving
- Archive concurrent with document printing
- Store multiple versions of a document
- 10 levels of user defined category indexing
- Document linking control
- Fast web browser-based retrieval
- Client API for application-based retrieval
- Index oriented archive browsing
- Full feature search capability



Document Imaging/Scanning

Windows client application that provides a document scanning and importing tool for capture of documents external to the UnForm processing environment.

- Windows client application
- Integrated work environment for image capture and upload
- TWAIN compliant scanning interface
- Multiple property assignment modes
- Barcode and OCR zone detection
- Automatically match or group images with related archive documents
- Extensibility via VB Script



Document Image Manager



Synergetic Data Systems, Inc.
2195 Talon Drive
Latrobe, CA 95682 USA

(800) 446-7374 or (530) 672-9970
Fax: (530) 672-9975
sales@synergetic-data.com
www.synergetic-data.com

UnForm is a registered trademark of SDSI. Other product names used herein may be trademarks or registered trademarks of their respective owners.

Universal web browser
document retrieval

WHEN IT COMES TO
DATA PROTECTION,
ASK DATA REFLECTOR®
IS A LIFE SAVER.



Highly affordable disaster recovery software.

With leading edge replication technologies, *Data Reflector®* software delivers active and efficient replication for operating system and application files. *Data Reflector®* uses standard TCP/IP communications over a LAN or WAN connection taking full advantage of compression technology to replicate data between the Production and Replication servers. *Data Reflector®* provides an excellent level of Disaster Recovery at a very affordable price.

- ◆ By the minute, hourly and daily replication
- ◆ Updates all directory trees and file systems
- ◆ Up to 10:1 file compression
- ◆ Preserves file ownership permissions, devices and times
- ◆ Minimizes planned and unplanned down time
- ◆ Removes deleted files on destination server
- ◆ Copies changed blocks only
- ◆ Preserves all user password and printer databases

Data Reflector

For more information, go to www.asktech.com
or call +1 (610) 617-0300

ASK TECHNOLOGIES, INC.
All rights reserved.



6



18



28



33

Partnership

- 24 RCG Uses Marketing Skills to Achieve OSAS "Top Dog" Status**
By Randy Ennis
Follow RCG's route to success; implementing sales and marketing focused changes that ultimately lead them to become the #1 OSAS performer.
- 45 Marex Returns Home to BASIS – A Personal Journey**
By Kurt Williams
Join this former BBx-turned-Java programmer's journey as he takes on the challenge to test drive new BASIS development tools.

Language/Interpreter

- 6 Spin, Slide, or View Your Data**
By Bruce Gardner
Explore these new features that will help streamline and modernize BBx applications.
- 7 If You Have Choices, We Have Choosers**
By Adam Hawthorne
Find out about new color, font, file and directory chooser controls that can enhance any GUI business application.
- 11 BBj Spell Checker is all the Buzz**
By Jim Douglas
Empower users to spell check their text inside text box controls.
- 14 A Tour of the BBjCharts API**
By Shaun Haney
Learn how to display application data in a variety of graphical charts.
- 28 Leap from 'Cut and Paste' to 'Drag and Drop'**
By Jim Douglas
Make the move from traditional editing to the slick drag and drop capability found in many other common applications.
- 35 Desktop Data Delivered**
By Adam Hawthorne
Discover the new freedom to access information anywhere: on the server or client.
- 40 Catching the XML Wave**
By Brian Hipple
Learn how to simplify the rather complex interaction with Java's XML packages using BBj Custom Objects.
-  **A Deeper Voyage Into the BBjCharts API**
By Shaun Haney
Explore the deeper terrain of charts. Learn how to customize the basic bar/line/pie charts and create dozens of other types using BBjGenericChart – a new control that allows developers to create ring, box and whisker, and Gantt charts, to name a few.

DBMS

- 33 Unleashing the Power of SPROC's Without SQL**
By Jeff Ash
Learn how to return an SQL record set or new Memory Record Set from information without SQL.
- 37 Solving the Data Warehousing Dilemma**
By Nick Decker
Realize the full potential of triggers and relational ESQl Tables and how they resolve age-old data warehousing problems.

Development Tools

- 18 Watch the Form Gen Wizard Trans "form" Data**
By Robert Del Prete
Get an up-close look at the Form Gen Wizard and see rapid development in action.

SPECIAL SUPPLEMENT

Brewing up GUI Apps with the Barista Application Framework
By Jon Bradley
Meet Barista™ – a powerful new framework for building applications quickly with simple data and interrelationship definitions.

COLUMNS



- 43 BBj 7.0 is a hit at Descore Showcase – By Laurence Guiney**
TechView2007 Travels Cross Country – By Gale Robledo
BASIS and OSAS Partners in Profit – By Gale Robledo

- 47 BASIS Takes Training to the Next Generation**
By Amer Child

NEWS

- 48 Jars, Jars, and More Jars**
By Bruce Gardner



e-article published exclusively at www.basis.com



For more information

Read about e-articles in *Gain the Advantage* – Use the Advantage Resource on our Web site at www.basis.com/advantage/mag-v10n1/gain.pdf

Have You Missed an Issue?

Find all of these articles and more on our Web site

www.basis.com/advantage

Brewing up new Blends for TechCon2007

It has only been a few short months since the BASIS acquisition of AddonSoftware assets and our Engineering team is furiously at work on both technology components of the acquisition; the Addon Rapid Development Environment (ARDE), now renamed Barista™; and AddonSoftware version 8.0, built entirely on Barista and BBj® technology.

The Barista Application Framework is a powerful data dictionary-driven application framework that facilitates the rapid development of GUI versions of existing BASIS applications or the development of new GUI applications. Barista is both an application framework and a run-time tool – it is not a code-generator. Given its dual function, it seemed appropriate to seek a snappy one-word designation; thus evolved the Barista name.

The name Barista refers to one who has acquired a level of expertise in the preparation of espresso-based coffee drinks, especially those Starbucks Americanos our head of engineering, Dr. Kevin King, consumes daily. This no doubt served as his inspiration for suggesting the name. Within certain circles, its meaning is expanding to include what might be called a *coffee sommelier*; a professional who is highly skilled in coffee preparation, with a comprehensive understanding of coffee, coffee blends, espresso, quality, coffee varieties, roast degree, espresso equipment, maintenance, latte art, etc. With BBj's foundations firmly in Java, it seems a most appropriate name for this new *application sommelier*; Barista!

This issue of the *BASIS International Advantage* is filled to the brim with exciting articles about new product features added to the recently released BBj 7. It also includes a pull-out supplement for Barista, now available in beta form from our nightly builds on www.basis.com. Keep your eye out for upcoming Webinar sessions for both Barista and, later in the year, for AddonSoftware Version 8.

Speaking of accounting software, take the time to review our Partnership columns wherein two BASIS developers share their practical knowledge with the BASIS reseller community. Response Computer Group Inc., the leading Open Systems Inc. reseller for 2006/2007, shares the secrets of their success on page 24 and Marex Computer Services shares on page 45 how they rediscovered BBx® technology and found their passions for SQL and object-oriented programming are right at home with BBj.

These are exciting times. BASIS continues to offer a modern Business BASIC development language, a robust and scalable DBMS, and powerful development tools coupled with unparalleled interaction with our partnership community. Research shows that key marketing considerations the user community applies in their evaluation of a software solution incorporate the 4 R's:

Reduction in risk for the customer
References and recommendations
Relationships
Realization of performance

Attend the BASIS TechCon2007 and extend your stay for the complimentary post-conference Sales and Marketing Workshop. Learn how to employ these elements while you kick-start your 2008 strategic planning process. These are the yardsticks by which we must measure ourselves and perform. Together, BASIS and the BASIS partnership community leverage the combination of our mutual track-record and longevity to deliver on these components of our customers' needs.

We succeed because you succeed; here's to more of the same! 



Nico Spence
Chief Marketing
Officer

The *BASIS International Advantage* magazine is published and distributed by BASIS International Ltd.

Editor in Chief Nico Spence
nspence@basis.com

Editor Susan Darling
sdarling@basis.com

Technical Editors Dr. Kevin King, Nick Decker
kking@basis.com, ndecker@basis.com

Copy Editor Peggy Lewis
plewis@basis.com

Art Director, Graphics Patricia Catlett
pcatlett@basis.com

Electronic Production Amer Child
achild@basis.com

Printing/Distribution Services
Albuquerque Printing Company

BASIS Corporate Portraits
Dale Frederick Photography

BASIS does not endorse any products mentioned in the *BASIS International Advantage* other than those products licensed by BASIS International Ltd.

The trademarks and registered trademarks owned by BASIS International Ltd. in the United States and other countries are listed at www.basis.com/company/trademarks.html. All other product and brand names are trademarks or registered trademarks of their respective companies.

Advantage Subscriptions
www.basis.com/advantage/subs.html



BASIS International Ltd.
5901 Jefferson Street NE
Albuquerque, NM 87109-3432

Phone +1.505.345.5232
US Sales 1.800.423.1394
International +1.505.338.4188
www.basis.com

General Information
info@basis.com

© 2007 by BASIS International Ltd.
All rights reserved.

Spin, Slide, or View Your Data

by Bruce Gardner



With each new version of BBj®, BASIS provides developers with powerful new language features and enhancements. BBj 7.0 delivers on these expectations with a set of new features that will help streamline and modernize your application: [BBjSlider](#), [BBjHtmlView](#), [BBjPrintPreview](#), and a variety of [BBjSpinner](#) controls. As with all BBj GUI controls, developers can create these controls programmatically or include them in a FormBuilder resource file requiring little or no coding.

Sliders

Sometimes, less is more. To represent a relative quantity in an application, a slider is the best control to use. Sliders are among the most natural and intuitive of interfaces. The user absorbs the information at a glance – right is more, left is less. If the slider orientation is vertical, up is more, down is less.

Microsoft Windows users are familiar with the slider in volume controls, seekers for sound and movie files, and color choosers. The new BBjSlider control displays a value based on the position of the slider in either horizontal or vertical orientations. The control is customizable, as shown in **Figure 1**, to meet the specific needs of any application including data binding, labeling, major and minor tick spacing, and more.

Spinners

If the ability to select specific values or strings is required, spinners are just the ticket. Spinners allow users to roll or 'spin' through a list of available options or type the input directly. Users often prefer spin controls because it allows them to make changes without moving their hand from the mouse.

BBj 7.0 meets this demand by adding spinner functionality to [EditBox](#), [InputD](#), [InputN](#), and [InputE](#) controls as shown in **Figure 2**. As with their non-spinning brothers, these controls can be databound and support data filling. Users can increment or decrement a value displayed and see resulting record set changes instantly. With fewer messy lists to open and a cleaner form, BBjSpinners make it easier than ever to provide quick and concise updates, insertions, and modifications of records.

HTML View

Recently, one of the most requested features was to add the ability to display HTML on a form. Not to be confused with full-blown browser functionality, the BBj 7.0 BBjHtmlView control delivers the ability to display HTML or the contents of a URL in a BBj-constructed window.

Most modern applications mix local and internet resources to provide their customers with up-to-date help and support information.

continued on page 10...

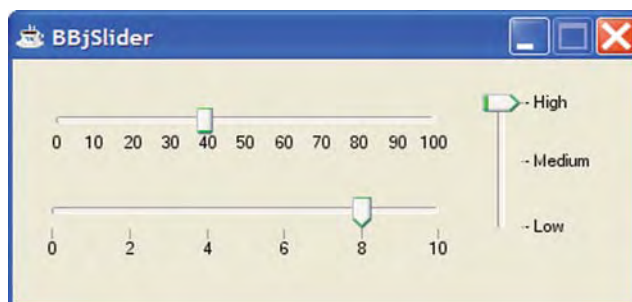


Figure 1. Various examples of BBjSlider controls



Figure 2. Example of InputN, InputD, and InputESpinners



Bruce Gardner
Quality Assurance
Engineer

If you Have Choices, we Have Choosers

By Adam Hawthorne



User interfaces continue to evolve, providing more customization and allowing users to tailor their environments to suit their own needs. In many applications, selecting a font style or color or even a file is as easy as clicking on the selection from a list of options. BBJ® 7.0 introduces new controls, called choosers, with which developers can offer similar choices to their users who expect real-time interaction with their application. This article introduces the various new BBJ chooser controls and shows how they can enhance any GUI business application.

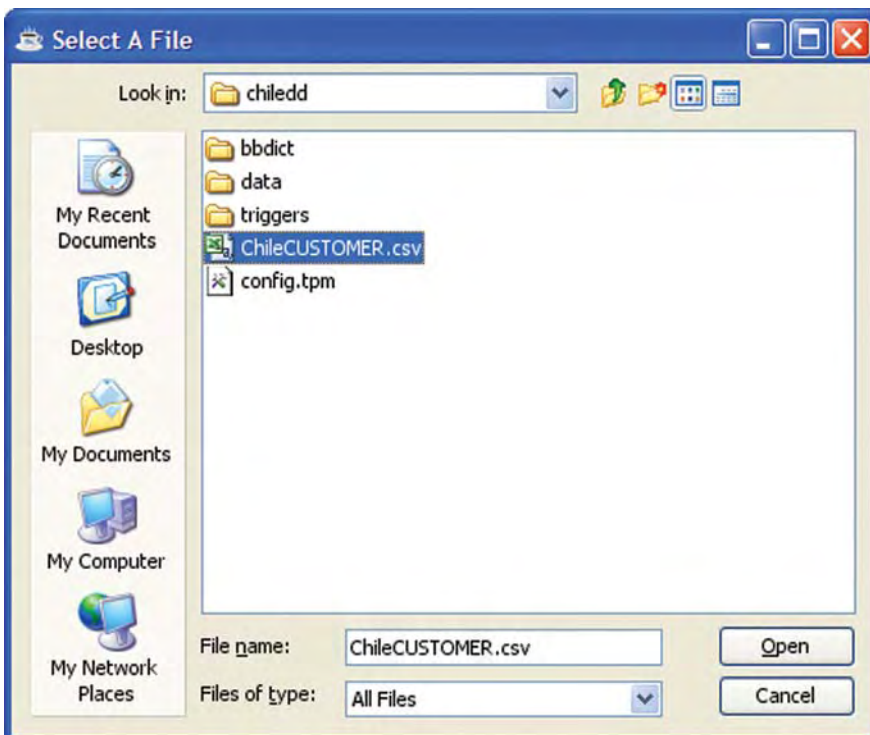
File/Directory Chooser

BBj developers are no strangers to using a file chooser dialog. The new **BBjFileChooser** control is a highly configurable SYSGUI-based complement to the existing **FILEOPEN()** and **FILESAVE()** functions and the **'FILEOPEN'** and **'FILESAVE'** mnemonics of the **SYSWINDOW**. For developers who only need a simple blocking file choice, these solutions will suffice, but to add a custom preview panel to a file chooser dialog or provide additional configuration options on the dialog window, it is only possible with the **BBjFileChooser**.

For example, a developer could put a file chooser control on a window with pre-existing controls and behavior. Then, allow the **BBjFileChooser** to interact with standard **READ RECORD** or **PROCESS_EVENTS** event loop to avoid interrupting program flow or to remain responsive while the dialog is visible. The developer could also add the file chooser to an existing form to provide real-time responses to file selections, or augment the UI with an Explorer-like directory tree. **Figure 1** shows a sample of the file chooser in action.

Flags on the **BBjFileChooser** gives developers a wide variety of options, such as:

☞ Allow the previously mentioned directory selection in an Explorer-like fashion by simply adding the flag **(\$0008\$)** to the **addFileChooser** method, turning a standard file chooser into a much more efficient directory chooser dialog as shown in **Figure 2** on the next page.



☞ Use the default behavior of an OPEN dialog, or specify the **SAVE flag (\$0100\$)** to achieve standard internationalized labels for the file chooser and its buttons.

☞ Browse on the server's filesystem by default, or allow users to browse on their own computer's filesystem using the client-browsing flag **(\$0004\$)**.

The **BBjFileChooser** also provides methods to interact with the user's view of the filesystem, flexible glob-based (pattern matching) file filter configuration, and full single and multiple file selection support.

continued...



Adam Hawthorne
Software Engineer

Figure 1. Sample file chooser in action

BBjFontChooser

The [BBjFontChooser](#) provides a fully functional chooser for fonts. Rather than building a dialog and performing the countless steps needed to determine the list of fonts; populating list controls with the fonts, sizes, styles, and other tedious tasks; developers can use the ready-made control that will correctly follow the locale settings of the client machine. It also automatically produces a list of fonts, sizes, and styles, and shows a preview of the font without any programmer interaction, although the programmer can set the text to preview the font (see **Figure 3**).

The BBjFontChooser seamlessly integrates with BBj's Visual PRO/5®–compatible treatment of fonts, specifically respecting the settings of the 'SCALE' mnemonic. For those who do not have a dependency on compatibility or simply do not want this level of compatibility, the \$0004\$ flag turns off the compatibility mode.

BBjColorChooser

The [BBjColorChooser](#) delivers a thorough treatment of the color spectrum. With three ways to make color choices, even the most detail-oriented graphics expert will have no cause to complain. The color chooser allows users to select colors based on standard swatches as well as RGB and HSB color spaces as shown in **Figure 4**. While including all the standard chooser features, the BBjColorChooser also adds the ability to hide the preview panel and implements the new drag and drop architecture to drag colors from one application to another.

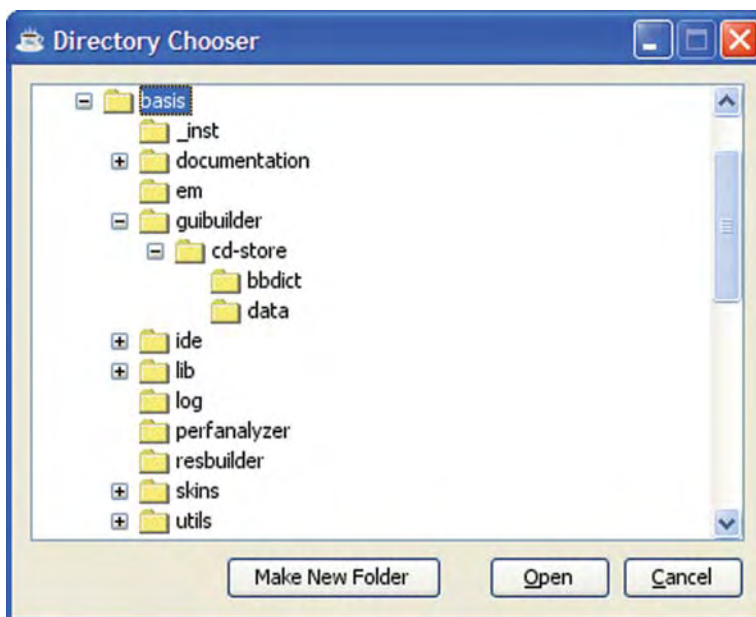


Figure 2. Sample directory chooser dialog

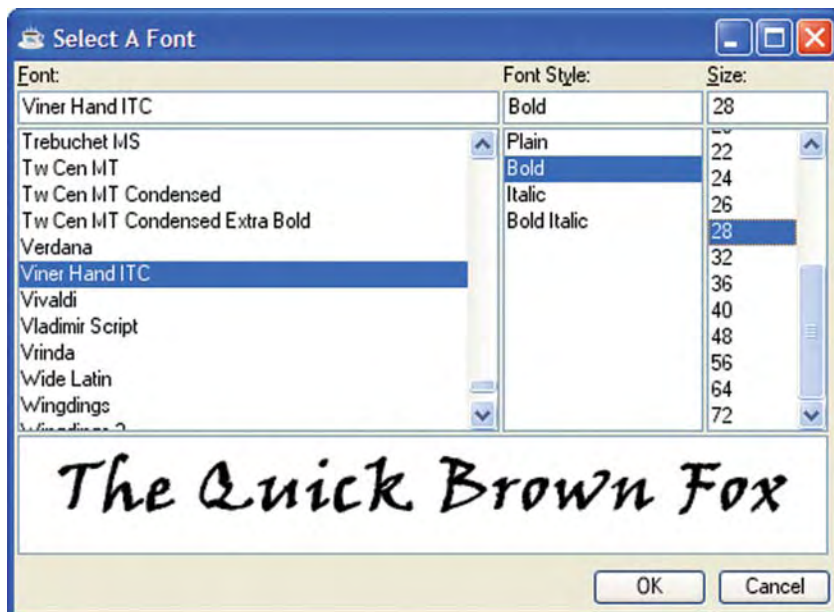


Figure 3. A dialog displaying the BBjFontChooser

Working with Control Buttons

The new chooser controls provide shared functionality regarding the standard control buttons. For example, when the user selects a control button like [OK] or [Cancel], each chooser fires an Approve or Cancel event. To mimic a traditional chooser dialog, add a chooser control to a window that has a \$00080000\$ dialog behavior flag, then set the chooser's control id to 1 like an [OK] button, and register for the Approve and Cancel callbacks. The chooser will correctly map the [Enter] and [Esc] keys to the included buttons and fire the events when appropriate. When the user completes the selection, the program can respond to the event by hiding or destroying the dialog window.

Multiple Choices

All of the chooser controls offer extended functionality and customization capabilities. For example, developers can customize the control button text if desired. The BBjFileChooser allows setting the text of the [Open] or [Save] button and the BBjFontChooser and BBjColorChooser allow setting the text of both the [OK] and [Cancel] buttons. It is even possible to hide the control buttons completely, making the choosers a perfect choice for inclusion in a palette.

continued...

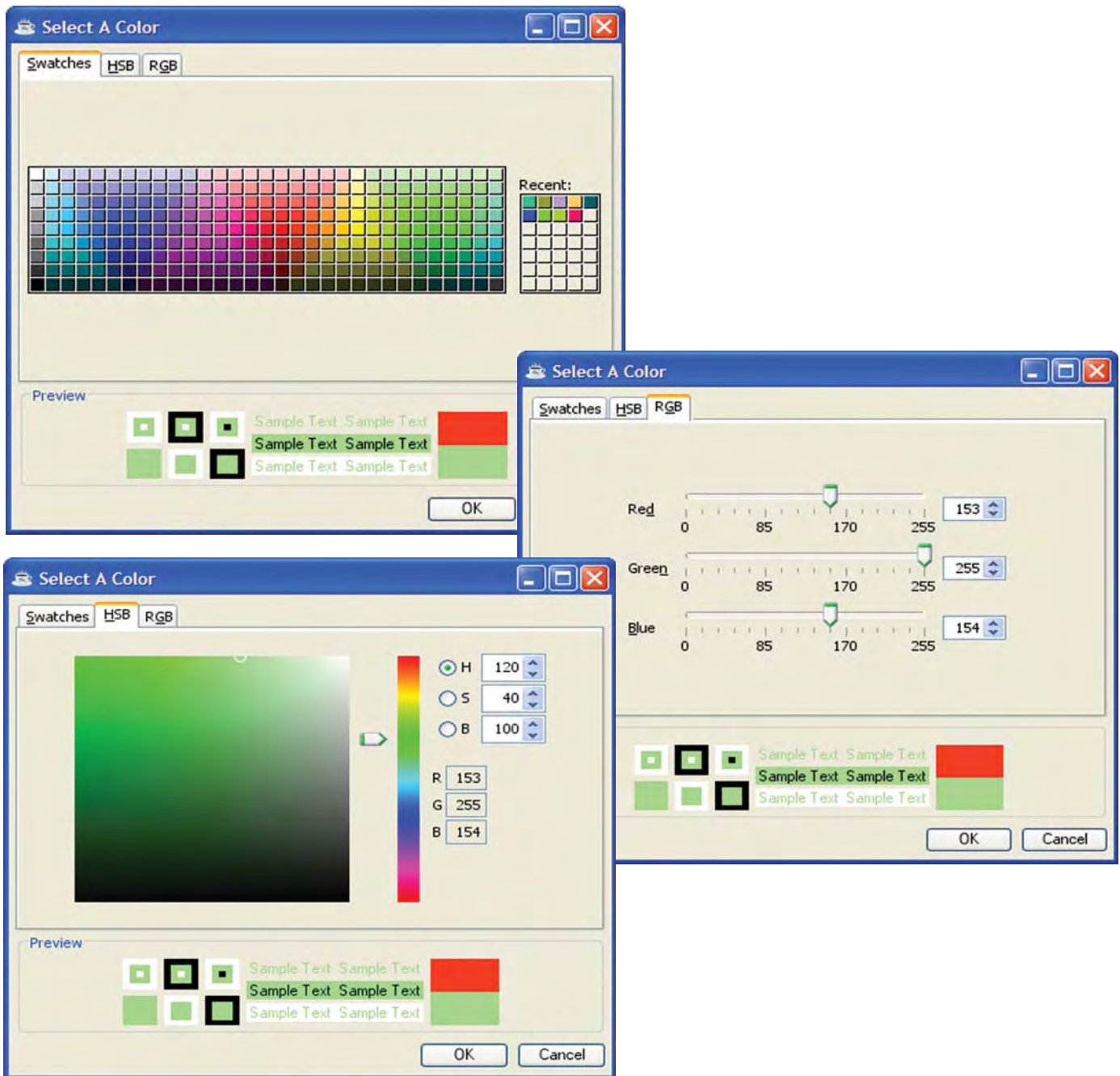


Figure 4. Color selection options

Chooser Interaction

Each chooser has an `approveSelection` and `cancelSelection` method to allow programmatic activation of the associated buttons. Use these methods to facilitate combining event-handling code and the response to in-program actions or use them to decouple the event from the response. Lastly, all of the chooser controls provide `Change` events that respond to real-time selection changes within the chooser. This enables the program to react instantly to any choice the user is considering, and is often used to provide previewing capabilities.

Summary

With BBJ chooser controls, BBJ developers can now provide behaviors and customizations easily and effortlessly, looking back on the bygone days of recreating standardized dialogs from scratch, and implementing and testing complicated controls and their interaction. Today, they can toss a control on a window and respond to the events, resting easy knowing that these controls conform to system standards with correct localizations. Customizing the user interface is just a matter of extending the concept of a 'standard dialog' by creating a form with a chooser and various other controls. Now that BBJ offers so many choices, developers can start choosing! 

Spin, Slide, or View Your Data

continued from page 6...

Perhaps the most common implementation of this feature is to display online documentation as shown in **Figure 3**. New callbacks – HYPERLINK_ACTIVATE, HYPERLINK_ENTER, and HYPERLINK_EXIT – provide the developer with a host of options to respond to user clicks and mouse moves within the control.



Figure 3. A BBjHtmlView displaying online documentation

Print Preview

Almost all applications have a print preview feature. Modern applications require more sophisticated methods to preview, navigate through, and display their output. With the new BBjPrintPreview control, BASIS provides an alternative to the default print preview allowing for a quick and easy way to add a fully customizable print preview to display the output of a BBjForm print job.

To create a customized preview similar to the one shown in **Figure 4**, use custom buttons, including the new MenuButton, and the many new methods now in BBj to navigate and display pages in any number of ways. Choose to display first and last pages or two pages at a time, utilize a page count, set the zoom factor, or localize the display, to name a few of the options available to the programmer with this new control.

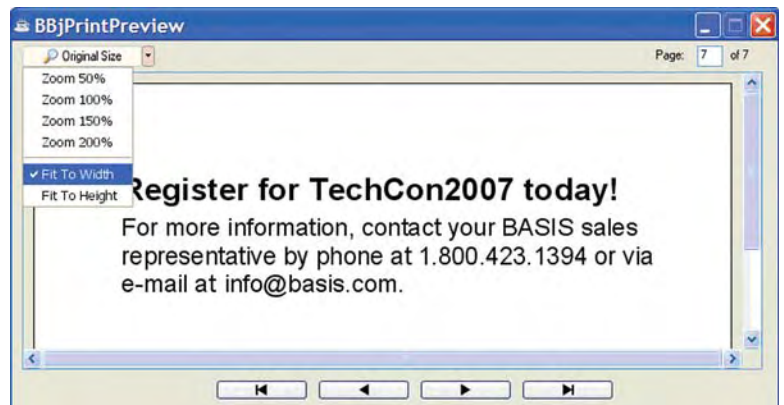


Figure 4. Zoom capability and custom navigation buttons in a print preview

Summary

BASIS continues to bring developers powerful new features and enhancements. [Download BBj 7.0](http://www.basis.com/products/bbj/download.html) today at www.basis.com/products/bbj/download.html and see how these new features can extend, streamline, and modernize any BBx application. 



BBj Spell Checker is all the Buzz!

For Spelling Bee Dropouts



By Jim Douglas

S

pell checking is a standard feature in most word processing programs which users around the world have come to depend upon. Sadly, many non-word processing applications such as spreadsheet programs, personal data managers, applications built on Visual PRO/5®, etc. do not offer this valuable tool.

With BBj® 7.0, this is no longer the case. Spell checking is now a feature developers can add to any text-based GUI control. Users can now happily spell check their input right inside their BBj application. This article provides an overview of the capabilities of this new feature implemented as the Interface *TextControl*.

Overview

Spelling bee dropouts can depend on this new feature to suggest alternate spellings and correct mistakes with just a click of the mouse. Developers can set this feature as a default in the application or give the user the option to toggle its activation. In either case, users can employ the spell check tool they depend on in other applications.

To enable or disable spell checking in any text-based GUI control, simply use the method `setSpellChecked` as follows:

```
cedit!.setSpellChecked(1)
```

What it Does

Similar to common word processing programs, the BBj spell checker underlines words that appear misspelled in a

distinctive zigzag pattern. By default, the underline is red, but developers can customize it with `setSpellCheckColor`. When the user right-clicks on an underlined word, BBj pops up a menu with several options:

- Suggested corrections (selecting a correction changes only this occurrence of the misspelled word)
- **Ignore All** (removes the red underline from all occurrences of this word in this control)
- **Add to dictionary** (adds this word to your custom dictionary)
- **Cut, Copy, Paste, and Select All**

If the user right-clicks anywhere in the control other than on a misspelled word, the right-click menu shows only **Cut, Copy, Paste, and Select All**.

How it Runs

To see what this new feature might look like in your application, experiment with a sample program available for download at the URL listed at the end of this article.

Figure 1 shows this sample in process. It displays each type of text control that offers this ability – BBjCEdit, BBjEditBox, BBjInputE, and BBjListEdit controls – all containing misspelled words identified with the red zigzag. The sample also shows the spell check menu resulting from right clicking on a misspelled word. A check box option at the bottom of the screen allows the user to toggle the spell check option.

continued...

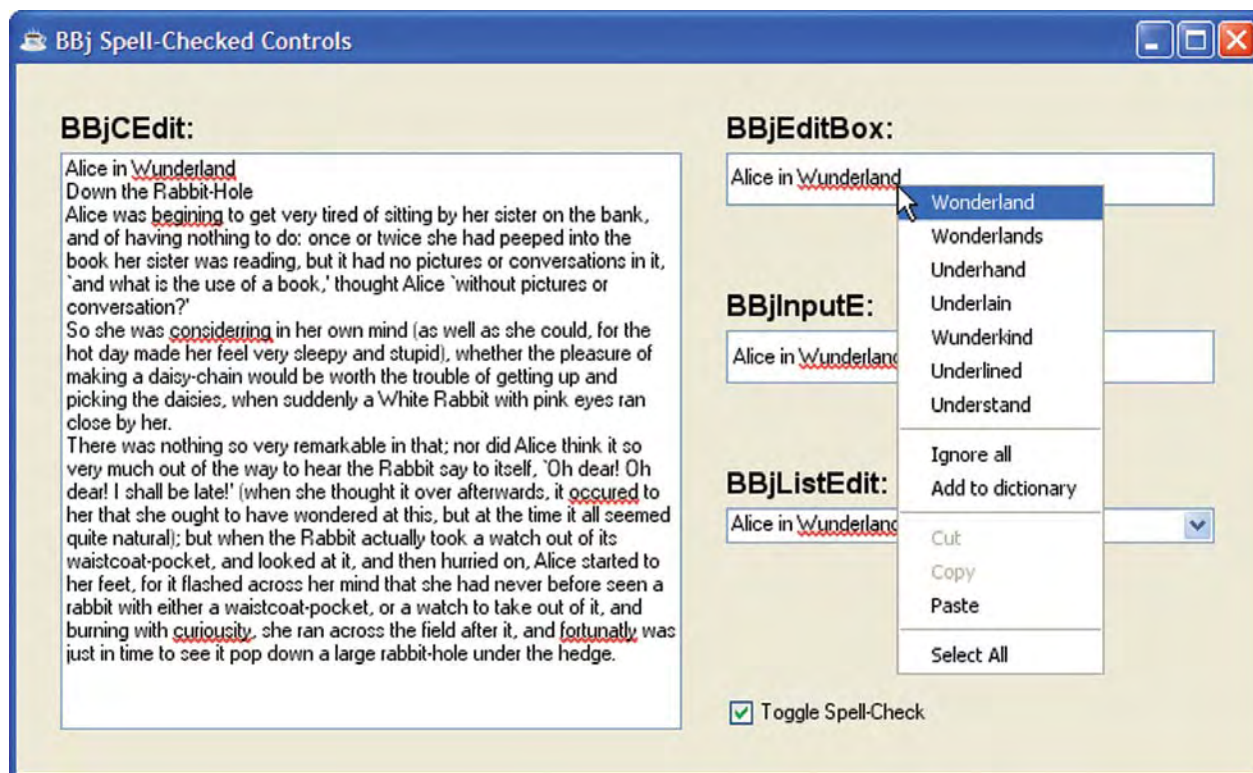


Figure 1. Spell check sample



Jim Douglas
Software Engineer
Contractor

Language Support

Spell checking in BBJ is currently available in several languages and dialects. Select from Dutch, English (American, Canadian, or British), French (including Canadian French), German ([pre- or post-reform spelling](#)), Italian, Spanish, and Swedish.

The default spell-checking language is taken from `STBL("LOCALE")`. For a complete list of supported language codes, see [setSpellCheckLanguage](#) in the online documentation at www.basis.com.

Standard and Custom Dictionaries

While the standard English dictionary ships with about [100,000 words](#), dictionaries for [other languages](#) typically include more words. The Italian dictionary, for example,

includes 417,000 words. In all cases, the goal is to include the vast majority of commonly used words in any given language while omitting infrequently used words. Over time, users will build up one or more custom dictionaries based on their unique requirements. By default, the spell checker creates a custom dictionary file named `CUSTOM.DIC`. Setting `STBL("DICTIONARY")` or calling the [setCustomDictionary](#) method will override this default.

Advanced Features

Many applications will simply call [setSpellChecked](#) to enable interactive spell checking. However, for specialized requirements, developers can customize the spell checker defaults and update the custom dictionary under program control via the following methods:

METHOD	COMMENTS
addWord	Adds a word to the custom dictionary. This is equivalent to selecting "Add to dictionary" from the right-click menu.
removeWord	Removes a word from the custom dictionary.
ignoreWord	Ignores a misspelled word without adding it to the customized dictionary, equivalent to selecting "Ignore all" from the right-click menu.
setMaxSuggestions	Sets the maximum number of suggestions to display in the right-click menu.
setSpellCheckColor	Changes the color of the zigzag underline.
setSpellCheckLanguage	Changes the spell-check language; the default language is from <code>STBL("LOCALE")</code> .
setCustomDictionary	Specifies a different custom dictionary file. The default custom dictionary is either <code>STBL("DICTIONARY")</code> or <code>CUSTOM.DIC</code> .
isMisspelled	Returns whether a word appears to be misspelled.
getMisspelledWords	Returns a list of words in this control that appear to be misspelled.
getSuggestions	Gets suggested corrections for a misspelled word.
setSpellCheckOption	Customizes specialized options. This is typically not required; the default option settings are appropriate for most applications.



continued...



For a complete list of methods relating to spell checking, see the [Interface *TextControl*](#) topic in the BASIS online documentation.

Thin Client Deployment

Because the language dictionaries are rather large and because most applications do not use all languages, the dictionary files are not included in the standard thin client jar configuration. When deploying a client application that uses spell checking, be sure to add the required dictionary file from the list below to the cache archive. 

LANGUAGE	SPELL-CHECK LANGUAGE FILE(S)
American English	SpellCheckerDictionary-en.jar SpellCheckerDictionary-am.jar
British English	SpellCheckerDictionary-en.jar SpellCheckerDictionary-br.jar
Canadian English	SpellCheckerDictionary-en.jar SpellCheckerDictionary-ca.jar
Spanish	SpellCheckerDictionary-sp.jar
French	SpellCheckerDictionary-fr.jar
Italian	SpellCheckerDictionary-it.jar
Dutch	SpellCheckerDictionary-du.jar
Swedish	SpellCheckerDictionary-sw.jar
German	SpellCheckerDictionary-ge.jar

Summary

A familiar interface such as BBj's spell checker is easy to use and intuitive. With one simple call to the specific text control, developers can empower users and spelling bee dropouts alike to perform their jobs to their greatest potential. Perfection may now be just a click or two away! 



Download the sample at
www.basis.com/advantage/mag-v11n1/spellchecker.zip



For an introduction to spell checkers, see
en.wikipedia.org/wiki/Spell_checker

**Renew your free subscription
to the
BASIS International Advantage Magazine
www.basis.com/advantage**

A Tour of the BBjCharts API

By Shaun Haney

W

ith the introduction of the BBjChart API in BBj® 7.0, it is now convenient to display application data graphically in the form of a chart. Charts add value to applications by allowing users to answer questions with a quick glance at a chart. Are we attaining our sales goals? Which geographic areas are generating the most revenue? Is our new Web site attracting customers?

Imagine Florence's Bakery, a local business that makes gourmet delicacies such as Lemon Raspberry White Chocolate Mousse Cake, Flourless Chocolate Torte, and Black Bottom Tiramisu. This bakery thrives locally and several customers gave feedback to Florence that no other bakery within the state makes such unique cakes. After some research and discussion, Florence, the proprietress, determined that an e-commerce system would be valuable for expanding her customer base outside of her neighborhood. Now, one year after the e-commerce system launched, it is in full swing. Still, one question looms, "Have sales really improved as much as it appears?" The BBjCharts API can help provide this information at a glance.



The Journey Begins

The BBjCharts API includes three "ready-to-use" charts and a highly customizable chart. The ready-to-use charts include a pie chart called **BBjPieChart**, a line chart called **BBjLineChart**, and a bar chart called **BBjBarChart**.

BBjGenericChart is a highly customizable chart control that allows the BBj programmer to create any of the dozens of additional charts in the **JFreeCharts API**. This journey will show us how Florence can use the BBjCharts API to answer her business questions. Along the way, we will explore uses for the different kinds of

BBjCharts and learn how to create them.

The First Step – Creating a BBjPieChart in the BASIS IDE

Using AppBuilder, it is easy to create a chart like the one in **Figure 1** and populate it with just a few lines of code. Florence wants to know what portion of this year's revenue e-commerce generated. Pie charts represent proportional data and since Florence wants this data quickly, AppBuilder is the ideal tool.



Figure 1. Pie chart created using AppBuilder

To create the pie chart in AppBuilder, first create a resource and place the BBjPieChart control onto the resource's main window. Right-click the resource file and select **Create AppBuilder File**. Double-click the new AppBuilder file to open it, and enter the code in **Figure 2** into the Init block (or cut text from the **appbuilder.txt** file in the downloadable .zip file referenced at the end of this article):

```
declare BBjPieChart pieChart!

REM Get the pie chart control from the window.
pieChart!=cast(BBjPieChart, BBjAPI().getSysGui().getActiveWindow().getControl(100))

REM Normally the data would be retrieved from a database, but the values
REM are read from the data statement for this self-inclusive example
while 1
  dread sliceName$, sliceAmount,end=*BREAK
  pieChart!.setSliceValue(sliceName$,sliceAmount)
wend

data "E-Commerce",50319.80, "Walk-in Sales",73506.80, "Telephone Orders",112055.10
```

Figure 2. Sample code from **appbuilder.txt**

After entering this code, build the AppBuilder file and run it to see the chart in all its brilliance.



Scenic Overlook – the BBjPieChart

AppBuilder creates a pie chart quickly and easily, doing all the work for us. However, would a BBj programmer generate the same pie chart using the BBjCharts API within a hand-coded program? The BBjPieChart is the easiest of the charts in the BBjCharts API to create and populate so that will be the next stop on our journey.

continued...



Shaun Haney
Quality Assurance
Engineer

BBjPieChart Creation

First, create the pie chart by invoking the method `addPieChart`. It takes seven arguments; five are common to all BBjControls, and the last two, which are boolean values, are specific to BBjPieChart. They indicate whether to display a legend and whether to display the chart with a 3D effect. For example, if creating a pie chart with the following line of code, the two 1's at the end of the call result in a pie chart with a legend and a 3D effect.

```
pieChart!=win!.addPieChart(101,0,0,640,480,1,1)
```

BBjPieChart Population

To populate a BBjPieChart, simply add slices of any size to the pie. BBjPieChart maintains a running total and displays each slice according to its portion of the total. Name each slice when adding it to the pie and the name appears in the legend. We add slices by invoking the method `setSliceValue`, which takes a slice name and a value as arguments and adds them to the pie chart. For example:

```
pieChart!.setSliceValue("A",2)
pieChart!.setSliceValue("B",5)
pieChart!.setSliceValue("C",3)
```

These statements will create a pie chart with a total value of 10. Slice **A** would occupy 20% of the pie; slice **B** would occupy 50% of the pie; and slice **C** would occupy 30% of the pie.

BBjPieChart Example

Read and run the sample code `piechart.src` shown in **Figure 3**. The resulting chart in **Figure 4** showed Florence that e-commerce generated approximately 20% of her revenue for this year.

```
REM Florence's Bakery: Portion of Sales From E-Commerce

REM declare our standard GUI Objects
declare BBjSysGui sysgui!
declare BBjTopLevelWindow win!
declare BBjPieChart pieChart!

sysgui!=BBjAPI().openSysGui("XO")
win!=sysgui!.addWindow(10,10,500,325,
: "Florence's Bakery: Portion of Sales from E-Commerce")

REM Create our pie chart
pieChart!=win!.addPieChart(win!.getAvailableControlID(),
: 5,5,490,315,1,1)

REM Populate the pie chart's data. These values are
REM hardcoded for simplicity. Data would normally
REM be read from a database or data files.
pieChart!.setSliceValue("E-Commerce",5031.98)
pieChart!.setSliceValue("Walk-in Sales",7350.68)
pieChart!.setSliceValue("Telephone Orders",11205.51)

win!.setCallback(win!.ON_CLOSE,"goodbye")

process_events

goodbye:
release
```

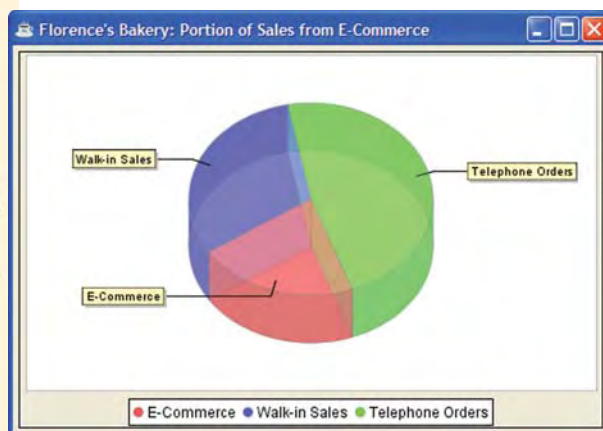


Figure 4. Pie chart created using BBjAPI

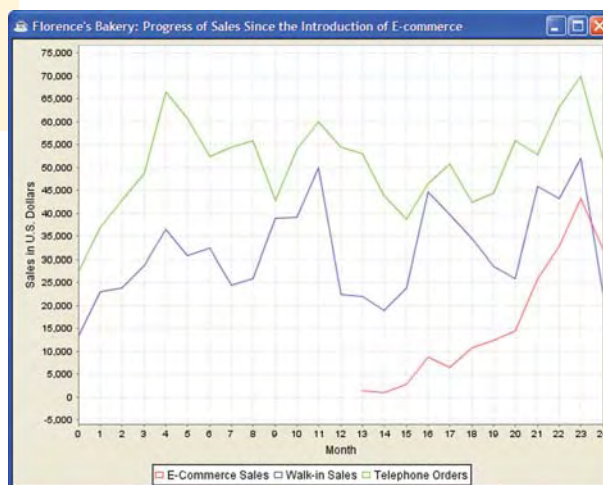


Figure 5. Line chart created using BBjAPI

Figure 3. BBjAPI code sample for the pie chart



Scenic Overlook – BBjLineChart

Line charts are especially helpful to show the progress of a series of data over a given period of time. One common use for the line chart, for example, is showing the rise and fall of prices on stocks. The line chart in **Figure 5** shows how sales from the e-commerce site compare with sales from telephone orders and walk-in sales.

The BBjLineChart Dataset: Series and XYValues

Every chart in the BBjCharts API has a dataset. The BBjPieChart's dataset is very simple and does not require preparation before creating the chart. By comparison, BBjLineChart's dataset requires slightly more preparation. The dataset consists of one or more series and each series consists of any number of Cartesian coordinates known as XYValues. A series represents one of the lines in the chart and the XYValues are the points that this line connects. In order to create a BBjLineChart, it is necessary to specify the number of series, i.e. lines, in the chart.

continued...

Creating a BBJLineChart

Using the method `addLineChart`, specify a label for the X axis, a label for the Y axis, the number of series, and whether to display a legend as shown below:

```
lineChart!=win!.addLineChart(101,0,0,640,480,"Month",  
: "Sales in U.S. Dollars",3,1)
```

Populating a BBJLineChart With a Series

Populating the line chart is a matter of naming the series and then specifying a set of XYValues for each series. Each series displays as a line connecting its specified XYValues. There is no limit to the number of XYValues that one can specify and no need to adhere to any discrete interval along the X-axis.

To name the series, use `setSeriesName`. This method takes a zero-based index, and a string indicating the name of the series. For example, to name the second line in the chart **Walk-in Sales**, type:

```
lineChart!.setSeriesName(1,"Walk-in Sales")
```

After naming the series, add XYValues to the series for its line to appear by calling `setXYValue`. This method takes the series' zero-based index, an x-value, and a y-value. In the example, the x-value represents a month and the y-value represents a sales amount for the month.

Suppose that in January, the bakery's first month, walk-in sales were \$5000 and then rose to \$7500 in February. The code for these facts, resulting in a rapidly rising line in the graph, would be as follows:

```
lineChart!.setXYValue(1,1,5000)  
lineChart!.setXYValue(1,2,7500)
```

BBJLineChart Example

Run the sample `linechart.src`, which shows the progress of three types of revenue for Florence's Bakery: sales from the e-commerce site, which did not launch until the 13th month; walk-in sales, and telephone orders. From the data in this example, Florence was able to determine that e-commerce was a significant source of revenue.



Scenic Overlook – the BBJBarChart



Bar charts use ingots or bars to give side-by-side comparisons of different series of data. Florence wants to know whether sales improved after the introduction of e-commerce. A bar chart would be useful for a monthly comparison of this year's and last year's sales.

Before creating the BBJBarChart, it is important to understand how its dataset is organized.

The BBJBarChart Dataset: Series, Categories, and Values

The BBJBarChart's dataset consists of series, categories, and values as shown in **Figure 6**.

Series - represents particular entities that the chart compares. Each bar color represents an entity and the names of these entities will appear in the legend. The entities compared in this example are **Last Year** and **This Year**.

Category - a group of the entity's values. In this case, the category is **months** so every **month** has a value for **Last Year** and **This Year**.

Value - the actual unit of data. In this case, a **value** represents total sales during a particular month for a particular year such as sales for the month of March 2006.

In another example, Florence wanted to find out the most popular types of cake that each sales associate sold. To compare types of cake sold by five sales associates, set **Employee** as the series and **Cake Type** as the category. This resulting bar chart would display a label for each type of cake at the bottom of the chart and one bar color would represent each employee.

continued...

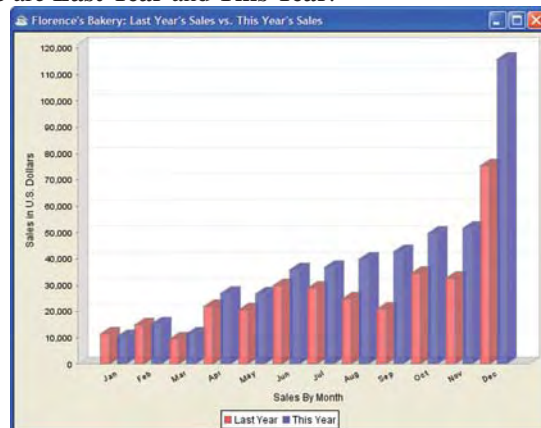


Figure 6. Bar chart created using BBJChart API

Creating the BBjBarChart

To create a BBjBarChart, use BBjWindow's factory method, [addBarChart](#). This method takes twelve parameters, including a label for the X axis, a label for the Y axis, the number of series, the number of categories, and boolean values specifying whether to show a legend, display the bar chart with 3D bars, and whether the bar chart is a horizontal bar chart.

For example, to create a bar chart with a control ID of 101 in the top left corner of the window with dimensions of 600 by 450 pixels, an X axis labeled **Sales By Month** and a Y axis labeled **Sales in U.S. Dollars**, 12 categories, 2 series, a legend, the bars in 3D, and without horizontal orientation, enter this statement:

```
barchart!=win!.addBarChart(101,0,0,600,450,"Sales By Month","Sales in U.S. Dollars",12,2,1,0,0)
```

Populating the BBjBarChart

In order to populate the bar chart, first give each series and category names, then set values for each series within a given category. If a category is unnamed, it will not appear on the chart even though the category exists and can accept values.

The method provided for setting the series names is [setSeriesName](#). This method takes a 0-based index and a string for the name of the series. The code to set up our series, This Year and Last Year looks like this:

```
barChart!.setSeriesName(0,"Last Year")
barChart!.setSeriesName(1,"This Year")
```

Similarly, the method to set each category name is [setCategoryName](#) and the index is 0-based so indexes 0 through 11 represent the 12 months of the year. The code to set a category name looks like the following:

```
barChart!.setCategoryName(6,"Jul")
```

The dataset for a bar chart is conceptually a two-dimensional array with the series as the first index and the category as the second index. Use these indexes to set each value. The method to set values in BBjBarChart, [setBarValue](#), takes a series index, a category index, and a value. For example, to set this year's sales for July to \$4,358, use the following line of code:

```
barChart!.setBarValue(1, 6, 4358)
```

BBjBarChart Example

Study and run the sample named **barchart.src**. The sample shows that sales indeed experienced an improvement over last year's sales with the introduction of e-commerce.



Invitation to Explore Deeper Terrain

The three basic charts in the BBjCharts API are sufficient for most purposes, but sometimes an application requires a different kind of chart. The online supplement to this article referenced at the end of this article covers how to customize the three basic charts using [getClientChart](#) and how to create dozens of other types of charts using [BBjGenericChart](#). BBjGenericChart allows developers to create ring charts, box and whisker charts, Gantt charts, and other charts listed at www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/ChartFactory.html.

The Final Stop

Even for such simple business needs as Florence's Bakery, the new BBjCharts API will display data graphically, easily, and effortlessly. If the familiar bar charts, line charts, and pie charts do not meet specific needs or do not present the data clearly enough, developers have all the options available under the sun using BBjGenericChart to access the JFreeChart API and produce many diverse charts. Clearly, the new BBjCharts API will be useful in a wide variety of BBj applications. 



Download the sample code from
www.basis.com/advantage/mag-v11n1/charts.zip



There is more!

For a deeper look into charts go to

www.basis.com/advantage/mag-v11n1/charts-deeper.html

See all of the available charts in the JFreeChart library by visiting

www.jfree.org/jfreechart/samples.html

Watch the Form Gen Wizard Trans “form” Data

By Robert Del Prete

The BASIS IDE now includes the [Form Gen Wizard](#), a new rapid application development tool for developers. The Form “Gen”eration Wizard allows developers to create new forms quickly that are bound to data files and/or SQL result sets via record sets. From there, AppBuilder quickly creates a working application. Rapid development has never been so easy!

Overview

The secret to the wizardry behind the Form Gen Wizard is leveraging [BBjRecordSet](#) controls, first added to the BBj® language back in 2003. Take time to refresh your memory of this powerful language concept by reviewing the foundational [BASIS International Advantage](#) articles on record sets referenced on page 22.



It is important to know that a record set definition does not need to rely on a data dictionary definition and can be as simple as defining a string template for an existing BASIS [MKEYED](#) file.

The Form Gen Wizard automatically creates controls and labels for each field the developer chooses to include in the form. It also adds a navigator, search buttons, and a databound grid, if desired. The Wizard places these controls in a window or child window and allows resizing of all controls via spinners. When added to the form, these controls are bound to the [BBjRecordSet](#) object and allow developers to deploy forms rapidly and as often as expanding applications require.

Create a Simple Form

To give you a clear picture of how the Form Gen Wizard works, follow this personal and straightforward how-to guide. Using the [ChileCompany](#) database, you will discover the power and magic of the Form Gen Wizard.

While I give specifics of how the Form Gen Wizard works, I am relying on your working knowledge of the BASIS FormBuilder and AppBuilder. If you are unfamiliar with these tools, it would be very beneficial to read the BASIS IDE articles referenced on page 22.

We are going to create a new “Master Detail” form in just a few steps using the Form Gen Wizard. First, [create a new resource file](#) in BBj FormBuilder, which is included in the BASIS IDE. Choose **File | New** from the BASIS IDE, select the [BBjResource](#) template and name the new form **CCMasterDetail**.

With the form created and opened for editing, delete the default form and create a new [BBjRecordSet](#). To create a record set in the resource, right-click on the **open_invoice.arc** in the [BBjGui Inspector](#) and choose **add RecordSet**. This displays the wizard that allows you to create the new record set in our resource. Choose an **SQL RecordSet** and apply the following query on the [ChileCompany](#) database:

```
select order_num, cust_num, order_date, ship_date, ship_zone, ship_method,
comment, salesperson, mdse_total, tax_total, frght_total from order_header
```

This creates record set 100 in our resource file. Let’s add another record set using the following query for our [BBjDataBoundGrid](#) control;

```
select line_num, item_num, qty_ordered, qty_shipped, price, disc_pct,
disc_amount, line_weight from order_line
```

After creating the record set, right-click on record **setRecordSet 100** in the [BBjGui Inspector](#) and select **Form Gen Wizard** from the popup menu. The dialog as shown in [Figure 1](#) now appears.

continued...



Robert Del Prete
Quality Assurance
Engineer

Field	Control	Label	BBj GUI Control	Search	Grid
ORDER_NUM	☒	Order Num	BBjInputN	☐	☐
CUST_NUM	☒	Cust Num	BBjInputE	☐	☐
ORDER_DATE	☒	Order Date	BBjInputD	☐	☐
SHIP_DATE	☒	Ship Date	BBjInputD	☐	☐
SHIP_ZONE	☒	Ship Zone	BBjInputE	☐	☐
SHIP_METHOD	☒	Ship Method	BBjInputE	☐	☐
COMMENT	☒	Comment	BBjInputE	☐	☐
SALESPERSON	☒	Salesperson	BBjInputE	☐	☐
MDSE_TOTAL	☒	Mdse Total	BBjInputN	☐	☐
TAX_TOTAL	☒	Tax Total	BBjInputN	☐	☐
FRGHT_TOTAL	☒	Frght Total	BBjInputN	☐	☐

Figure 1. Default Form Gen Wizard dialog menu

Modify the Form

Let's explore some of the many options available in the Form Gen Wizard.

First, it is a good idea to rename the form, so change the name to **Chile Company Master/Detail Form**. Change **Columns** to **2**. This will generate the controls in two rows. Change **Label Width** to **70**. See **Figure 2**.

By default, the Wizard includes the option **Generate a navigator and databound controls**. Adding a **BBjNavigator** to the window provides an easy way to page through the records available from the SQL query. Another option is **Generate controls onto a child window**. This allows you to include the form in an already-created window and then adds that generated form as a child window.

The Form Gen Wizard displays all fields based on the SQL statement that we included earlier and by default, selects them to appear in the form. In our example, unselect the ORDER_NUM field since the navigator will use it. Our changes to the default settings are shown in **Figure 2** and our form starts to look like **Figure 3**. For display purposes, you can reorder the fields using the arrow buttons located under the list of fields, as desired.

Now, click [OK] and the Form Gen Wizard creates the form and adds the labels, fields, and BBjNavigator to the form. During this process, the Wizard binds the controls to the BBjRecordSet object.

Using FormBuilder, we can modify the resultant form and its controls like any other resource. Double-click on the newly created form in the BBjGui Inspector to load it into FormBuilder. Note that all of the record set bindings are complete and all controls appear on the form in a logical layout. Change the size of the window to a height of **335** and a width of **495**. Also, change the bound field of the Navigator from CUST_NUM to ORDER_NUM using the **boundfield** property under the Navigator's properties.

Move the controls and labels in the second column as shown in **Figure 3**. Add a **BBjGroupBox** around these controls titled **Order Master Information**. Add an **Order Num:** label next to the **BBjNavigator** and a second **BBjGroupBox** below the first to accommodate the **BBjDataboundGrid**.

continued...

Field	Control	Label	BBj GUI Control	Search	Grid
ORDER_NUM	☐	Order Num	BBjInputN	☐	☐
CUST_NUM	☒	Cust Num	BBjInputE	☐	☐
ORDER_DATE	☒	Order Date	BBjInputD	☐	☐
SHIP_DATE	☒	Ship Date	BBjInputD	☐	☐
SHIP_ZONE	☒	Ship Zone	BBjInputE	☐	☐
SHIP_METHOD	☒	Ship Method	BBjInputE	☐	☐
COMMENT	☒	Comment	BBjInputE	☐	☐
SALESPERSON	☒	Salesperson	BBjInputE	☐	☐
MDSE_TOTAL	☒	Mdse Total	BBjInputN	☐	☐
TAX_TOTAL	☒	Tax Total	BBjInputN	☐	☐
FRGHT_TOTAL	☒	Frght Total	BBjInputN	☐	☐

Figure 2. Edited version of the Wizard's default settings

Figure 3. Resized controls and new group boxes

Form Wizard Global Settings

Form Title: Form

Grid Width: 455 Grid Height: 110

Columns: 1 Control Height: 20

Label Width: 100 Control Width: 130

Horizontal Gap: 10 Vertical Gap: 5

☐ Generate a navigator and databound controls

☒ Generate controls onto a child window

Form Wizard Fields

Field	Control	Label	BBj GUI Control	Search	Grid
LINE_NUM	☐	Line Num	BBjInputN	☐	☒
ITEM_NUM	☐	Item Num	BBjInputE	☐	☒
QTY_ORDERED	☐	Qty Ordered	BBjInputN	☐	☒
QTY_SHIPPED	☐	Qty Shipped	BBjInputN	☐	☒
PRICE	☐	Price	BBjInputN	☐	☒
DISC_PCT	☐	Disc Pct	BBjInputN	☐	☒
DISC_AMOUNT	☐	Disc Amount	BBjInputN	☐	☒
LINE_WEIGHT	☐	Line Weight	BBjInputN	☐	☒

OK Cancel Help

Figure 4. Edited Form Gen Wizard for the BBjDataBoundGrid

Chile Company Master/Detail Form

Order Master Information

Order Num: [Navigator] [Next] [Previous]

Cust Num: [Text Box] Comment: [Text Box]

Order Date: 07/31/07 Salesperson: [Text Box]

Ship Date: 07/31/07 Mdse Total: [Text Box]

Ship Zone: [Text Box] Tax Total: [Text Box]

Ship Method: [Text Box] Frghrt Total: [Text Box]

Order Detail Information

Line Num	Item Num	Qty Ord...	Qty Ship...	Price	Disc Pct	Disc Am...	Line We...

Figure 5. Form with the new grid

The second record set that we created previously will allow the generation of the databound grid with the columns required for the application. The Form Gen Wizard can also create the **BBjDataBoundGrid**. Right-click on the second record set in the BBjGui Inspector and choose Form Gen Wizard. Deselect the **Generate a navigator and Databound controls** option and choose a **Grid Width of 455** and a **Grid Height of 110**. Select the **Generate controls onto a child window** option. In the **Form Wizard Fields**, deselect all of the check boxes in the **Control** column and check all of the controls in the **Grid** column, **Figure 4**. Since we are only going to use the resulting child window, we do not need to modify the **Form Title**. Click [OK] to generate the new form.

Next, delete the extraneous form and drag the child window into the main form. Move it into the second group box we previously created. Change the child window association name to **Childddbgrid** so that we can reference it in the Init block. The updated form appears in **Figure 5**.

Create the Application

Now, we can convert the form into a running application via AppBuilder. Right-click on the **.arc** file in the Filesystems pane and choose **Create AppBuilder File**. We want the **.gbf** file to appear in the same directory so use AppBuilder's default settings for the project.

To complete the application, let's add a few more items in AppBuilder. Double-click on the newly created **.gbf** file to open up the project in AppBuilder. Register for a **WIN_CLOSE** event on the form and add a release to the **WIN_CLOSE** subroutine.

Next, add the Init event to registered events and add the following code to get the record set and record data for later use:

```

declare BBjNavigator          gb__navigator!
declare BBjRecordSet          gb__recordset!
declare BBjRecordData         gb__recorddata!
declare BBjDataBoundGrid      gb__dbgrid!
declare BBjRecordSet           gb__orderLineRS!
declare BBjWindow              gb__window!

REM - Get the recordset and recorddata for later use
gb__window! = bbjapi().getSysGui().getWindow(gb__sysgui_fin.current_context)
gb__childwindow! = bbjapi().getSysGui().getWindow(gb__win.Childddbgrid)
gb__navigator! = cast(BBjNavigator, gb__window!.getControl(100))
gb__recordset! = gb__navigator!.getTargetRecordSet()
gb__dbgrid! = cast(BBjDataBoundGrid, gb__childwindow!.getControl(100))
gb__orderLineRS! = gb__dbgrid!.getBoundRecordSet()

gosub Sync_Detail_Grid

```

continued...

Create a new code block for the application called **Sync_Detail_Grid** to synchronize the BBJDataBoundGrid to the current record as follows:

```
rem ' -----
rem ' Sync_Detail_Grid
rem ' -----

Sync_Detail_Grid:
rem ' Retrieve the current record:
gb_recorddata!=gb_recordset!.getCurrentRecordData()

rem ' Get the order number
orderNumber$ = gb_recorddata!.getFieldValue("order_num")

rem ' Detach the grid and close the orderline recordset
gb_dbgrid!.unbindRecordSet()
gb_orderLineRS!.close()

rem ' Recreate the orderline recordset based off of the new order number
connect$ = "jdbc:basis:localhost:2001?database=ChileCompany&user=admin&pwd=admin123"
modes$ = ""
sql$ = "select line_num, item_num, qty_ordered, qty_shipped, price, disc_pct, "
sql$ = sql$ + "disc_amount, line_weight from order_line where order_num=" + orderNumber$
gb_orderLineRS! = bbjapi().createSQLRecordSet(connect$, modes$, sql$)

rem ' Attach the DB Grid to the new orderline recordset
gb_dbgrid!.bindRecordSet(gb_orderLineRS!)

return
```

We also need code to handle the navigator events. Choose the **Navigator** in the form and add **NAV_FIRST**, **NAV_LAST**, **NAV_NEXT**, and **NAV_PREVIOUS** to registered events. The default code that AppBuilder includes will suffice but we need to un-remark two statements and add our code block for syncing the grid as shown in Figure 6.

continued...

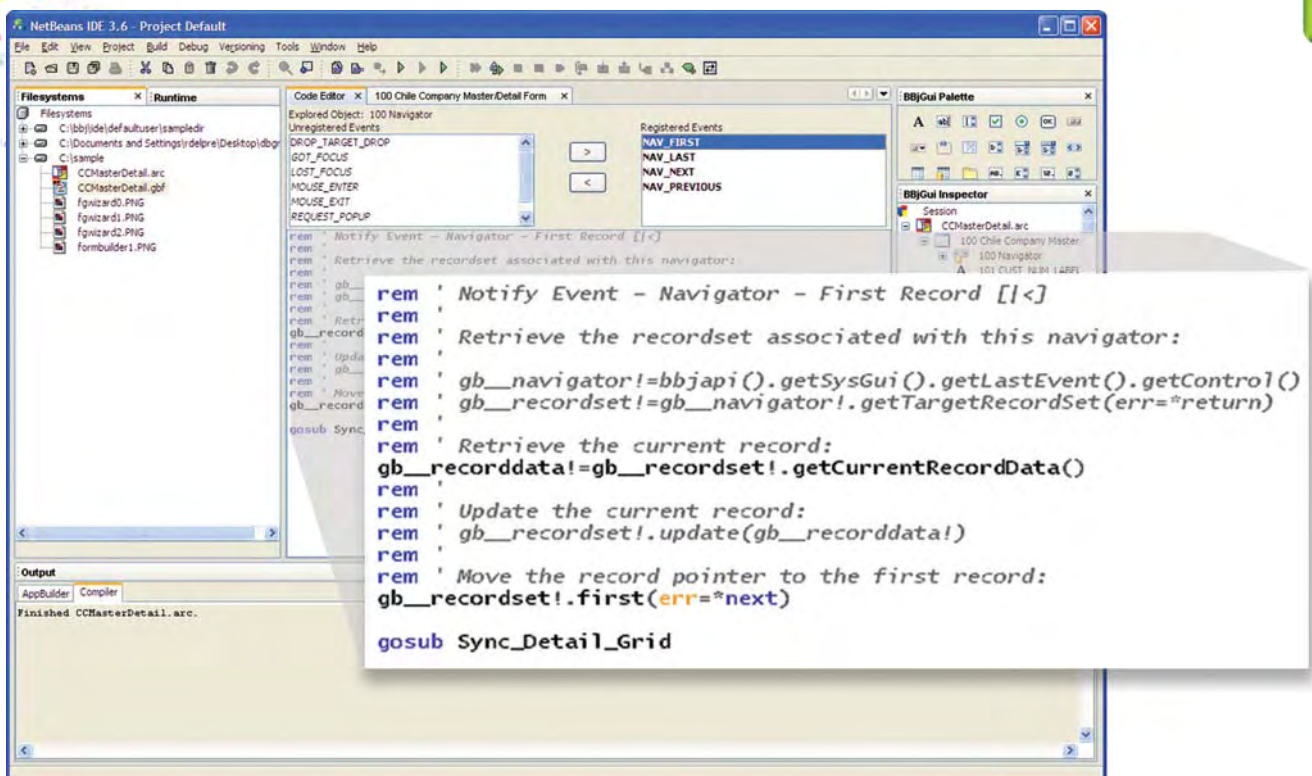


Figure 6. Modified **NAV_FIRST** code block

Once you have added the appropriate blocks of code, build and execute the application via AppBuilder. The resulting application appears in **Figure 7**.



Chile Company Master/Detail Form

Order Master Information

Order Num: 3076111

Cust Num: 000058 Comment:

Order Date: 08/03/93 Salesperson: CAU

Ship Date: 08/04/93 Mdse Total: 86

Ship Zone: 97 Tax Total:

Ship Method: UPRD Frght Total: 8

Order Detail Information

Line Num	Item Num	Qty Ord...	Qty Ship...	Price	Disc Pct	Disc Am...	Line We...
1	000016	3	3	1.75	0	0	0.08
2	000002	12	12	6.5	0	0	2.72
3	000005	1	1	2.75	0	0	0.11

Figure 7. Chile Company Master/Detail Form

Results

The newly generated application displays order information based on order number. The navigator is functional and all of our **BBjInputE** and **BBjInputN** fields are bound to a record set allowing the form to display the selected fields from the database for each record. The databound grid displays the Order Detail Information.

Summary

In just a few steps, we created a functional application using the Form Gen Wizard. The form includes all the data we requested with appropriately created labels and fields as well as the databound grid and navigator. The best part of all, it took very little time and minimal effort to create. 



For information on record sets, read

Why Use the BBjRecordSet?

www.basis.com/advantage/mag-v8n1/recordset.pdf

Using the BBjRecordSet

www.basis.com/advantage/mag-v7n3/bbjrecordset.pdf

For information on the BASIS IDE, read

AppBuilder: The BASIS IDE Gets RAD GUI Development Integration

www.basis.com/advantage/mag-v10n1/appbuilder.pdf

The State of the IDE

www.basis.com/advantage/mag-v9n2/ide.pdf

For information on databound controls, read

BBj Databound Controls

www.basis.com/advantage/mag-v7n3/databound.pdf



Finding and uniting the industries best talent with the best companies is our passion.

Business Basic Professionals . . .

Speak today with a member of our team about our Career Management Program. Because of our highly specialized matching process, we help professionals find immediate positions as well as achieve their long term career objectives.

call us today
(352) 569-9203

or visit us at
www.3dtek.com

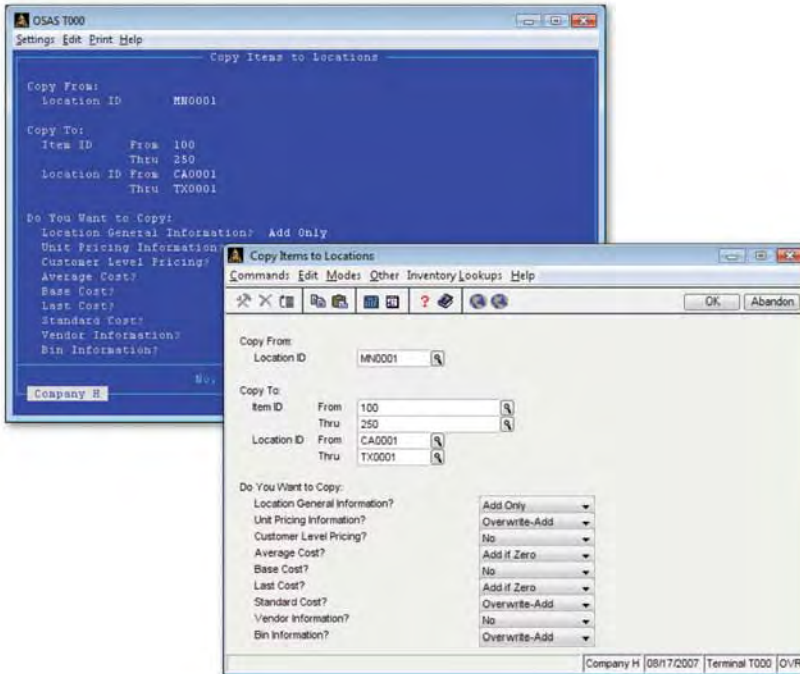


Employers . . .

By spending hundreds of hours to identify and pre-qualify candidates to your unique criteria, we are able to provide a short list of candidates who are highly skilled, but most important, who are also excited about pursuing your opportunity.



Accounting and Business Software for three decades !



Business management:

- > Business Intelligence & Analysis Reporting
- > Executive Dashboards
- > Robust Suite
- > Industry Specific

Integrated suite of Modules include:

- > Accounting
- > Distribution
- > Manufacturing
- > Construction
- > Executive reporting
- > Plus hundreds of 3rd party products

Developers:

Free Source Code for both graphical and character BBj®, PRO/5® & Visual PRO/5®

Ask about our Partner Programs

For a free web demo or a reseller kit call Darlene

1-800-328-2276 ext. 5719

or visit our web site: www.osas.com

RCG Uses Marketing Skills to Achieve OSAS "Top Dog" Status

by Randy Ennis

Like many BBx® developers, I started my career as a computer programmer writing Business BASIC applications. In February 1983, my employer sold Rexon computers with a whopping 5 MB for data storage, 64K system RAM, bundled with Rexon Accounting Software (RACS) and IDOL. The RACS software, which was actually Open Systems Accounting Software (OSAS) version 1.0, was a basic accounting package. With no vertical packages, we had to write the code to meet the customer's needs in order to close the deal. As a result, we developed solutions for a wide array of clients: point of sale, distribution, service, collections, property management, construction, communication companies, and even produce growers.

Our business structure has changed quite a bit. In 1988, I bought in as a partner of [Response Computer Group, Inc.](#) (RCG). There were several acquisitions (we actually purchased RECAP from Alpha Micro in the mid 90's), buy-outs, and eventually I became the sole owner. Today, RCG is a full service computer company with a staff comprising BBx programmers, OSAS installation and support, and IT technicians. We service and sell Linux and Microsoft servers, routers, networking, IP/PBX phone systems, and OSAS accounting software.

As a Top 25 OSAS dealer during the past four years, we were successful, but spread across too many areas. I attended many BASIS TechCons and Open Systems Inc. (OSI) OSAS reseller conferences, listening to various

presentations advising us to "find your niche," "focus on a vertical market, do strategic planning, and develop a marketing plan," and "keep your customers on maintenance." This stuff all sounded great, however I could never find the time to implement any changes.

At TechCon2004 in New Orleans, I attended Nico Spence's Sales and Marketing Workshop that was an intensive and very worthwhile eight-hour day. Some of the material was a review for me, but everything we covered was right on target. The most significant concept I took from this course was the need to "work *on* the business" not just work *in* the business. Our company had always been focused on customer service and pushing the technology envelope to develop innovative solutions. In the past, it had never felt okay to "waste" time creating a business plan when I could have been writing code or designing something new. However, Nico helped me realize that strategic planning should be a priority! I made time for it and it paid off.

[I] realize that strategic planning should be a priority! I made time for it and it paid off.

continued...



Randy Ennis
President
Response
Computer Group



Left to right: Paul Lundquist, OSI VP of Sales; Joe Pint, OSI Regional Sales Rep; Lisa Soulopoulos, OSI Regional Sales Manager; Curtis Ennis, RCG Programmer; Randy Ennis, RCG President/CEO; Kathy O'Connor, OSI Regional Sales; Michael Bertini, OSI CEO



Response
Computer
Group, Inc.

Partnership

Language/Interpreter

DBMS

Development Tools

System Administration



Bottom row from left: Billy Clauges, Programmer/Analyst; Faith Ennis, Controller; Randy Ennis, President; Curt Ennis, Programmer/Analyst **Back row:** Bryan Eshelman, MCSE/Technician; Colin Ennis, Web Designer; Pat Coulter, General Manager; Garrette Slonacher, MCSE/Technician; Randy Wilson, Programmer/Analyst

In September 2005, OSI teamed up with Bret Romney, President of Ascend Strategies, to launch the Learning Partner Program (LPP) that offered both hands on workshops and monthly conference calls focused on marketing education. One focus of LPP was strategy planning, where partners define and set clear business goals with target completion dates and assigned responsibilities. They dedicated other areas of the program to project management, process improvement, marketing, and team building. LPP is currently expanding the program to include a partner roundtable.

We stepped up efforts to keep our customers current on software maintenance.

Lasting change requires a steady, consistent effort. Slowly RCG began to implement changes derived from what I had learned at these workshops. We stepped up efforts to keep our customers current on software maintenance (BASIS'

SAM and OSI's CES), improved client communications, and focused our software development efforts. We routinely blasted e-mail and fax announcements, published newsletters, and mailed event reminder postcards. We also leveraged our relationship with key business partners that included OSI, BASIS, Payment Processing Inc., and other BBx resellers. This may sound like a cliché but "Use your resources!" really works.

All the hard work paid off when Open Systems, Inc. recognized RCG as their #1 Partner for 2006-2007's top sales performance at their annual conference in June 2007. A major factor of this success was our ability to upgrade four large mattress manufacturers to OSAS 7.0. This project had some unique challenges and a very rigid timeframe. It required us to realign our resources and juggle a few other projects, but everything fell into place and we delivered the project on time.

I am very proud of our entire team for helping us to achieve this great accomplishment. RCG is proof that if you invest quality time to work on your business, set well-defined goals, make strategic plans, and increase your sales and marketing efforts, you will see positive results. Visit Response Computer Group at www.rcgweb.com. 



Attend the Sales and Marketing Workshop on Nov 7th - it conveniently follows TechCon2007 and is free to any TechCon attendee!
Register at www.basis.com/events/techcon2007

Continue the Journey... BBx*cellerate* to

DEVELOPMENT TOOLS

- See rapid application AppBuilder functionality integrated into the BASIS IDE
- Learn how to use the IDE for BBj®, Visual PRO/5®, and PRO/5® development tasks
- Boost productivity with the IDE's project management capabilities

LANGUAGE/INTERPRETER

- Review the CUI extensions to the BBj language that implement enhancements to BBx®
- Discover how to improve programmer productivity and enhance applications developed with BBj, the newest generation of BBx
- Learn to easily adopt the popular look-and-feel of Vista, Apple, or Linux platforms

DATABASE MANAGEMENT

- Experience the new BASIS RDBMS file type
- Hear about the all new SQL engine

SYSTEM ADMINISTRATION

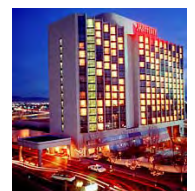
- Learn how to implement the myriad of deployment tasks
- Capitalize on increasingly popular hardware configurations such as clustering and redundancy
- Deploy in cross-platform environments – mix and match Vista, Linux, and Mac

PARTNERSHIP & TRAINING

- Network with colleagues and partners
- Capture hands on knowledge and real-world application during post conference training
- Learn how to leverage the newest generation of BBx in your marketplace during the post conference marketing workshop

POINTS OF INTEREST

Albuquerque Marriott



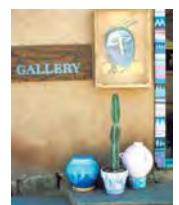
Albuquerque Marriott is a well-known restaurant and registration company.

STARTUP, "the first"

STARTUP
ALBUQUERQUE and the
Personal Computer Revolution

dinner and a stroll through the city that began in Albuquerque.

A visit to Santa Fe



adventure stop in movie Santa Fe acclaimed

Save \$100

Register before September 23 at www.basis.com

the Next Generation



FE



INTEREST

Marriott hosts this year's conference. Located in Albuquerque's upscale uptown district, the Marriott is walking distance to some of the finest shops and restaurants in the southwest. Conference registration includes a two nights' stay and complimentary high-speed internet connection.

The "New Mexico Museum of Natural History and Science" dedicated to the microcomputer, located in the New Mexico Museum of Natural History and Science, will host the Monday night event. Attendees and companions will enjoy a walk through time as they see the innovative origins of Albuquerque and went on to change the world.

A walk via the Turquoise Trail is an optional feature for conference companions. A morning walk in Madrid, the featured NM town in the recent National Geographic Wild Hogs, precedes an exquisite lunch in the Plaza de Fe and walk through the plaza of this highly-anticipated, award-winning city.

www.basis.com/events/techcon2007

TechCon2007 BASIS Training

Extend your stay in Albuquerque and take the opportunity to expand your knowledge at hands-on instructor-led BASIS training courses. Map out your personal training experience by selecting the one-day à la carte courses that best guide your journey, or choose a three-day track for your trek!

À LA CARTE *Choose one course on any or all three days*

- November 7**
- ☐ Sales and Marketing Workshop
 - ☐ The BASIS IDE Overview and Update
 - ☐ System Administration and Deployment Strategies
- November 8**
- ☐ The BASIS DBMS including SQL, Stored Procedures, and Triggers
 - ☐ Application Interoperability Using WebServices, JSP, Servlets, and JavaBBjBridge
- November 9**
- ☐ RAD Development with AppBuilder and FormBuilder in the IDE
 - ☐ Object-oriented Programming with BASIS

BARISTA DEVELOPMENT TRACK

- November 7-9** ☒ Developing in Barista™ – BBj's new data dictionary-driven RAD GUI Tool

BASIS DEVELOPMENT SUITE TRACK *Includes*

- November 7** ☒ The BASIS IDE Overview and Update
- November 8** ☒ The BASIS DBMS including SQL, Stored Procedures, and Triggers
- November 9** ☒ RAD Development with AppBuilder and FormBuilder in the BASIS IDE

SPECIALIZED TRAINING TRACK *Includes*

- November 7** ☒ Administration and Deployment Strategies
- November 8** ☒ Application Interoperability Using WebServices, JSP, Servlets, and JavaBBjBridge
- November 9** ☒ Object-oriented Programming with BASIS

Rev up your engine and register today!

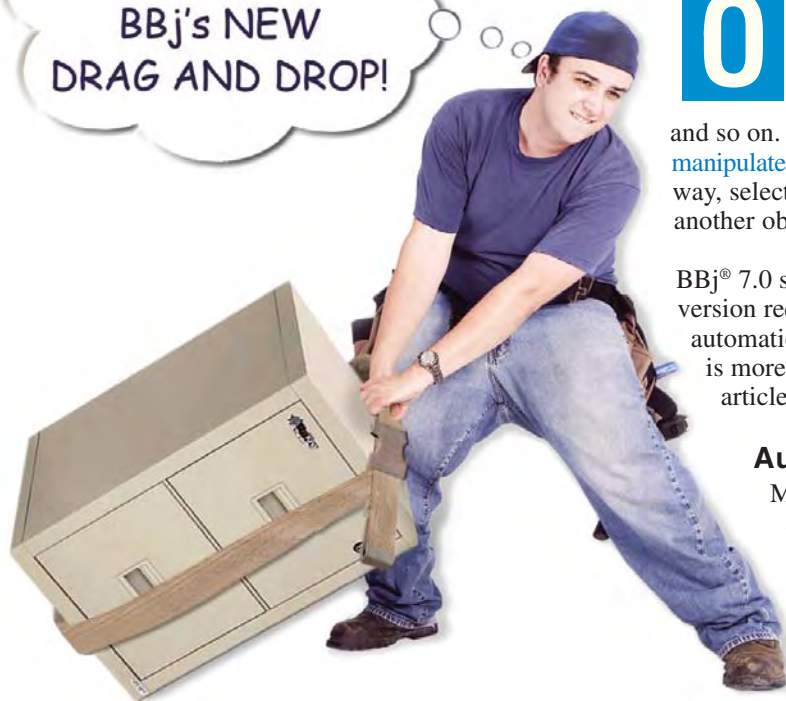
www.basis.com/events/techcon2007/registration



Leap From 'Cut and Paste' to 'Drag and Drop'

By Jim Douglas

BOY, I WISH
I HAD
BBj's NEW
DRAG AND DROP!



One of the hallmarks of event-driven GUI applications is that the user is in control. The developer sets up an environment consisting of various windows and GUI objects and the user interacts with the environment at will, clicking a button here, typing a bit of text there, and so on. **Drag and drop** gives the user even more freedom to **manipulate** the GUI environment directly, in a natural and intuitive way, selecting data from one GUI object and dragging it to another object.

BBj® 7.0 supports two versions of drag and drop. The basic version requires no special programming; it is enabled automatically for all text-based controls. The advanced version is more flexible, but involves a bit of programming. This article will look at the basic (automatic) version first.

Automatic Drag and Drop

Most BBj 7.0 GUI controls support a basic form of drag and drop that can copy chunks of text from one BBj GUI control to another, or between a BBj GUI control and an external application, like a word processor.

This version of drag and drop is similar to the standard Cut, Copy, and Paste functions supported by all text controls. The following sample contains no special drag and drop code; it demonstrates that BBj 7.0 text controls inherently support drag and drop:

```
rem ' Default Drag-and-Drop

sysgui = unt
open (sysgui)"X0"

sysgui! = bbjapi().getSysGui()

title$ = "Drag-and-Drop as Cut-and-Paste"
window! = sysgui!.addWindow(200,200,300,130,title$, $00010003$)

editbox! = window!.addEditBox(101,10,10,280,30,"")
input! = window!.addInputE(102,10,50,280,30,"")
reset! = window!.addButton(1,200,90,90,30,"Reset")

gosub reset

CALLBACK(ON_BUTTON_PUSH,RESET,sysgui!.getContext(),reset!.getID())
CALLBACK(ON_CLOSE,APP_CLOSE,sysgui!.getContext())

PROCESS_EVENTS

APP_CLOSE:
RELEASE

RESET:
  editbox!.setText("The quick brown fox jumps over the lazy dog.")
  input!.setText("")
RETURN
```



Jim Douglas
Software Engineer
Contractor

The user can move text from one control to another, equivalent to cutting text from the source control and pasting it to the target control. The CUT drag action is indicated by the special cursor shown in **Figure 1**.

continued...

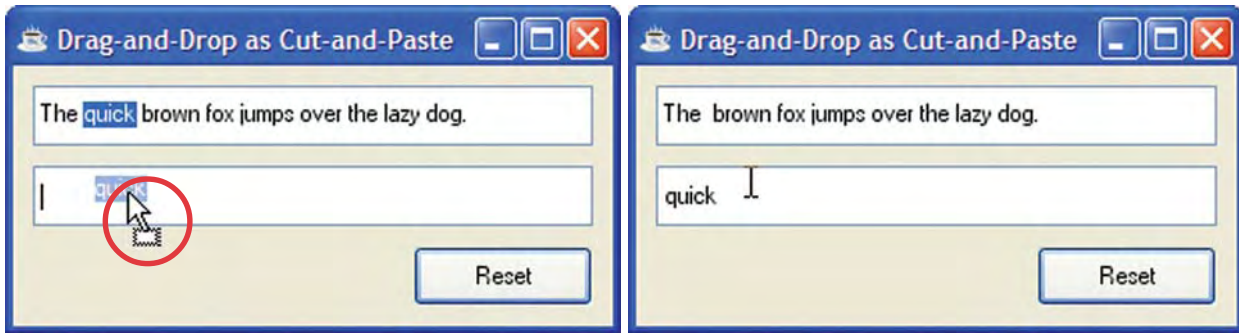


Figure 1. The cut cursor in action and the drop result

By pressing the CTRL key while dragging, the user can copy the text instead of moving it. The COPY drag action is indicated by the cursor shown in **Figure 2**.

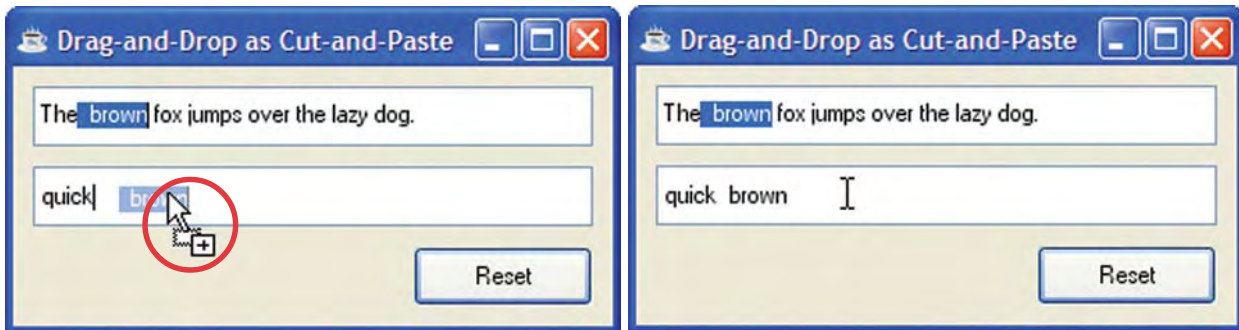


Figure 2. The copy drag cursor in action and the drop result

The [Apple Human Interface Guidelines](#) recommend using drag and drop as a supplementary user interface technique, not as the only way to accomplish a given task. In this sample, drag and drop gives the user a simple alternative to cut and paste. For a more complete sample, see [Default Drag and Drop Example](#) in the online documentation.

Programmable Drag and Drop

The basic version of drag and drop support handles the typical case, where a user wants to drag a chunk of text from one control to another. For more advanced or specialized applications, the developer will want to implement a more customized approach. This [sample program](#) implements a selection dialog with two lists: available options on the left and selected options on the right. The user can select one or more options from the **Available Options** list and click a button to transfer those options to the **Selected Options** list as shown in **Figure 3**.

continued...

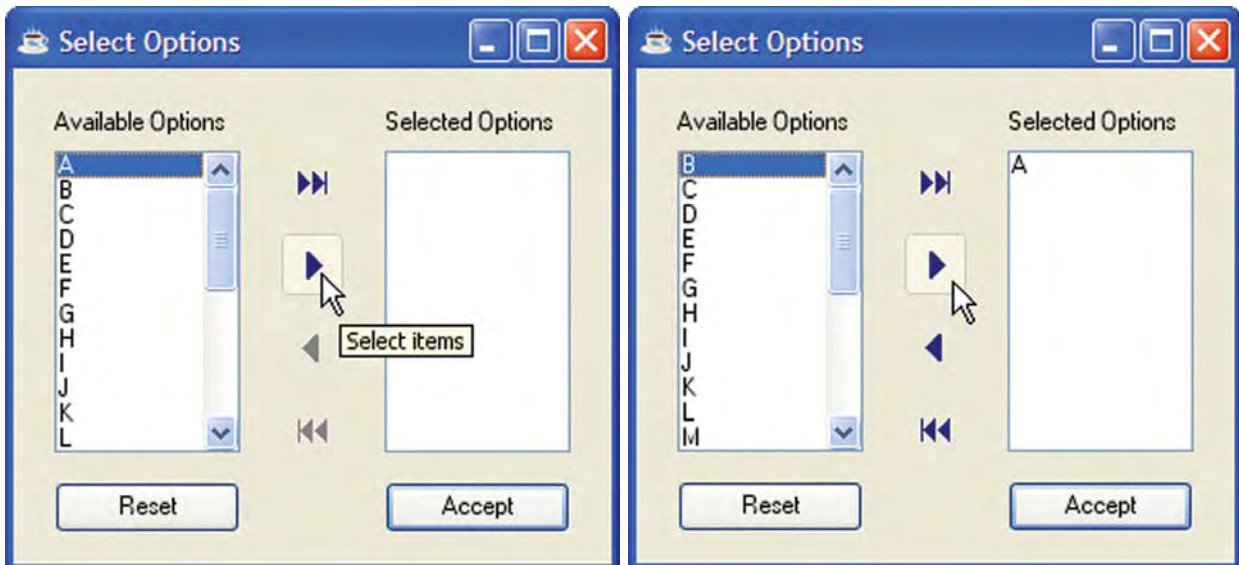


Figure 3. Selecting from a list with a click and its results

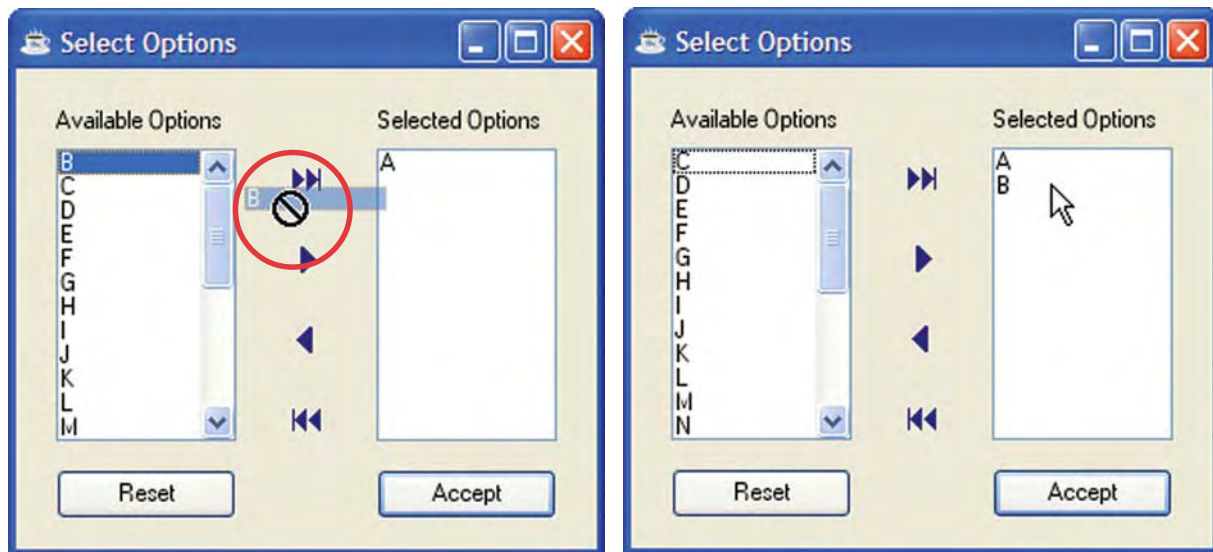


Figure 4. The move cursor indicating it is positioned over a "no drop zone" on its way to the final drop point

Drag and drop provides a more direct and intuitive way to perform the same task. The user clicks on one or more options, as in the first example, but instead of clicking a button to move the option from one list to the other, he or she simply drags and drops the chosen options to the **Selected Options** list. See **Figure 4**.

As the user drags data across a window, the drag cursor is continually updated to indicate whether data can be dropped on a particular control, and if so, how it would transfer. Notice in the first screen of **Figure 4** that as the mouse drags the data across the gray center, the cursor changes to a universal "no" symbol to indicate it is over a control that cannot accept the data. When the cursor appears as an arrow and a single rectangle as shown in the first screen of **Figure 5**, it indicates that dropping the data at that location will move the data rather than copy it.

Users can also use drag and drop to rearrange data within a single GUI control. For example, if the order of items in the selection list is significant, the developer can choose to implement drag and drop functionality to enable the user to reorder items within a list by dragging them. See **Figure 5**.

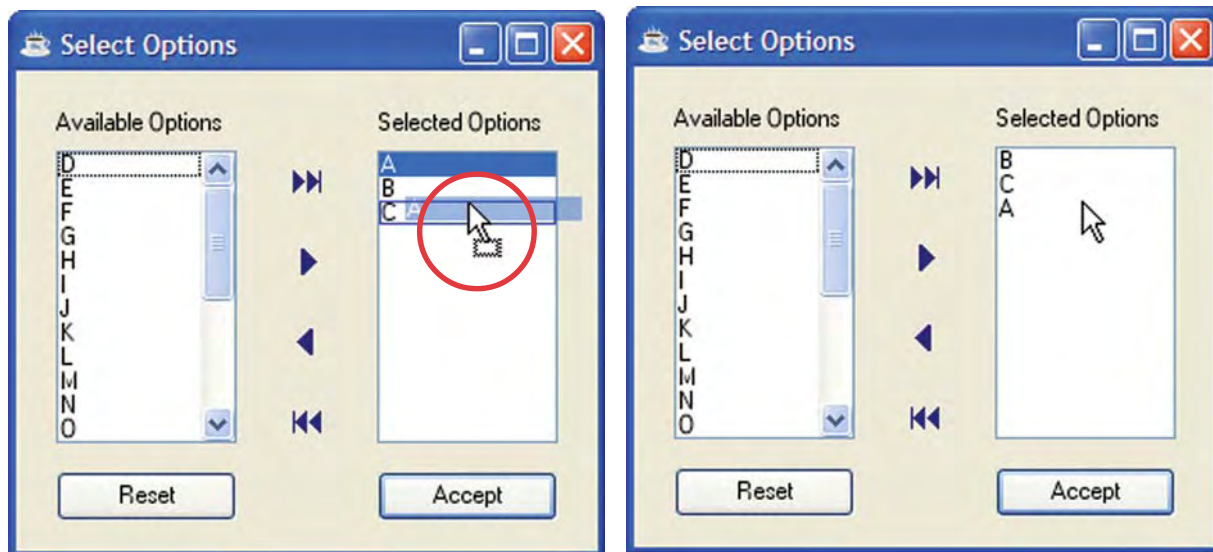


Figure 5. The move cursor in action, dragging and dropping to reorder the list

Drag and Drop Events

When a control registers for the `BBjDropTargetDropEvent` using the `ON_DROP_TARGET_DROP` callback, BBj switches from the default drag and drop behavior to the programmable behavior. In this sample program, registering these two callbacks causes BBj to fire `BBjDropTargetDropEvents` in response to data being dropped onto either of these controls:

```
CALLBACK(ON_DROP_TARGET_DROP,DROP_ITEMS,sysgui!.getContext(),available!.getID())
CALLBACK(ON_DROP_TARGET_DROP,DROP_ITEMS,sysgui!.getContext(),selected!.getID())
```

The corresponding event handler can be a bit overwhelming at first glance, but after breaking it down line-by-line as it appears in the next chart, the code is easily understood.

continued...

ON_DROP_TARGET_DROP EVENT HANDLER	COMMENTS
Drop_Items:	
<code>event! = sysgui!.getLastEvent()</code>	Get the BBjDropTargetDropEvent .
<code>dragSource! = event!.getDragSource()</code>	Get the drag source BBjControl.
<code>dropTarget! = event!.getControl()</code>	Get the drop target BBjControl.
<code>sel! = event!.getSelection()</code>	Get a BBjVector containing the selection list of items that were dragged from the drag source. The intent of this sample program is to transfer data between BBjListBox controls, so that the selection is a list of selected rows in the same format as BBjListBox::getSelectedIndices .
<code>location = event!.getDropLocation().get(0) + 1</code>	Get the drop location within the drop target. The intent of this program is to transfer data between BBjListBox controls, so we know that the drop location is a zero-based row number.
<code>if dragSource! = dropTarget! then</code>	If the user is moving items within a list, then...
<code>if sel!.size() <> 1 then return</code>	For the sake of simplicity, only allow a single item to be moved at once within a list.
<code>sel = num(sel!.get(0))</code>	Get the index of the dragged row .
<code>if location = sel then return</code>	If the user dropped the row onto itself, do nothing.
<code>item\$ = dragSource!.getItemAt(sel)</code>	Get the text from the dragged row.
<code>if location > sel then</code>	If the data was dropped below its original location, then...
<code>dropTarget!.insertItemAt(location,item\$)</code>	...insert it at the new location
<code>dragSource!.removeItemAt(sel)</code>	...and remove it from the original location
<code>else</code>	...otherwise, the data is being dropped above its original location, so:
<code>dragSource!.removeItemAt(sel)</code>	...remove it from the original location...
<code>dropTarget!.insertItemAt(location,item\$)</code>	...then insert it at the new location.
<code>endif</code>	...end of moving within a list.
<code>else</code>	...otherwise, the user is moving from one list to another:
<code>if sel!.size() then</code>	If the selection includes at least one row, then...
<code>for i=0 to sel!.size()-1</code> <code>item\$ = dragSource!.getItemAt(num(sel!.get(i)))</code> <code>dropTarget!.insertItemAt(location+i,item\$)</code> <code>next i</code>	Copy each of the selected rows from the drag source to the drop target...
<code>for i=sel!.size()-1 to 0 step -1</code> <code>index = num(sel!.get(i))</code> <code>dragSource!.removeItemAt(index)</code> <code>next i</code>	...and remove them from the drag source.
<code>endif</code>	End of move from one list to the other.
<code>canSelect = available!.getItemCount()</code> <code>selectItems!.setEnabled(canSelect)</code> <code>selectAll!.setEnabled(canSelect)</code>	Enable or disable the selection buttons based on whether there are any items left in the "Available" list.
<code>canDeselect = selected!.getItemCount()</code> <code>deselectItems!.setEnabled(canDeselect)</code> <code>deselectAll!.setEnabled(canDeselect)</code>	Enable or disable the deselection buttons based on whether there are any items currently in the "Selected" list.
<code>endif</code>	
<code>return</code>	

The [Drag and Drop Events Example](#) includes a more complex event handler for dragging and dropping between arbitrary controls. [continued...](#)

Drag/Drop Actions

In the first sample program, dragging text from one text control to another causes it to be moved. The user can override this default behavior by pressing the CTRL key while dragging.

In the second sample program, data is always moved. It is not possible to select copy because that wouldn't make sense in this particular application.

In the selection list program, the programmer overrides the default drag and drop actions, specifying that MOVE is the only supported action for all drags and drops involving these controls:

```
available!.setDragActions(sysgui!.ACTION_MOVE)
available!.setDropActions(sysgui!.ACTION_MOVE)
selected!.setDragActions(sysgui!.ACTION_MOVE)
selected!.setDropActions(sysgui!.ACTION_MOVE)
```

A third kind of drag/drop action, the ACTION_LINK that appears as a , is rarely used. It indicates that the application should establish a linkage between the drag source and the drop target. There is no general definition of what this means in practice; the interpretation of the link action is determined on a case-by-case basis by each application program.

Drag/Drop Data Formats

These sample programs work with plain text, but a user can transfer almost any kind of data with drag and drop. The `BBjDropTargetDropEvent::getText` method returns plain text and the `BBjDropTargetDropEvent::getData` method returns all available formats. The `MIME type code` specifies the data format; common values are "text/plain" and "text/html." Data dragged from a `BBjListBox`, `BBjTree`, or `BBjGrid` transfers as both plain text and HTML-formatted text, while data dragged from BBj text controls only transfers in plain text format. The [Drag and Drop Events Example](#) demonstrates how to process non-text data, such as dragging a color from a `BBjColorChooser` control.

Everything is Negotiable

Drag and drop always involves a negotiation between two controls. When the user initiates a drag, the source control packages its data in a variety of formats. At the same time, it establishes the actions (MOVE, COPY, LINK, or any combination of the three) that it is prepared to support. When the user drags the data over a potential drop target, the two controls enter into a negotiation.

- If the drag source offers a data format and action that the drop target is prepared to accept, the cursor indicates the selected action – ACTION_MOVE ACTION_COPY or ACTION_LINK.
- If the drag source does not offer a data format and action that the drop target is prepared to accept, the cursor is set to ACTION_NONE.
- If the drag source is prepared to offer the data via more than one action that is supported by the drop target, the user can choose between the available options by pressing a modifier key while dragging (SHIFT for MOVE, CTRL for COPY, or CTRL+SHIFT for LINK). The [Drag and Drop Actions Example](#) demonstrates the effects of the various drag/drop actions.

Drag/Drop Types

The developer can impose additional restrictions on the type of data that can be dragged between various controls. For example, the list controls in the selection dialog sample only accept drops from each other. This is done by specifying drag and drop type codes for each of the list controls as follows:

```
OptionListBox$ = "OptionListBox"
OptionListBox! = bbjapi().makeVector()
OptionListBox!.add(OptionListBox$)

available!.setDragType(OptionListBox$)
available!.setDropTypes(OptionListBox!)
selected!.setDragType(OptionListBox$)
selected!.setDropTypes(OptionListBox!)
```

The [Drag and Drop Types Example](#) in the online documentation demonstrates more complex rules for controlling drag and drop behavior between multiple controls.

Summary

This article is a brief introduction to drag and drop in BBj. Drag and drop enables the developer to enhance BBj applications with rich and flexible new features, giving the users a new way to transfer data between GUI controls. For more information and sample programs, see [BBj Drag and Drop](#) in the online documentation. 

Unleashing the Power of SPROCs Without SQL

By Jeff Ash

Partnership

Language/Interpreter

DBMS

Development Tools

System Administration

Stored procedures are a powerful and very flexible part of the SQL engine inside BBj®. They give developers a way to embed callable program logic in the database, thereby making it available to BBj and any other third party ODBC/JDBC application. One of the features stored procedures provide is the ability to return a result set to the client application. Prior to BBj 7.0, developers could return any result set based on an SQL query from an SQL channel. Now, stored procedures can return an SQL record set or the new SQL-less Memory Record Set, further expanding the power and flexibility of stored procedures. This article focuses on how to create a memory record set from information generated by an SQL-less query. Additionally, this article illustrates how to create a stored procedure programmatically.



What is an SQL Result Set?

Stored procedures have the option of returning nothing, a single value, or an entire result set of data to the client application. An SQL result set consists of zero or more records where each record is comprised of one or more columns of information such as name, address, phone number, etc.

What is a Memory Record Set?

The new Memory Record Set, created from [BBjAPI: createMemoryRecordSet](#), makes it possible for a stored procedure to return a result set made up of arbitrary information that the stored procedure builds in any way the developer sees fit. For example, a stored procedure could use existing business logic with READ RECORD file operations to gather its information. Then, this information can be placed into a memory record set and sent back to the client application. The client application sees this information in the same way as a result set created from an SQL query. Thus, any BBj or third party ODBC/JDBC application can use it just like an SQL SELECT result.

Creating and Using a Memory Record Set

Creating a memory record set requires that the developer define the layout of the records to be contained in the record set. To do this, use the following string template:

```
rs! = BBjAPI().createMemoryRecordSet("ITEM_NUM:C(6)")
```

Now, adding values to the Memory Record Set is easy. This code snippet inserts a single record into the Memory Record Set:

```
data! = rs!.getEmptyRecordData()
data!.setFieldValue("ITEM_NUM", "000234")
rs!.insert(data!)
```

If the record set has more than one column defined in it, simply add more [setFieldValue](#) calls to set the additional column values. When the stored procedure has finished populating the record set, it must return that record set to the SQL engine so that the client application can receive the information. To do this, simply call the new [setRecordSet](#) method on the [BBjStoredProcedureData](#) object.

A Sample Stored Procedure Utilizing a Memory Record Set

The following CREATE PROCEDURE call creates a stored procedure named GET_ITEMS in the ChileCompany sample database that demonstrates using READ RECORD statements to populate a Memory Record Set, and returns that record set to the client application as an SQL result set. To create the stored procedure, execute the SQL code in **Figure 1** using the [BBj Enterprise Manager](#). Right-click on the ChileCompany database, select the "Execute SQL" option, and paste in the code.

continued...



Jeff Ash
Software Engineer

```

CREATE PROCEDURE get_items (prod_cat CHAR(2) IN) RESULT_SET
{ _BEGIN_ }
    REM Open the file
    chan = UNT
    OPEN (chan) "C:\Program Files\basis\demos\chiledd\data\ITEM"

    REM Get the parameter value specified by the calling program
    sp! = BBJAPI().getFileSystem().getStoredProcedureData()
    prod_cat$ = sp!.getParameter("PROD_CAT")

    REM Create a memory record set to hold the results of
    REM our read operations.
    rs! = BBJAPI().createMemoryRecordSet("ITEM_NUM:C(6)")

    REM Iterate over the file and find the items that
    REM have the specified PROD_CAT
    TMPL$ = "ITEM_NUM:C(6),DESC:C(30),PROD_CAT:C(2),STOCK_UOM:C(3),"
    TMPL$ = TMPL$ + "COST:N(12),WEIGHT:N(12),WT_UNIT:C(2),PRICE:N(12)"
    DIM REC$:TMPL$
    while (1)
        READ RECORD (chan, ERR=eof) rec$
        if (rec.prod_cat$ = prod_cat$) then
            REM Found a match, so add it to the Record Set
            data! = rs!.getEmptyRecordData()
            data!.setFieldValue("ITEM_NUM", rec.item_num$)
            rs!.insert(data!)
        endif
    wend

eof:
    CLOSE (chan)

    REM Set the returned result set value to the record set.
    sp!.setRecordSet(rs!)
{ _END_ }

```

Figure 1. Sample stored procedure

Now that the database contains the stored procedure, it is ready for use. To test this stored procedure, simply execute the following SQL CALL statement directly from the Enterprise Manager's SQL Browser dialog:

```
CALL get_items('CH')
```

The results appear as a list of the items in the database with a PROD_CAT of **CH** as shown in **Figure 2**.

Summary

Stored procedures offer a method of executing SQL on the server side of a client-server application, delivering enormous performance benefits when sorting through large volumes of data. Oftentimes, the developer can sort or process that data more efficiently using the fast READRECORD constructs of the BASIS language. Now using memory record sets, developers can more easily reuse existing business logic and data structures created in BBj, in their stored procedures, extending the power of the BBj language to the world of third party applications.

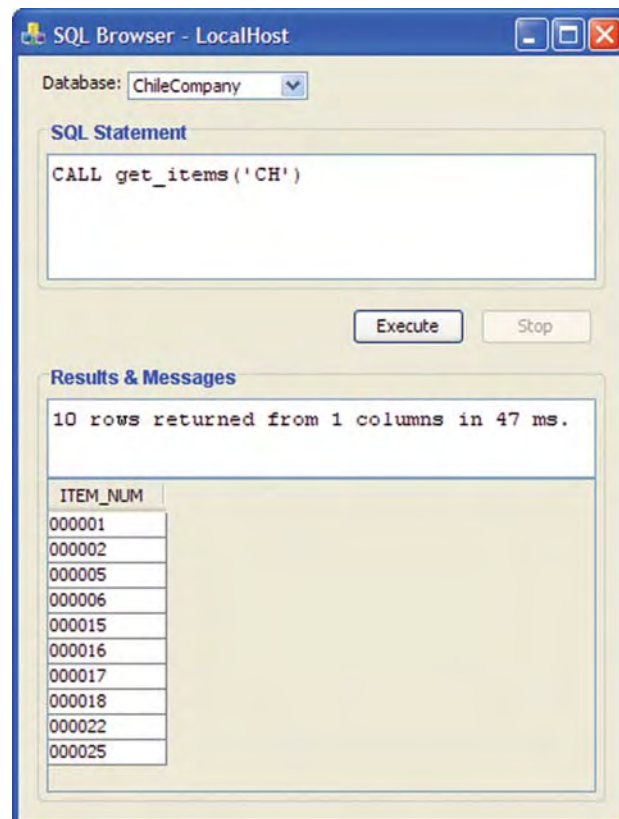


Figure 2. Result of the SQL CALL statement

Desktop Data Delivered

By Adam Hawthorne



While earlier versions of BBj® allowed access to the client's filesystem via a fat client deployment, access in a thin client deployment was not available. Frequently, developers expressed the desire for file access outside of fat client deployment; they wanted flexible and full access to files and directories anywhere in their client-server deployment.

BASIS offers two solutions. New `MODE="CLIENT"` options on the `FILEOPEN` and `FILESAVE` functions provide easy interaction with the client's filesystem from the traditional standard dialogs. However, when the application requires a customized dialog, or more flexibility choosing files and directories, enhancements in BBj 7.0 expand the user's ability to interact with the client computer. The new `BBjFileChooser` control and `BBjClientFileSystem` object allow unprecedented interaction with the client's filesystem. This article focuses on the latter solution and the new freedom developers have to access information, anywhere, regardless of whether it is server- or client-based.

Introduction

The `BBjClientFileSystem` provides a straightforward interface to handle system-wide file actions. Using `BBjClientFileSystem`, the developer can obtain a specific file from the root filesystem, the user's home directory, or an absolute path to the desired file. Another new object, the `BBjClientFile`, provides the API for individual files. Its API models the Java class called `java.io.File`, which represents the name of a legal file on the client's filesystem. `BBjClientFile` objects may represent non-existent files or directories, and the API provides methods to create a non-existent file or directory. Most of the methods on the `BBjClientFile` match their respective methods on an instance of a `java.io.File`. These include self-explanatory methods such as `canRead`, `canWrite`, `isDirectory`, `exists`, etc. For more information about the other methods, refer to the `BBjClientFile` in the online documentation at www.basis.com.

Specifically, these four `BBjClientFile` methods allow data exchange between the client and the server:

RETURN VALUE	METHOD
string	<code>BBjClientFile::copyFromClient()</code>
void	<code>BBjClientFile::copyToClient(string fileName\$)</code>
string	<code>BBjClientFile::getContents()</code>
void	<code>BBjClientFile::setContents(string contents\$)</code>

To give these methods context, let us put them to work.

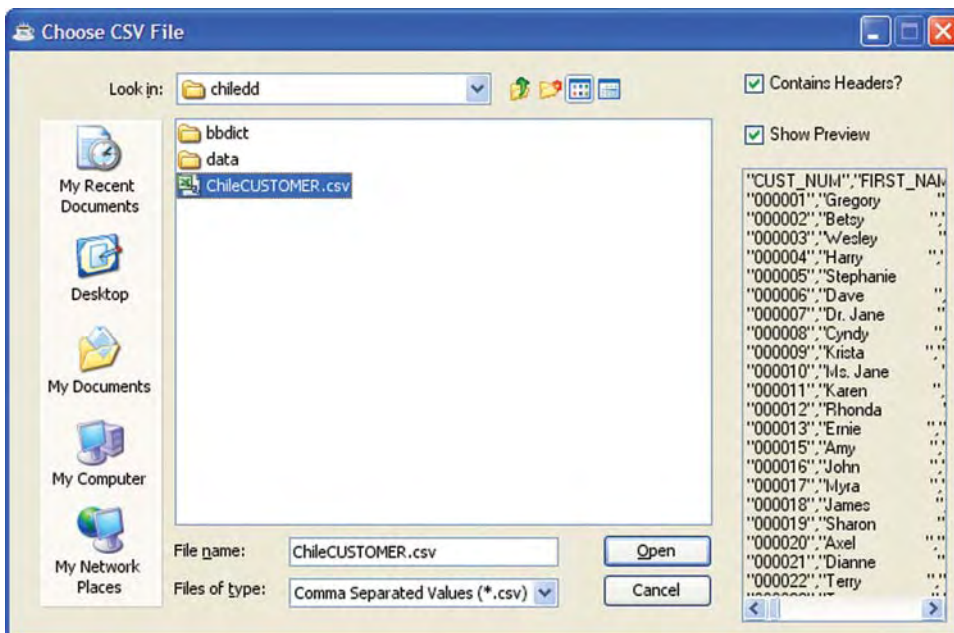


Figure 1. The customized file chooser dialog with file preview capabilities

Sample Program

To demonstrate the `BBjClientFileSystem`, look at the sample program `clientFSArticle.src`, downloadable from the URL noted at the end of this article. It allows users to select a CSV file on their own computer to display in a grid. It uses the aforementioned `getContents` and `copyFromClient` methods to obtain the contents of the file in two different contexts. When showing a customized client-side file chooser dialog (see **Figure 1**), the sample uses `getContents` to show a preview of the CSV file. It retrieves the data from the client and places it into the preview control on the file chooser dialog displayed on the client. This allows the user



Adam Hawthorne
Software Engineer

continued...

to see the contents of the file before selecting it to determine whether the file has a header row. This is a classic example of going above and beyond the standardized file open dialog; it extends the basic dialog with preview capabilities and other user options to provide all of the necessary information to display the CSV file.

Once the user selects a file, the application uses the `copyFromClient` method to copy the file from the client's machine to the server machine, thus obtaining the data in the file. When the sample calls `copyFromClient` on the `BBjClientFile` variable **file!**, BBj retrieves the data in the file named by the **file!** variable, copies it to a temporary file located on the server, and returns the name of the temporary file. When the copying operation completes, the program parses the file into a **CSVTable** object and displays the table in a grid on the client system, as shown in **Figure 2**. When the program terminates, it automatically deletes the temporary files that this method created. To retain a copy of the file on the server after the end of the program, use the **RENAME** verb to move the file to a different location.

CUST_NUM	FIRST_NAME	LAST_NAME	COMPANY	BILL_ADDR1	BILL_ADDR2	CITY
000001	Gregory	Baldrake	Bogus Stuff	8508 Manitoba NE		Albuquerque
000002	Betsy	Heebink	Betsy Inc	1499 378th		Madison
000003	Wesley	Osborn	Tasteful Gifts	3222 Gregory Street NE	P.O. Box 1997	Seattle
000004	Harry	Chuckie	Long Island Inc	9406 Sand Pebble Court		Palm Springs
000005	Stephanie	McIntyre		1712 Elm Terrace		Modesto
000006	Dave	Strum	FishLand	6441 Concord Street		Hagerstown
000007	Dr. Jane	Booker	Greencastle Moose Farm	3898 Arcadia Trail East		Greensboro
000008	Cyndy	Sikes	Gourmet Giftware	421 Lawton Ave	Suite 2	DeFond
000009	Krista	Nugent	Utopian Cattle	421 Navajo Trail NE		Albuquerque
000010	Ms. Jane	Ghast	Massage Land	1941 Wallace Avenue		March
000011	Karen	Dennison	Rexall Drugs	3626 Ash Avenue NW	Apt 13	Albuquerque
000012	Rhonda	Frenchi	Bing Bong Gift co	5523 Russell Drive		Albuquerque
000013	Ernie	Augustine		3120 Carlos Court SE		Albuquerque
000015	Amy	Alcott	Misha Gifts	107 Cody Trail NW		Albuquerque
000016	John	Roberts	Bimland Ketchup	196 Ridge Road		Bellevue
000017	Myra	Tomlin	Condiments & Thensome	397 Windsong Drive		Houston
000018	James	Thomas		P.O. Box 989		Albuquerque
000019	Sharon	Reeves		3107 84th Avenue		Seattle
000020	Axel	Pasternak		5405 Shelby Court		Albuquerque
000021	Dianne	Wellesley		6802 Cassidy Lane		Santa
000022	Terry	Green		1631 Santa Maria		Middle
000023	Tom	Sanderson		6610 Spanish Elm Drive		El Paso
000024	Grover	Evanston		1120 Salem Avenue		Boise
000025	Doris	Williams		2900 Vista Del Rey		Pueblo
000026	John	Tuscany	Monkey Shines	220 Rio Grande	Apt 16	Los Angeles
000027	Sally	Larimore		P.O. Box 2100		Portland
000028	Kathy	Nightengale		7121 College Ave		Ithaca
000029	Meghan	Abbott		8057 Tarheel		Raleigh

Figure 2. The contents of the CSV file displayed in a grid

The majority of the code in this sample application comes from parsing the CSV file. The source code that shows the file chooser dialog and copies the file from the client, including the code to preview files and respond to events, is relatively small. The supplemental program, **csvLibrary.src**, provides an object-oriented model for a simple table and a class with methods to produce the table from a file.

Summary

BBj 7.0 gives developers who leverage these new API tools much greater flexibility in interacting with the client computer. They can store or retrieve configuration, data files, or even customized programs on or from the client computer using `BBjClientFileSystem` and `BBjClientFile`. Programs can deliver PDF files, spreadsheets or any other useful files directly to the user's computer at a location the user chooses using the `BBjFileChooser` in client mode and can retrieve configuration or other files as needed. Now, enjoy the freedom to place files where you need them using the `BBjClientFileSystem`. Deliver data, right to your desktop!



Download the sample code from
www.basis.com/advantage/mag-v11n1/clientfilesystem.zip

Solving the Data Warehousing Dilemma with BBJ's Newest DBMS Features

By Nick Decker

BASIS continues to add several groundbreaking features such as triggers and ESQL Tables to the BBJ® database management system. Although some of these terms and concepts may not be new to many developers, the challenge always seems to be realizing the full potential of such features. Ironically, developers sometimes fail to exploit the most potent features fully because they are so versatile. Triggers fall squarely into this category – they are so incredibly useful that we continue to discover new ways to utilize them in working systems to solve age-old problems. One such problem that plagues many is the implementation of a standardized and efficient data warehousing system. A secondary problem is the choice of RDBMS to house the data. This article focuses on the nuances of data warehousing and how to go about solving the associated dilemmas.

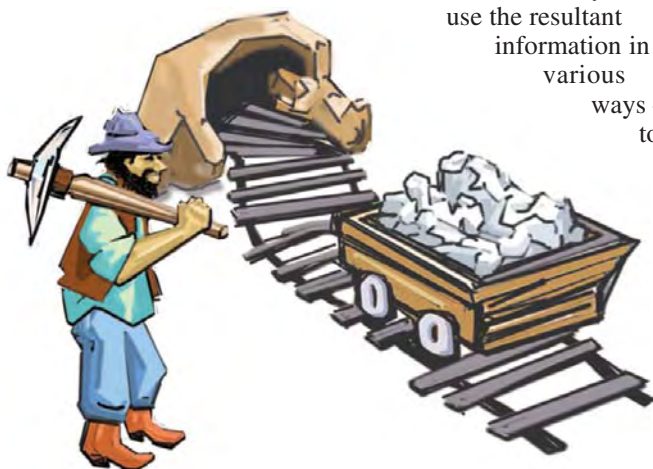
Data Warehousing

Before digging into the details of applying triggers and ESQL's relational Tables, let us step back for a moment and look at the concept of data warehousing and its function. Simply put, a data warehouse is the main repository or storage of company data. This data is both current data used to maintain the normal day-to-day operations of the company as well as historical data accumulated over the life of the company. Both banks of data are critical to the life and function of the company. Without access to the current data, there is no way to process orders, invoice customers, or receive payments. Without access to the historical data, management would be unable to analyze past sales to determine profitability and whether the company is meeting its forecast goals.

Data Mining

In order to make use of the data at their disposal, companies perform "data mining." Similar to the miners digging deep into the earth in hopes of extracting a rare and valuable diamond, corporate analysts perform complex SQL queries and sift through mountains of

data to retrieve valuable information. Analysts use the resultant information in various ways – to



predict and forecast future sales, to see if recent business decisions such as whether a new advertising campaign has affected sales, and so on.

As necessary as data mining may be, it comes at a cost. The data retrieval and analysis usually places a heavy load on the system, sometimes slowing down daily operations to a crawl. Needless to say, this situation is impracticable and requires a solid solution. Companies have proven that data mining is indispensable, but how do they accommodate the load on the system? As with most complex problems, no simple solution is readily available. However, after a lengthy "tour of duty" in the trenches, engineers finally unearthed a standardized, multi-part structured solution to the data mining dilemma.

Solution – Part 1

The first step seems quite obvious – make an offline copy of the operational data system. At first blush, this seems to be a perfect solution. The data analysts can tie up the offline system running elaborate queries and the daily operation of the company will be completely unaffected. However, after implementing this step, engineers noted several outstanding problems. As usual, the solution is never this simple.

For starters, data retrieval from the offline system was still very slow, even though it was running on a separate system from operations. Analysts wrongly assumed that once they separated the historical data, their queries and data extraction would speed up tremendously. A closer look revealed the true nature of the slowness – non-normalized data. Most of the company's data was located in antiquated table formats. They found tables that did not



Nick Decker
Engineering
Supervisor

continued...



make good use of indexes, formats that allowed for multiple data types in a single field, fields comprised of multiple data elements, and so on. Most modern-day database administrators would shudder at such a design, but the truth is that many administrators often ignored data normalization with little consequence – that is until SQL entered the picture.

SQL ushered in a new era of data independence. Armed with third party reporting tools such as Crystal Reports, all of the company's data was fair game. Users, even those without any database experience, were suddenly performing ad-hoc queries left and right. Since the software application design did not optimize the underlying tables for data retrieval from generic reporting software, the queries would sometimes take forever to complete. This was a great disappointment as everyone depended on the speedy execution of the fixed, specialized queries written into the company's software. The solution to this glaring problem was clear – normalize the data.

Solution – Part 2

Since the offline database was separate from the company's operational dataset, normalizing the data was not an insurmountable task. Even though the offline database was to contain the same data, it did not require the same data organization. Developers were free to restructure the data in a normalized format without

having any impact on the operations software. After completing the normalization, analysts were thrilled to see their queries execute orders of magnitude faster. The celebration ended quickly, though, as they discovered their next problem: the

offline database was out of date. They made no attempt to synchronize the offline version of the data with the version that operations used, so by now, the offline version was missing thousands of records and was no longer a complete and current set of data.

Solution – Part 3

Analysts easily recognized the next step in the solution – they had to keep the offline database in synch with the company's main dataset. They accomplished this by copying all of the data from operations to the data warehouse at regular intervals. However, this too proved to be problematic. Even though the data was *more* current than before, it still was not *actually* current. Once users got a taste of accessing data that was relatively current, it left them wanting more. A 24-hour lag on the operations database was surely an improvement, but soon everybody wanted real-time data and instant updates.

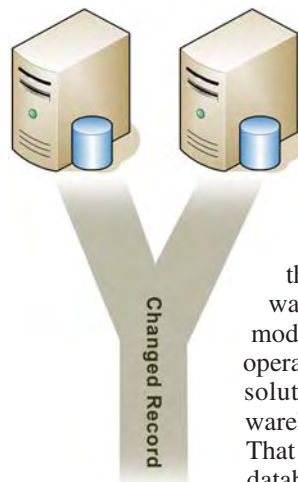


Additionally, the synchronization process turned out to be extraordinarily time consuming. As part of the synch process, nightly cron jobs kicked off programs that removed all of the records from the offline database. The next step involved painstakingly copying

every single record from the operations database to the data warehouse. This process, while effective, was terribly inefficient because users do not modify a majority of the records in the database. Typically, they add new records and modify recent records, but most historical data remains unchanged on a day-to-day basis. The synch process never differentiated between modified and non-modified records, so it dutifully removed and recopied the same inactive data night after night. Not only did this take *a lot* of time, it ended up taking *most* of the time required to synch.

Solution – Part 4

Analysts solved this final hurdle in the data warehousing project by rethinking the synchronization process. Instead of blindly removing and copying every record in every file, they needed to synchronize in a more intelligent manner. The most effective way to perform the synch involved copying only the small percentage of records that changed instead of every individual record. Additionally, in order to satisfy the need for real-time updates, they would have to update the data warehouse immediately after a user modified or removed a record in the operations dataset. Simply put, the solution was to update the data warehouse on a transactional basis. That is, whenever the daily operations database changed, the same change should appear in the data warehouse.



Problem Solved!

This last tweaking to the system solved the problem for everybody. Operations continued unaffected by the offline data warehouse. Not only did developers solve their performance degradation problem by moving a copy of the data offline, but their legacy application did not require altering to accommodate new data structures and table layouts. Analysts were also pleased. They were able to quickly execute intricate queries and process an abundance of data on a live dataset.



Using BBj for Data Warehousing

Over the years, many BASIS developers have implemented their own data warehousing solution by keeping an offline copy of their corporate data. The typical solution closely follows our historical overview. The first step is to create a new relational and fully normalized database for offline storage of the data. The next step is to copy all of the operational data over to the normalized database on a nightly basis. The process, however, has always stopped there. The developers were stuck at the fourth part of the solution; they lacked real time updates to keep the data live and eliminate the costly bulk copies. With the advent of BBj's new database trigger capability, developers can now complete the final step to solve the data warehouse dilemma.

continued...

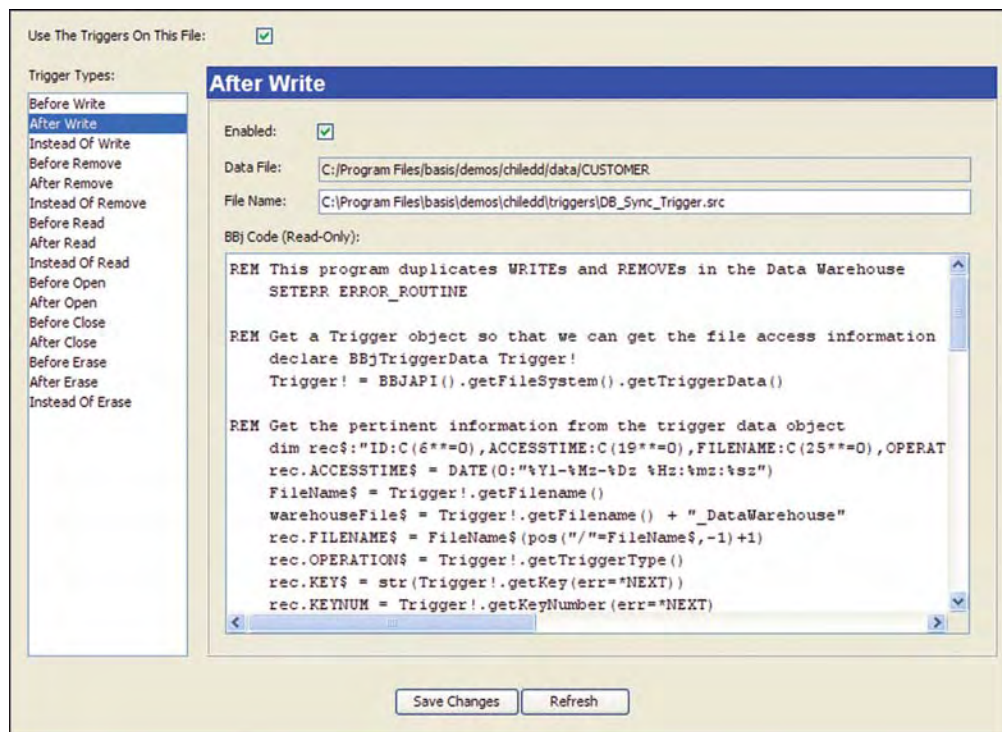


Figure 1. Trigger definition that synchronizes the data warehouse files in real time

Trigger Overview

Because triggers allow developers to execute a given program automatically whenever a particular file system activity occurs, triggers are perfect for keeping two databases in synch. As mentioned earlier, the goal is to update the data warehouse table the moment a user modifies or removes a record in the operations table. Triggers handle this with ease as the developer can configure the system to run a specialized synchronize program when one of these events occurs on the operational database. See **Figure 1**.

More specifically, the developer defines “After Write” and “After Remove” triggers to fire on an operations table. Now, whenever anyone modifies the table in any way, the respective trigger fires and executes the developer’s synchronization code. This program instantiates a **BBjTriggerData** object and then retrieves pertinent information such as the name of the modified file, the type of file operation, the affected record, and so on. Armed with all of the essential details regarding the table access, the program would then make the corresponding change to the appropriate warehouse table; either making the same modification to the same record or removing the record as required.

Improving Data Access Time

Another new BBj database feature, ESQL Tables, further enhances the data warehousing solution. ESQL Tables offer many advantages, one of the most noteworthy is that they outperform SQL access to traditional **MKEYED** files. They also support standard SQL data types so on-the-fly data mapping for third party SQL reporting tools is a thing of the past. Additionally, they are field-based rather

than record-based, so it is possible to read and write distinct portions of a data record without having to access the record in its entirety. Their support for variable-length records is also a tremendous bonus as it translates to enormous savings in disk space, record access, and reduced backup times. All of these features add up to a blazingly fast back-end database for the offline data warehouse, optimized for speed and space. It also provides fast and standardized access from an SQL-enabled application.

Summary

As the BASIS DBMS advances over time, new and exciting features such as triggers and relational ESQL Tables unearth numerous possibilities that were never available in the past. While a feature like database triggers may seem relatively simple at first glance, its straightforwardness can be deceptive. Triggers have dozens of potential use cases, from audit trails and access restriction to our most recent use case – real-time database synchronization.

By combining BBj’s diverse triggers with its relational ESQL Tables, developers can implement an accelerated and powerful data warehousing solution. Triggers ensure that the various tables remain synchronized with one another and completely eliminate the need for bulk copies that tie the system up for hours overnight. Relational ESQL Tables ensure that analysts can sift through a plethora of information in the quickest way possible. By combining both of these new technologies, BASIS developers can finally solve the data warehousing dilemma! 



Using Triggers to Maintain Database Integrity
www.basis.com/advantage/mag-v10n1/triggers.html

BASIS Puts Triggers to Work
www.basis.com/advantage/mag-v10n1/workingtriggers.html



Catching the XML Wave

How I Used BBJCustomObjects to Utilize XML

By **Brian Hipple**

As a BBx® Web services developer, I have longed for XML support in the BASIS Products Suite. While BBJ® does not directly support XML, it does support embedded Java code and the many Java XML packages that create and manipulate XML documents. Therefore, the great news is that I, or any BASIS developer, can create BBJ Custom Objects to simplify the rather complex interaction with Java's XML packages and extend functionality not provided by the Java implementation. Any BBJ application can then use these custom objects in a traditional fashion.

What is XML?

XML, or EXtensible Markup Language, is a cross-platform software- and hardware-independent tool for transmitting information.

XML is much like the familiar HTML, Hyper Text Markup Language, although XML's main function is to describe the data rather than format and display data as HTML does. Both languages use tags, which are keywords surrounded by <> to convey information for the data. The tags used in HTML documents are predefined and the creator of an HTML document can only use tags that are defined in the HTML standard (<p>, <h1>, etc.). In XML, the tags are not predefined; the document author must define them. **Figure 1** is an example of an XML document in the BASIS IDE XML Editor.

Why use XML?

XML technology allows developers to future-proof their applications and services, ensuring that the data they are managing today will easily adapt to future needs. With XML, developers can transform today's data into new data formats as they emerge or build new applications and services to accomplish new tasks with existing data. XML also serves as a common platform for transmitting and sharing data between disparate systems, allowing the rapid development of Web services that query, retrieve, and share data among many sources. (<http://www.sitepoint.com/books/xml1/>)

XML is not just a universal data format; it is also a universal library of tools that includes XSLT, XPath, XQuery, and DOM. Developers use these tools for transforming documents between otherwise incompatible formats, presenting data in particular sample styles and formats, querying data from data sources, and manipulating data in a hierarchical, tree-like form. XML standards are implemented in every major programming language, ensuring developers that they can always access their most important asset in the future – their data. (<http://www.sitepoint.com/books/xml1/>)

Why use BBJ Custom Objects?

BBj version 6.0 introduced BBJ Custom Objects that enable developers to make use of object-oriented programming. My decision to use a BBJ Custom Object for my XML functionality enabled me to reap the benefits these objects provide: increased development resources, maintainability, readability, and reusability. I was able to streamline functionality into the most commonly-used methods; therefore, a BBJ developer should be able to easily decipher the methods.

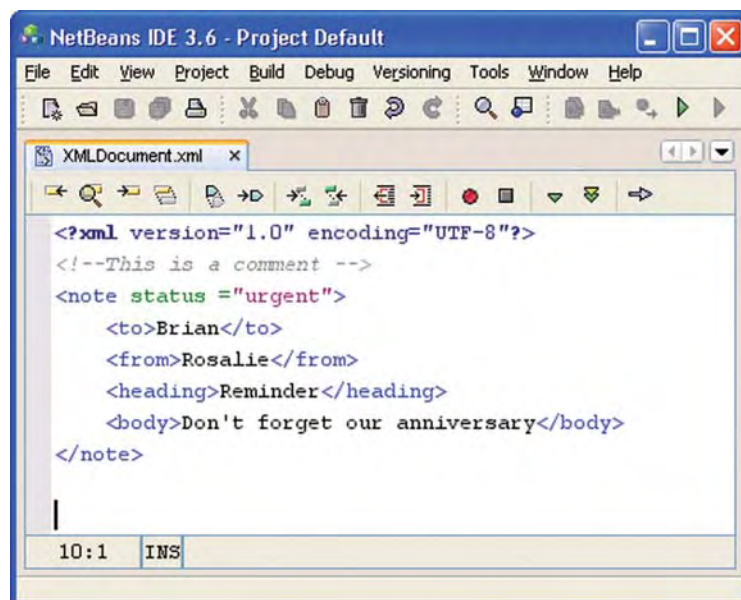


Figure 1. A sample XML document



Brian Hipple
QA Test Engineer
Supervisor

continued...

Reusability is a major advantage of using a BBj Custom Object. Once I thoroughly tested my object, any BBj application could now use this object to access XML documents with the assurance of quality. Incorporating this functionality separately into each application would have required a great deal more coding and testing.

The Custom Object - XMLDoc

The resultant BBj Custom Object name is XMLDoc, which includes multiple constructors that provide for the creation of an XMLDoc object from a new document, an existing document, a string, or from a specified URL. The XMLDoc methods provide many commonly used XML functions such as addRootElement, addElement, setAttribute, addTextNode, and addCommentNode. Along with these methods are the combination of these methods for a macro-type interface which are not provided in the Java XML APIs that includes addElementWithAttribute, addElementWithAttributes, and addElementWithTextNode. Search and destroy methods getAllMatchingNodes and removeAllMatchingNodes will search or remove nodes based on criteria such as a "node name" and "type of node." Also, the output method writeToConsole can write the XMLDoc to the BBj console for debugging purposes and writeFile writes the contents of the XMLDoc to a new file. **Figure 2** shows the XMLDoc source that employs these methods.

```
rem Create a XML document from scratch
xmlDoc! = new XMLDoc()
rootElement! = xmlDoc!.addRootElement("RootElement")
rootElement!.setAttribute("Attribute1Name", "Attribute1Value")
rootElement!.setAttribute("Attribute2Value", "Attribute2Value")
child1Element! = xmlDoc!.addElement(rootElement!, "Element1")
commentNode! = xmlDoc!.addCommentNode(child1Element!, "This is a comment node")
textNode! = xmlDoc!.addTextNode(child1Element!, "This is a text node")
child2Element! = xmlDoc!.addElementWithAttribute(rootElement!, "Element2", "AttributeName", "AttributeValue")
xmlDoc!.writeToConsole()

rem Save the XML document to a file
xmlFileName$ = dir("") + "demo.xml"
xmlDoc!.writeToFile(xmlFileName$)

rem Determine if the XML file is well formed
print "XML file: " + xmlFileName$ + " is well formed = " + str(XMLDoc.isWellFormed(xmlFileName$))

rem Create a XML document from an existing xml file
xmlDoc! = new XMLDoc(xmlFileName$, BBjAPI().TRUE)

rem Create a XML document from an URL
urlString$="http://www.poweredbybbj.com/QAMemos/QAMemos?WSDL"
xmlDoc! = new XMLDoc(new URL(urlString$))
print "XML document created from URL " + urlString$ + ":"
xmlDoc!.writeToConsole()

rem Get all nodes with a name of operation
nodeName$="operation"
nodeVector! = xmlDoc!.getAllMatchingNodes(nodeName$)
print "The following nodes with the name of " + nodeName$ + " were found: "
numNodes=nodeVector!.size()
if numNodes > 0
  for i = 0 to numNodes-1
    node! = cast(Node, nodeVector!.get(i))
    print "node! = ", node!
  next i
endif
```

Figure 2. XMLDoc source code

Using the XMLDoc Object

To exercise the XMLDoc object in an application, I wrote a small BBx program named WeatherXMLDemo.bb. This program passes the URL for a weather site with parameter information that requests the seven-day forecast, in XML format, for any zip code.

This creates an instance of an XMLDoc that can then retrieve the forecast data and display this information in a grid. See WeatherDemo code in **Figure 3** and the results in **Figure 4**.

This program shows just how easy it is to retrieve information from the internet in XML that many Web services provide – and use this data in your application. Simple applications like this weather retrieval program demonstrate the use of XML to transfer information over the Web, but it does not end there. XML is highly adaptable and works well for applications with widely differing degrees of complexity.

continued...

```

rem Create a XML document from an URL
urlString$="http://xoap.weather.com/weather/local/"
urlString$=urlString$+zipCode$
urlString$=urlString$+"?dayf="+str(numDays)
urlString$=urlString$+"&prod="+prod$
urlString$=urlString$+"&par="+partnerID$
urlString$=urlString$+"&key="+licenseKey$
weatherXMLDoc! = new XMLDoc(new URL(urlString$))
gosub DisplayXMLInfo
return

DisplayXMLInfo:
rem Get the city information
cityNodes! = weatherXMLDoc!.getAllMatchingNodes("dnam")
if (cityNodes!.size() < 1)
    response=MsgBox("No weather information found for zip code",0,"Invalid Zip code")
    return
endif
cityInfoEB!.setText(weatherXMLDoc!.getNodeText(cast(Node,cityNodes!.getItem(0))))
rem Get all nodes with a name of day
nodeVector! = weatherXMLDoc!.getAllMatchingNodes("day")
numNodes=nodeVector!.size()
weatherData! = api!.makeVector()
if numNodes > 0
    for i = 0 to numNodes-1
        node! = cast(Node,nodeVector!.get(i))
        weatherData!.addItem(weatherXMLDoc!.getNodeAttribute(node!,"dt"))
        weatherData!.addItem(weatherXMLDoc!.getNodeAttribute(node!,"t"))
        highNode! = weatherXMLDoc!.getNode(node!,"hi")
        weatherData!.addItem(weatherXMLDoc!.getNodeText(highNode!))
        lowNode! = weatherXMLDoc!.getNode(node!,"low")
        weatherData!.addItem(weatherXMLDoc!.getNodeText(lowNode!))
    next i
endif
weatherGrid!.setCellText(0,0,weatherData!)
return
  
```

Figure 3. The WeatherXMLDemo source code

Recently, I consulted with a BASIS customer on their Ford Motor Company Web service that transferred all the information in XML, including parameters for all the methods. It was evident to me they chose XML for future compatibility. For example, if one of the Web service methods should require more information, the XML specification will change for the method, but the method signature does not have to change. As a result, this customer would not need to make any changes to the client application.

Summary

It amazes me that XML has become the standard data format for transferring information over the internet in such a relatively short period or time. While the BBJ Custom Object that I created provides basic XML functionality, my hope is that over time I, or another BBJ developer, will take this object and extend its functionality. For example, we could incorporate the ability for a DTD or XML schema to describe the XML in detail and validate the XML, or the ability to use XML Namespaces to avoid element name conflicts by providing qualifications in the XML documents. Using Custom Objects in BBJ with the strength of the Java XML API gives BBJ developers the ability to harness this data standard and provide information that will thrive and prosper for many years to come. XML puts your data and application in the fast lane! 

Date	Day	High	Low
Jul 13	Friday	87	61
Jul 14	Saturday	92	61
Jul 15	Sunday	91	62
Jul 16	Monday	91	63
Jul 17	Tuesday	89	63
Jul 18	Wednesday	88	62
Jul 19	Thursday	88	61

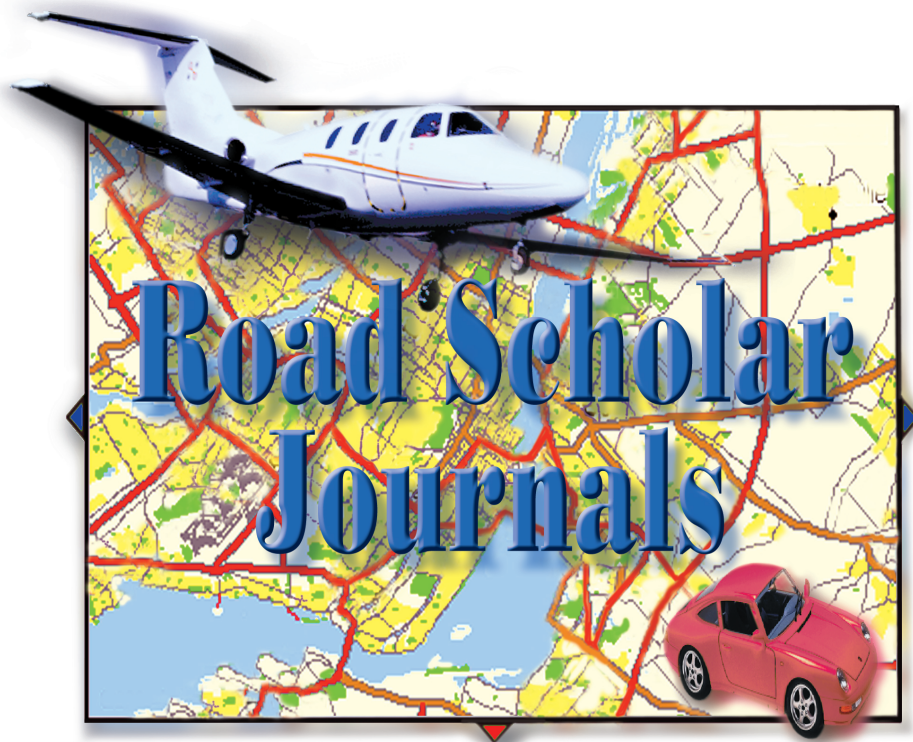
Figure 4. The WeatherXMLDemo application



Download and run the sample referenced in this article at
www.basis.com/advantage/mag-v11n1/XML.zip

A Primer for Using BBJ Custom Objects
www.basis.com/advantage/mag-v10n1/primer.pdf

Applying Custom Objects to Existing Code
www.basis.com/advantage/mag-v10n1/custom.pdf



BBj 7.0 is a hit at Descore Showcase



Descore, the BASIS Canadian Distributor, divided up their most recent product showcase between three cities. Nico Spence and I began the tour in Toronto and Montreal last November and headed west to Vancouver on June 1st to conclude the trek.

Ahhh, late springtime in beautiful Vancouver...not a bad life for BASIS Road Scholars. In addition to the fragrant flowers, there was the smell of Java in the air, or BBj® to be precise, as the presentations focused on this newest version of BBx®.

The presentations started with a review of BBj technology. Nico highlighted the enhanced interface in FormBuilder, the integrated AppBuilder, and the improved performance analyzer. He showcased the new

language features as he discussed symbolic labels, DENUM, CLEARP, and the ability to do a command line search. The attendees learned about the new DBMS features of triggers and stored procedures, and left with ideas on how to implement these capabilities to improve their existing applications.

Nico finished the session by sharing the new features and functionality of BBj 7.0 and giving a sneak preview of what BASIS will cover during the upcoming TechCon2007 in Albuquerque. With parting words and some fond farewells, we reluctantly ended our brief stay in beautiful Vancouver, but with the comfort of knowing that the audience was excited about both current and forthcoming BBx technology. By all measures, our trip was a success.

— **By Laurence Guiney**



Nico Spence presents to the Vancouver audience



Laurence Guiney
Senior Account
Manager



BASIS and OSAS Partners in Profit



his past June 7-9, in downtown Minneapolis, Open Systems Inc. (OSI) personnel were hard at work hosting their annual conference entitled *Partners in Profit*. This forum brings the OSI reseller community together to learn, share ideas, network with their peers, and form partnerships.

It is always a privilege to participate in such a well-organized conference.

Nico Spence, Dr. Kevin King, and I participated as exhibitors and presenters at the conference. We were among a great group of exhibit partners who offered a wide range of vertical and after-market solutions to the OSI community. It was an excellent opportunity for us to talk with resellers about their experiences and successes with OSAS 7.0, and to give them a preview of forthcoming BASIS technology features for their customization efforts.

In addition to exhibiting, Nico and Kevin participated as presenters in two BASIS breakout sessions. They focused on how OSAS resellers can use our current and future BASIS technology to enhance the OSAS suite of applications and make them even more attractive to the existing and prospective users. They demonstrated how to utilize the newest features in BASIS technology effectively, thus giving the resellers more power to address their customer needs.

In the general session, Dave Link, Vice President of OSAS Product Development, presented the audience with the first look at OSAS 7.5, which promises to be a much-anticipated feature-rich release of OSI's flagship product. We are committed to supporting Dave and his team as our strategic partner during their development of OSAS 7.5.

A tradition at the OSI conference is to recognize the top 25 resellers for their sales efforts during the year. The #1 reseller was Response Computer Group (RCG) of Milford, DE. This all-OSAS shop does the right stuff to get the job done, which is why they are such a huge success. Their professional staff provides hardware and software services, specializing in customization and offering many vertical solutions to meet the demands of today's business. RCG is an active BASIS developer and uses BBj tools to get the job done. Everyone at BASIS extends to RCG our hearty "Congratulations, on a job well done!" (Read their story on page 24.)

Likewise, we congratulate Open Systems, Inc. on an excellent conference and say "Thank you, for the opportunity to share our technology vision with your OSAS community."

— by Gale Robledo



Gale Robledo
Account Manager

TechView2007 Travels Cross Country



Chief Marketing Officer, Nico Spence, flew from coast to coast earlier this year to present the 2007 BASIS TECHNical overVIEW. I joined Nico in sunny California and Florida; Amer Child, Sales and Training Specialist, joined Nico for the northern tour in New Jersey and Illinois.

This one-day TECHVIEW2007 seminar provided the opportunity for BBx® developers, end users, and resellers to learn about BASIS technology and experience it through hands-on activities. The day's highlights included the fine points of the BASIS IDE, an overview of the new BASIS DBMS, several system administration features, and the latest BBx product suite.

By all accounts, TECHVIEW was a great success. Nico interacted with all participants and provided ways to gain productivity at all levels of BASIS technology deployment. Everyone left excited and with ideas about how to improve their own mission critical applications. We accomplished our prime communication objectives; to

- **Understand** customer needs,
- **Educate** customers, and
- **Introduce** customers to features that they and their clients could implement immediately and profit from the many benefits.

Face-to-face time with you, our customer, is always so valuable. It is the best way to clearly understand your needs and the challenges that you encounter, both of which are so important to our mutual success. Speaking for all of us in the BASIS Sales department, we always enjoy meeting you in person. We greatly value your business and strive to improve our service to you.

In the future, we plan to continue our education series with more TECHVIEWS, Webinars, and training sessions. Please check our [Web site](#) often for events near you!

And don't miss [TechCon2007](#) on November 4-6. We look forward to seeing you here in Albuquerque!

— by Gale Robledo



Marex Returns Home to BASIS - A Personal Journey

By Kurt Williams

Rewind to 1979. I wrote my very first line of Business BASIC code on an MAI BasicFour 410. It was exciting to control a video screen even if it was just controlling where characters went on the screen. Before Business BASIC, I wrote programs for batch mode computers that took input from decks of punch cards and sent output to printers, more punch cards, magnetic tape, and, if I was lucky, a magnetic disk! Using a video screen to communicate with a user was a huge advancement.

Editor's note: If these first legs of Kurt's personal computer and programming journey ring familiar to you with punch cards, magnetic tape, and MAI BasicFour, you will especially enjoy the [STARTUP exhibit](#) featured at TechCon2007. [STARTUP: Albuquerque and the Personal Computer Revolution](#) (www.startupgallery.org) is a gift from Microsoft co-founder Paul Allen to Albuquerque. It is the first-ever exhibit dedicated to the history of the PC and software, much of which unfolded in Albuquerque.

Fast-forward twenty years. After much hard work developing and maintaining Business BASIC software and holding diverse positions including some with BASIS International Ltd. (BBx Product Manager and member of the Engineering Team), I left the Business BASIC world entirely. I accepted a position running an IT department for an electrical products distributor that managed a geographically-large distributed wide area network with several hundred users. There, I used several technologies including Java and SQL.

As my Java development skills and SQL knowledge grew, new object-oriented design, integrated development environments, and relational database management tools made programming in Business BASIC seem like ancient history. Capitalizing on this new knowledge, I later started a software contracting business to focus on developing business applications using Java, MySQL, and other technologies. Marex Services was born. Now several years later, Marex Services has built a healthy client base of satisfied customers.

A few months ago, BASIS asked Marex Services to develop some code for them using their most advanced technologies. I was skeptical. I was a Java developer now but they persisted. They insisted that the new BASIS technologies competed quite well against other development tools and then challenged me to take on the task and prove them right. I accepted the challenge.

Re-entering the BASIS World

Upon initial examination, nothing had changed in the BASIS world. All the old language features, database functionality, and development tools were still in use. But with further examination, I discovered that many more tools were available; a new Integrated Development Environment (IDE) built on the already familiar Net Beans open source project, a plethora of new DBMS file types and most significantly, an object-oriented Business BASIC backed by a powerful BBjAPI (Application Programming Interface) as well as a new SQL engine.



Discovering the BASIS IDE

Soon, I experienced just how powerfully the BASIS IDE brought all the disparate development tools together. I edited, compiled, and ran my programs all in the IDE's debugger. When my compilation errors occurred, they displayed in the compiler output window and a quick double-click on any of these errors opened the source file and highlighted the offending line. When my changes were complete and I was ready to save them to the system, I used the IDE's CVS client to commit the changes to my CVS repository.

The IDE contains several plug-in modules that were very handy. The data dictionary plug-in allowed me to access the data dictionary either locally or remotely and get answers to questions quickly. Similarly, I could easily add or edit new data dictionary definitions. The FormBuilder IDE plug-in provided me with a visual layout tool for graphical user interfaces. I could quickly build and maintain the GUI forms that I used in my business application. The AppBuilder module, integrated with the FormBuilder module, provided an easy-to-use framework for coding event handlers for my newly created forms. This framework was within the same IDE interface so it managed the event queue to deliver on its RAD promise.

What I discovered was that the IDE is as strong a work bench for my development projects as any IDE that I have ever used. In fact, it is light years beyond any BBx development tool I used years ago.

Discovering the BBjAPI

In the days when I developed in Java and approached a problem, I thought about what resources the Java API provided that I could use to solve this problem. Now working in BBj®, my thought process was similar, but expanded. I first thought about solving the problem using the BBjAPI and most of the time I found the

continued...



Kurt Williams
Principal
Marex Services

solution. However, if I needed a bit more help, I moved outward to the Java API. With this synergy, the two working together always provided a solution. There really were no limits!

One of my greatest discoveries about the BBJAPI was how much time it saved me. For example, if I was using PRO/5® and had a list button on my form that I needed to load from a database table, I would have opened the file, read each record that met my criteria from the file, and placed it into the list button control. Using the record set capabilities in the BBJAPI, I reduced all those steps to the following:

```
connectStr$=" jdbc:basis:ourServer:2001?database=dbName&user=admin&password=myspw"
sqlStr$="SELECT StateAbbr FROM States Order By StateAbbr"
statesRecordSet! = BBJAPI().createSQLRecordSet(connectStr$, mode$, sqlStr$)
statesListButton!.fillFromRecordSet(statesRecordSet!, "StateAbbr")
```

The **connectStr\$** variable was the information the SQL Engine needed to connect to **ourServer**. It further said to use port **2001** and open the database **dbName** as the user named **admin** with the password **myspw**. The **sqlStr\$** entry was the SQL SELECT statement needed to load the **States** list button. The third line created an SQL record set object using **connectStr\$** and **sqlStr\$**. The fourth line filled the BBJListButton object from the BBJRecordSet object created in the previous three.

This BBJAPI approach was straightforward and easy. There was no need to build a read loop loading each individual record read into the list button and no need to handle the error when the loop gets to the end of the file. The **fillFromRecordSet** method called in the BBJListButton object handled all these functions.

Considering all these features, I found the BBJAPI was a rich toolbox of objects and methods for managing the interfaces and processes that made up my business applications.

Discovering the SQL Engine

The BBJ SQL Engine was a first rate SQL processor for managing my applications data whether it resided in the BBJ DBMS files or in a database manager such as MySQL. Much to my surprise, I found that I could do everything with the BBJ SQL Engine that I was accustomed to doing with SQL tools outside of BBJ.

For example, I needed to develop a list of serial numbers registered to a given customer that also had an end user record registered to the serial number. Using a traditional Business BASIC approach, I would have written something like this:

```
read (serialNumber, key=customerNumber$, knum=1, dom=*next) serialNumRec$
readLoop:
  read record(serialNumber, end=doneLoop) serialNumRec$
  if serialNumRec.customerNumber$ <> customerNumber$ then goto doneLoop
  read record(endUserLink, key=serialNumRec.serialNum$, dom=readLoop) endUserLink$
  read record(endUserData, key=endUserLink.endUserNumber$) endUserData$

  REM .. do whatever we need to do ..

  goto readLoop:

doneLoop:
```

While this worked, this approach was time consuming. However, when I moved the processing to the BBJ SQL Engine as follows, it ran much faster:

```
connectStr$="jdbc:basis:ourServer:2001?database=dbName&user=admin&password=myspw"
sql$="SELECT T2.SERIAL_NBR, T3.ACTIVE_FLAG, T1.*, T4.INDUSTRY_NAME "
sql$=sql$+"FROM EndUserData T1 "
sql$=sql$+"INNER JOIN EndUserLink T2 ON T1.END_USER_NBR = T2.END_USER_NBR "
sql$=sql$+"INNER JOIN SerialNumber T3 ON T2.SERIAL_NBR = T3.SERIAL_NBR "
sql$=sql$+"WHERE T1.CUSTOMER_NBR=' " + customerNumber$ + "' "
sql$=sql$+"ORDER BY T2.SERIAL_NBR"
ourRecordSet!=BBJAPI().createSQLRecordSet(connectStr$, mode$, sql$)

while 1
  rowData! = ourRecordSet.getCurrentRecordData()

  REM .. do whatever we need to do ..

  ourRecordSet.next(err=*break)

wend
```

continued...

With this SQL SELECT, the SQL Engine handled all the checking to see that I was only retrieving end user data linked to serial numbers registered to the customer in question. The BBj SQL Engine was certainly equal to any SQL processor that I had ever worked with.

Reflections

At the conclusion of this challenge, I discovered an impressive set of tools to help the programmer develop robust and reliable code, more rapidly than ever before. The BASIS IDE is equal to the tool I once used to develop Java code and in fact, I can also use it to develop Java code. The BBjAPI provides a rich array of objects for creating business applications. The BASIS SQL Engine is a versatile tool for the manipulation and retrieval of data from BASIS DBMS and foreign databases as well.

BBj and all its associated tools is a world-class development suite.

Now, after working with BBj on several more development jobs, I am convinced, and am the living proof that what BASIS told me is true; BBj and all its associated tools is a world-class development suite. Unlike the book title by Thomas Wolfe that so many freely quote, "You Can't Go Home Again," I left Business BASIC years ago for the Java world and have in fact, returned home. My foundational Business BASIC development experience applied directly to all I learned while working with Java and allowed me to get things done far more quickly with the BASIS IDE. The results of this

great journey were highly maintainable and very reliable business applications.

As they say, "home again, home again!" 



See also *Confessions of a Language Polygamist*
www.basis.com/advantage/mag-v10n1/confess.html



BASIS Takes Training to the Next Generation

By Amer Child

Since BASIS began offering free one-hour seminars over the Internet, the demand for such online education increased steadily. Developers attended these sessions by calling an 800 number for the audio presentation and connected to a Web link with their PC for the slide presentation. From the convenience of their own office, home, or customer site, they listened, watched, and interacted with questions and comments through an online chat session and on occasion, a telephone. In many cases, customer needs and concerns determined the subject. BASIS covered such topics as "Step by Step: How to get Started With the BASIS IDE" and "Fresh Productivity Gains With Visual PRO/5 and BBj Using the BASIS IDE." BASIS Partners gained great new insights at such exclusive Webinars as "X"cellerate Your Revenue Stream by Delivering Value to Your Client."

With an overwhelming positive response, the opportunity for and benefit from offering full product training in this paradigm became obvious. Developers would save the tremendous traveling expenses of hotel and transportation, not to mention travel time. They would learn from the comforts of home or office or both. BASIS would dedicate technical support to assist any student with individual needs. In fact, hosting training in Albuquerque allows BASIS engineers to join any question and answer session where they can share their expertise. This pool of resources would greatly benefit the attendees.

Even before BASIS had a chance to offer an online course, two different companies from two different continents requested training for their developers. Not only was the expense to send their developers for classroom training in the US prohibitive, but with offices in different cities and in one case, different continents, they would still incur significant travel expenses to get their developers in one location for a BASIS-led onsite training. Clearly, the solution was online training.

Well, the results exceeded both BASIS' and the customers' most optimistic expectations. With that foundational experience, BASIS will move forward to offer future BASIS Product training online. This new training paradigm allows for additional customer-centric training options while saving customers travel time and money. It also provides a convenient, personal and comfortable environment conducive to individual learning needs. Down the road, pre-recorded courses could be downloadable for review at a more convenient time for the student. More than ever, customer feedback will be a key determinant for BASIS to offer training classes with focus on the topics that will move BASIS developers' business and software to the next level. BASIS remains committed to delivering better training, in more efficient and cost-effective ways. 



Amer Child
 Training & Sales
 Support Specialist

Jars, Jars, and More Jars

By Bruce Gardner

Along with the many new features and enhancements of BBj® 7.0 come a new set of jar files (Java ARchives) and a new method to streamline their usage. Several of these new jars correspond to exciting new controls such as bar, line, and pie charts. BBj 7.0 optimizes the search for classes and simplifies the class path by taking advantage of jar indexing.

Q What is jar indexing?

A Jar indexing optimizes the search process of class loaders in applets and Web start applications, reducing the number of jars downloaded to the client machine and greatly simplifying the class path specification. With applets and Web start, it is no longer necessary to list all of the jars needed for your application. On the server side, jar indexing makes it easier to run BBj on operating systems, such as SCO, where limits are placed on the length of the class path.

In BBj 7.0, the main jar **BBjIndex.jar** indexes all of the jar files in the **<bbjhome>lib** directory. As a result, the default class path contains only one jar file: **BBjIndex.jar**.

Q What jars are available?

A In addition to the main jar, **BBjIndex.jar**, 7.0 adds the new jars shown in the table on the next page and combines and/or renames existing jars to reflect their function more correctly.

Q Will the jar changes affect my Web start implementation?

A Yes. In previous revisions of BBj, the **ThirdParty.jar** was a key jar element listed in the JNLP file. BBj 7.0 deprecated the **ThirdParty.jar** so we recommend changing the entry in the JNLP to **BBjUtil.jar**.

Note: In order to minimize the initial impact of this change, the **ThirdParty.jar** is now a copy of **BBjUtil.jar**, which allows most Web start applications to continue to work without a change in BBj 7.0. However, note this change since BASIS will not ship **ThirdParty.jar** in future revisions of BBj.

Applications that take advantage of new BBj 7.0 controls will also need to list the associated jar files. For example, a Web start application that uses the BBjBarChart would require a **Charts.jar** element in the application's JNLP file. Similarly, to use the new spell checker feature available in all BBj 7.0 text-based controls (see Interface *TextControl* in online documentation at www.basis.com), a developer would include **SpellChecker.jar** and **SpellCheckerDictionary-xx.jar** as elements in the JNLP file. Where **xx** corresponds to the desired language.

Q Can I take advantage of the new jar indexing feature with my Web start implementation?

A Yes. Web start applications can list the **BBjIndex.jar** as an element to load classes from such unlisted jars as those mentioned on the next page. However, two conditions must be met:

1. The Web start JNLP file must contain a **BBjThinClient.jar** element and a **BBjUtil.jar** element. The **BBjThinClient.jar** must appear as the first element in the application JNLP.
2. Both the server and client systems must have JRE 1.6 or greater installed to take advantage of jar indexing.

continued...



Bruce Gardner
Quality Assurance
Engineer



NEW JAR SET		OLD JAR SET	
BBjIndex.jar Charts.jar FontChooser.jar HelpAll.jar JDBCPooling.jar SpellChecker.jar SpellCheckerDictionary-xx.jar (10 jars*)			
ConfiguratorHelp.jar		config.jar	
GuiBuilderHelp.jar		guibuild.jar	
BBjEM.jar		b3odbc.jar BBjEM.jar BBjXML.jar	
BBj.jar		BBj.jar CustomIDE.jar	
BasisInstall.jar		BasisInstall.jar	
BBjFileSystem.jar		BBjFileSystem.jar	
BBjJDBC.jar		BBjJDBC.jar	
BBjPlugin.jar		BBjPlugin.jar	
BBjPortID.jar		BBjPortID.jar	
BBjServer.jar		BBjServer.jar	
BBjSQL.jar		BBjSQL.jar	
BBjThinClient.jar		BBjBridge.jar BBjThinClient.jar	
BBjUtil.jar		BBjUtil.jar	
ThirdParty.jar (Duplicate of BBjUtil.jar**)		ThirdParty.jar	
bcel.jar		bcel.jar	
CustomLookAndFeel.jar		skinlf.jar	
PDF.jar		itext.jar	
RXTXcomm.jar		RXTXcomm.jar	
* Each of these jars provides a spellCheck dictionary for a different language or dialect: SpellCheckerDictionary-am.jar, SpellCheckerDictionary-br.jar, SpellCheckerDictionary-ca.jar, SpellCheckerDictionary-du.jar, SpellCheckerDictionary-en.jar, SpellCheckerDictionary-fr.jar, SpellCheckerDictionary-ge.jar, SpellCheckerDictionary-it.jar, SpellCheckerDictionary-sp.jar, SpellCheckerDictionary-sw.jar ** The ThirdParty.jar is deprecated with BBj 7.0.			

continued...



Q Are there any drawbacks to using jar indexing with Web start?

A Generally, no, however one possible drawback to jar indexing in a Web start environment is the lack of caching of the indexed jars on the client. If caching is needed for certain jar files, use the 'lazy' attribute. When used in conjunction with the **BBjIndex.jar**, the 'lazy' attribute will download the jar on demand, and keep the jar in the client cache. See the sample JNLP file below.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+"
  codebase="http://www.test.com/jnlp"
  href="indexed.jnlp">
  <information>
    <title>Sample Index JNLP</title>
    <vendor>BASIS International Ltd.</vendor>
    <icon href="myicon.gif"/>
    <description>Sample JNLP With Index and Lazy Elements</description>
    <offline-allowed/>
  </information>
  <security>
    <all-permissions/>
  </security>
  <resources>
    <j2se version="1.6+" initial-heap-size="48m" max-heap-size="48m"/>
    <jar href="BBjThinClient.jar"/>
    <jar href="BBjIndex.jar"/>
    <jar href="BBjUtil.jar"/>
    <jar href="Charts.jar" download="lazy"/>
    <jar href="SpellChecker.jar" download="lazy"/>
    <jar href="SpellCheckerDictionary-en.jar" download="lazy"/>
    ...
  </resources>
</jnlp>
```

Q Will the JAR changes also affect applets?

A Yes. In previous revisions of BBj, the **ThirdParty.jar** was a key jar element loaded in the applet HTML file or Java Script. BBj 7.0 deprecated the **ThirdParty.jar**, so we recommend changing the **ThirdParty.jar** element to list **BBjUtil.jar**.

Applications that take advantage of new BBj 7.0 controls will also need to list the associated jar files or list the **BBjIndex.jar** in the HTML or Java script file. Here is a sample Java script with **BBjIndex.jar**:

```
string = '';
string += '<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93" codebase="http://
java.sun.com/products/plugin/autodl/jinstall-1_4_0-win.cab#Version=1,4,0,mn" height="400"
width="500">\n';
string += '\n';
string += '  <param name="type" value="application/x-java-applet;version=1.6">\n';
string += '  <param name="scriptable" value="false">\n';
string += '  <param name="CODE" value="com.basis.bbj.client.bbjapplet.BBjApplet">\n';
string += '  <param name="cache_archive" value="../../lib/BBjThinClient.jar,../../lib/
BBjUtil.jar,../../lib/BBjIndex.jar">\n';
string += '  <param name="cache_option" value="Plugin">\n';
string += '  <param name="cache_version" value="1.0.0.0,1.0.0.0,1.0.0.0">\n';
...

```

Q How will the JAR changes affect installation?

A The BBj 7.0 installation will check for and remove any pre-existing BBj jars in the **bbjhome/lib** directory of the target installation directory. It is important to note that BBj 7.0 *must be uninstalled* prior to installing an older version of BBj to the same directory. Jar and configuration files new to BBj 7.0 will cause unpredictable behavior in older versions of BBj.



The BASIS IDE

Essential in the process of



☕ Designing

☕ Developing

☕ Debugging

☕ Deploying

☕ Delivering



Appearing on developers' desktops all over the world



U.S. 1.800.423.1394 • International +1.505.338.4188 • www.basis.com



**SCO® UNIX® IS THE MOST
RELIABLE OPERATING SYSTEM**



OpenServer™ Renowned for its phenomenal stability, maintainability, quality and reliability. OpenServer 6 sports a modern windows-looking interface.



UnixWare® A mature and proven operating system to support your most critical line of business applications, yet is affordably priced to host all computing needs.



HipCheck™ Can your mobile phone do this? Manage your Windows and UNIX servers from your mobile phone. With HipCheck you can:

- Reboot servers
- Kill individual print jobs
- Reset user passwords
- Check error logs
- Run system commands
- Receive alerts

Plus much more from your mobile phone. To learn more, visit www.scomobile.com/hipcheck



**SCO GROWS
YOUR BUSINESS**

www.sco.com

©2007 The SCO Group. All Rights Reserved. SCO, associated logos, and taglines are trademarks of The SCO Group. All other brand and product names are trademarks or registered marks of their respective companies. UNIX is a registered trademark of The Open Group in the United States and other countries.



BASIS International Ltd.
5901 Jefferson Street NE
Albuquerque, NM 87109-3432

ADDRESS SERVICE REQUESTED

Presorted Std
US POSTAGE
PAID
Albuquerque, NM
Permit No 42