BASIS International Advantage

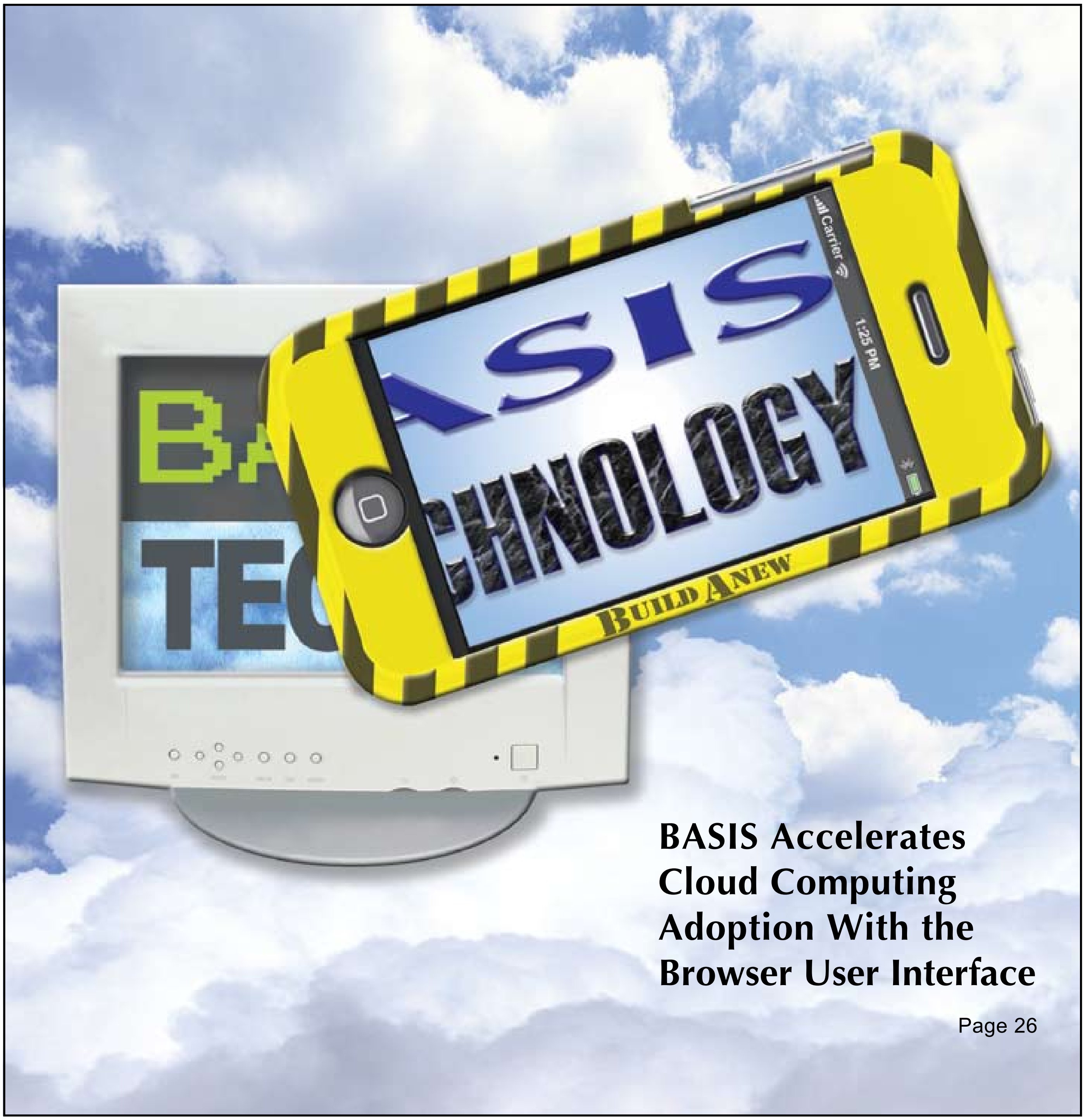
Volume 13 • Autumn 2009

Barista for Your RDBMS of Choice

Page 14

Jazz Up Your Applications – Seamlessly Embed JasperReports

Page 16



BASIS Accelerates Cloud Computing Adoption With the Browser User Interface

Page 26

Document Management Solutions with UnForm®

Production › Delivery › Archiving › Scanning

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

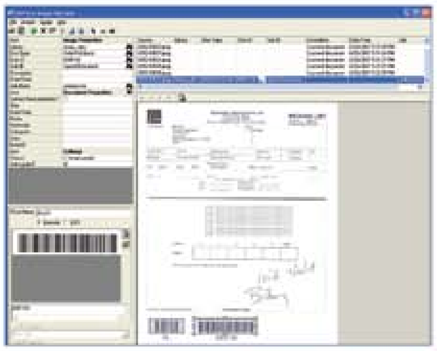
The UnForm Document Archiving and Management component provides the ability to capture, store and retrieve paper-based and electronic documents.

- 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

Universal web browser document retrieval

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.

ASK *Data Reflector*™ Software
Overview & Features

EXECUTIVE SUMMARY

Businesses simply cannot afford data loss. Instantaneous and reliable access to data is one of the key competitive advantages for corporations operating in today's economy. ***Data Reflector***™ provides a simple way to replicate data residing on your UNIX, AIX & Linux servers. ***Data Reflector***™ software integrates with the most popular operating systems for local and/or remote data replication. Automated scripts, wizards, and replication reduce complexity and provide greater flexibility of executing tasks. ***Data Reflector***™ results in a unique, cost effective disaster recovery solution, with less effort to configure offering value to your IT infrastructure.

CONTACT INFORMATION

+1 (610) 617-0300 Office
+1 (610) 617-0307 Fax
sales@asktech.com
www.asktech.com

Data Reflector™ **provides an excellent level of Disaster Recovery at a very affordable price.**

With leading edge replication technologies for most UNIX, AIX & Linux flavors, ***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 technologies to replicate data between the Production and Replication servers.

Data Reflector™ accelerates recovery, cuts cost and does not disrupt the performance of the applications it protects. It's the perfect recovery solutions for software corruption, security breaches, virus attacks, accidentally deleted files and other causes of data loss.

- By the minute, hour, 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 downtime
- Removes deleted files on destination server
- Copies changed blocks only
- Preserves all user password and printer databases
- Recover large amounts of data in only minutes
- Reverse data corruption in a fraction of the time/labor required for recovery from tape
- Eliminate the exposure to major data loss of conventional backup and restore technologies
- Obtain reliable disaster recovery by replicating your data to a remote location

ASK Technologies, Inc. is a national provider of IT hardware, software, and professional services with proven experience and expertise in various technology areas, stretching across multiple vertical markets. With a specialization in multi-service network integration, ***ASK*** creates innovative and cost effective solutions to solve our clients' mission critical business needs.

6

12

22

30

e-article published exclusively at www.basis.com

BUILD ANEW

"Build Anew"...the theme of our recent and most successful TechCon2009 and AddCon2009 conferences in Albuquerque...is a rallying cry to our developer community to refresh, renew, or regenerate their software applications. As we collectively review the recent turmoil of the world and domestic economies and look forward to signs of a recovery, it is wise to take stock of the opportunities before us and the tools that are at our disposal to affect our own recovery. It is therefore most appropriate that this issue of the *BASIS International Advantage* brings with it information about a swathe of instantly useful utilitarian tools and product enhancements to help you build anew.

In our last issue, we highlighted the new building blocks available from BASIS, a horizontal set of accounting modules, and a fantastic application development framework – Barista®. In this issue, we augment those building blocks with new BBx® and Java-based utilities that you can use in any development project, be it built with the aid of Barista, AppBuilder, or hand-coded BBx code in the IDE. The utilities are highlighted with a new temporary 'construction themed' tab Utility and include a language translator utility, a report utility, a live-data-copy utility, an automated update utility, and fax and e-mail utilities.

Improved Java and BBj® interoperability listed on page 28 will further facilitate the use of publicly available Java libraries that can be used as additional application building blocks. Several articles also deal with enhancements and improvements to Barista, AppBuilder, and source code management in the IDE. AppBuilder generated programs benefit from the use of the PROCESS_EVENTS paradigm and Barista sports the ability to build applications on top of a RDBMS such as the BASIS ESQL table-type or a wide range of third party RDBMSs. System administration is made easier with a built-in Jetty Web Server and, coming early in 2010, LDAP authentification support.

All of these goodies might be less important to you than the three most significant highlights of this issue – the seamless integration of the open-source WYSIWYG iReport functionality that boasts both SQL and legacy READRECORD data access capabilities; the use of the integrated Jetty Web server to offer BBx application logic as a Web service; and the announcement of the Browser User Interface (BUI). BUI is set to revolutionize the development of Web applications and the deployment of existing GUI applications in most any Web browser, be it on a desktop or on a cell phone!

Happy reading, and, as always, we welcome your feedback about any and all of our endeavors to help you *Build Anew*! ■

Nico Spence
Chairman & CEO

Dr. Kevin W. King
President & CIO

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 Services
Cottonwood Printing Company, Inc.

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/legal/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

© 2009 by BASIS International Ltd.
All rights reserved.

Recipes for Successful Report Writing

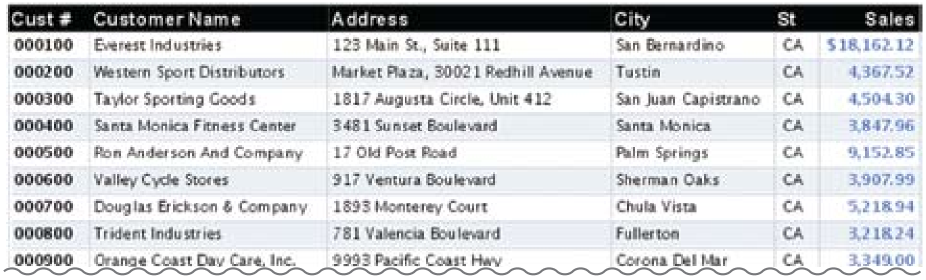
iReport is an easy-to-use open source report writer that allows BBj® developers to create custom reports in a quick and efficient manner. Built using the Netbeans framework, the iReport interface is one with which most of you will be familiar and can quickly and easily create custom reports. You can also achieve a more sophisticated look or capability that is not obvious with very little extra effort. This article shares recipes for several eye-pleasing enhancements that you can apply to your reports. Read more about the BBj-specific iReport integration points supplied by BASIS in *Jazz up Your Applications* on page 16 of this issue.

Alternating Row Colors

To easily achieve alternating row colors using the rectangle report element, create a rectangle around a row in the iReport Designer and size it so that it covers the entire row. Next, change the rectangle to the desired alternating row color and check the “Opaque” box in the rectangle properties. At this point every row will use the colored background. To tell iReport that it should only use the background for alternating rows, modify the “Print when Expression” property, a Java expression that must resolve to either True or False to determine whether to include the component in the generated report. To show the background on the even rows, use the variable `$V{PAGE_COUNT}` that provides the current number of records that have been processed in the current page. In other words, it gives the row number, in real time, as the report is created. Next, create the following expression to determine if the current row is even, resulting in a Java Boolean that iReport uses to determine whether or not to draw the background color:

```
new Boolean( $V{PAGE_COUNT}.intValue() % 2 == 0)
```

The conditional expression causes even rows to display the colored rectangle as shown in **Figure 1**.



Cust #	Customer Name	Address	City	St	Sales
000100	Everest Industries	123 Main St., Suite 111	San Bernardino	CA	$18,162.12
000200	Western Sport Distributors	Market Plaza, 30021 Redhill Avenue	Tustin	CA	4,367.52
000300	Taylor Sporting Goods	1817 Augusta Circle, Unit 412	San Juan Capistrano	CA	4,504.30
000400	Santa Monica Fitness Center	3481 Sunset Boulevard	Santa Monica	CA	3,847.96
000500	Ron Anderson And Company	17 Old Post Road	Palm Springs	CA	9,152.85
000600	Valley Cycle Stores	917 Ventura Boulevard	Sherman Oaks	CA	3,907.99
000700	Douglas Erickson & Company	1893 Monterey Court	Chula Vista	CA	5,218.94
000800	Trident Industries	781 Valencia Boulevard	Fullerton	CA	3,218.24
000900	Orange Coast Day Care, Inc.	9993 Pacific Coast Hwy	Corona Del Mar	CA	3,349.00

Figure 1. Example of even rows displaying the colored background

Hyperlinks

Today, the need to include hyperlink components in reports is on the rise. These hyperlinks can be one of three distinct types in iReport and BBJasper, each with a different purpose – the Reference type, Command type, and ReportName type. All of these types are specified in the hyperlink properties of a report element. To access the Hyperlink Properties dialog box, right click the report element inside the iReport designer and choose “Hyperlink” from the popup menu.

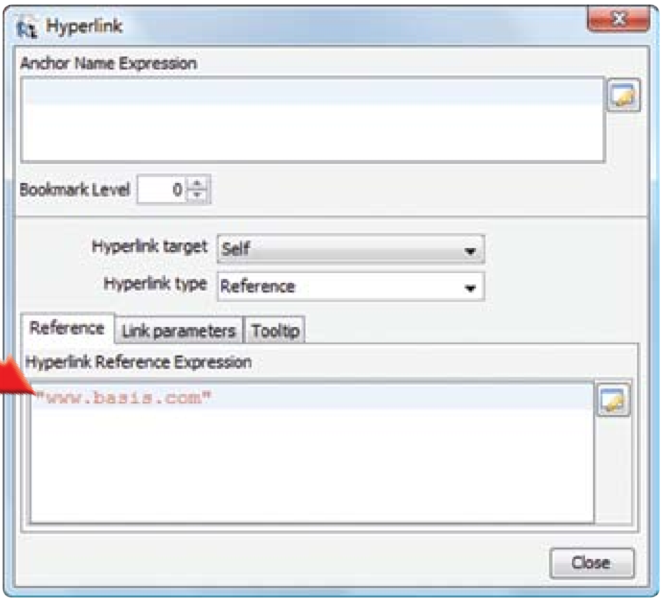
Reference Hyperlink

The Reference type is the kind of link that most users associate with the word hyperlink. When the report displays via the BBJasper utility, clicking the link causes the BBInvoker utility to open the system’s default Web browser and display the target Web page that is determined by the URL specified in the hyperlink definition.

To create a Reference hyperlink, enter the URL in the Hyperlink Reference Expression box of the Reference tab. The value is a string and must be surrounded by quotes as shown in **Figure 2.**

>>

By Robert Del Prete
Quality Assurance Engineer


Hyperlink

Anchor Name Expression

Bookmark Level 0

Hyperlink target: Self

Hyperlink type: Reference

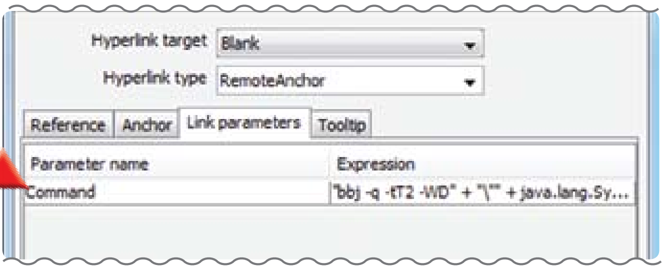
Reference | Link parameters | Tooltip

Hyperlink Reference Expression

"www.basis.com"

Close

Figure 2. Reference hyperlink example

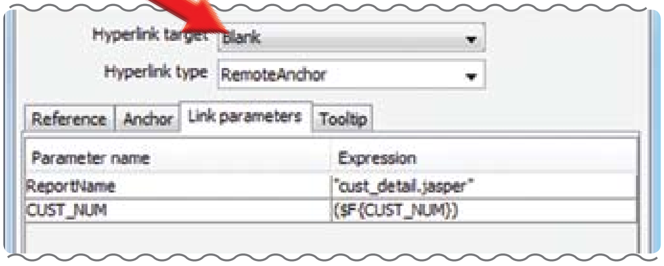

Hyperlink target: Blank

Hyperlink type: RemoteAnchor

Reference | Anchor | Link parameters | Tooltip

Parameter name	Expression
Command	"bbj -q -tT2 -WD" + "\"" + java.lang.Sy...

Figure 3. Command hyperlink example


Hyperlink target: Blank

Hyperlink type: RemoteAnchor

Reference | Anchor | Link parameters | Tooltip

Parameter name	Expression
ReportName	"cust_detail.jasper"
CUST_NUM	($F{CUST_NUM})

Figure 4. ReportName hyperlink example

Command Hyperlink
The Command hyperlink type takes advantage of BBj's SCALL capability to run another program or BBj application. The Command parameter passes the expression given to the command shell. The expression is a string and also requires quotation marks. **Figure 3** shows the parameter name Command and the following expression that launches the Digital Dashboard demonstration program to detail sales figures for the desired month:

```
"bbj -q -tT2 -WD" + "\"" + java.lang.System.getProperty("basis.BBjHome") +
"/demos/digitaldashboard" + "\"" + " digitaldashboard.src - " +
$P{ORDER_MONTH} + "&"
```

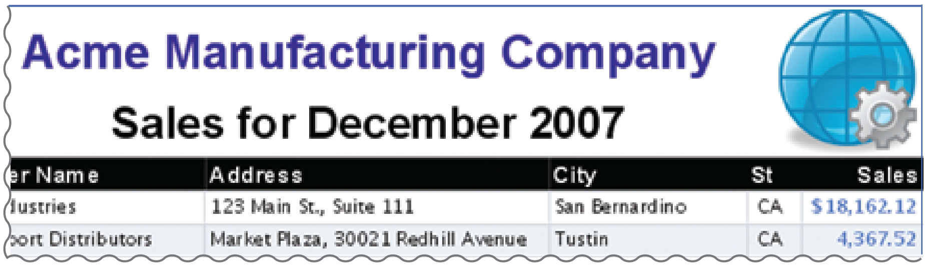
ReportName Hyperlink
The ReportName hyperlink instructs BBJasper to launch another Jasper report. Use the ReportName parameter to specify the path and name of the new report, as well as any additional parameters the report requires. The report author may also control the resultant window's sizing, placement, and title via additional parameters. If the Hyperlink target is set to Self, the new report will replace the current report in the viewer window. If the Hyperlink target is set to Blank (see **Figure 4**), the report appears in an additional viewer window.

Expressions

Expressions can be used in many situations when designing a report. For example, it is possible to display a readable month name when the input is a numeric value. An additional field in the database can take care of this but why bother when iReport has the capability of utilizing programmable expressions? To change the parameter $P{MONTH} into a readable month value, use the expression below. >>

```
"Sales for " + ( $P{MONTH}.equals("1") ? "January" :
( $P{MONTH}.equals("2") ? "February" :
( $P{MONTH}.equals("3") ? "March" :
( $P{MONTH}.equals("4") ? "April" :
( $P{MONTH}.equals("5") ? "May" :
( $P{MONTH}.equals("6") ? "June" :
( $P{MONTH}.equals("7") ? "July" :
( $P{MONTH}.equals("8") ? "August" :
( $P{MONTH}.equals("9") ? "September" :
( $P{MONTH}.equals("10") ? "October" :
( $P{MONTH}.equals("11") ? "November" :
( $P{MONTH}.equals("12") ? "December" : "" )))))))))))) + " " +$P{YEAR}
```

This expression converts the **"12"** in **${P_MONTH}** to the string December, provides the year from the $P{YEAR} parameter, and adds additional text to create the report's title as shown in **Figure 6.**



Acme Manufacturing Company

Sales for December 2007

er Name	Address	City	St	Sales
lustries	123 Main St., Suite 111	San Bernardino	CA	$18,162.12
oort Distributors	Market Plaza, 30021 Redhill Avenue	Tustin	CA	4,367.52

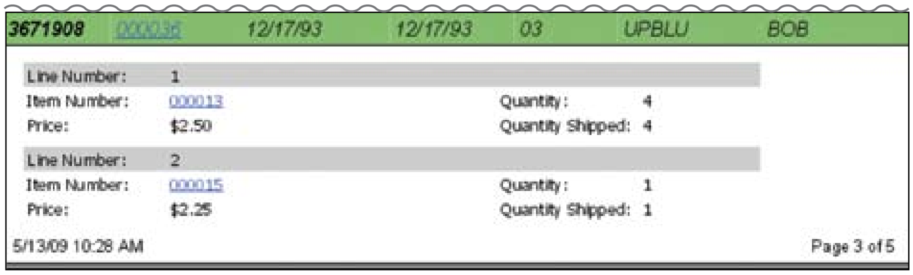
Figure 6. Example of month and year

Evaluation Time

Report Elements in iReport have an "Evaluation Time" text field property. This property is particularly useful for parameters or variables that need evaluation at different times during report creation. Useful values for this field include:

Now	Evaluates the expression immediately
Report	Evaluates the expression at the end of the report
Page	Evaluates the expression at page end
Column	Evaluates the expression at column end

When creating a new report, iReport automatically includes the $V{PAGE_NUMBER} variable by default. With the Evaluation Time set to **Report**, it evaluates the variable once the report is complete and gives the total number of report pages. However, with the Evaluation Time set to **Now**, it displays the page number that the report is currently creating. Therefore, the same variable can be used to create a report footer that shows the current page and total number of pages for the report as shown in **Figure 7,** just by modifying the Evaluation Time property.



```
3671908   000036   12/17/93   12/17/93   03   UPBLU   BOB

Line Number:   1
Item Number:   000013                     Quantity:          4
Price:         $2.50                      Quantity Shipped:  4
Line Number:   2
Item Number:   000015                     Quantity:          1
Price:         $2.25                      Quantity Shipped:  1
5/13/09 10:28 AM                                        Page 3 of 5
```

Figure 7. Current and total page count example

While the page number example serves as a good illustration for the use of the Evaluation Time property, it is so common that the creators of iReport made this property even easier to access via the Tools section of the component palette. Expanding the Tools section reveals new options that provide a shortcut to some of the more popular but previously obscure features such as adding the page number, total pages, or the 'Page X of Y' as shown in **Figure 8**. Dragging and dropping this option into the report does everything covered previously in one easy step, as iReport creates the two text fields referencing the page variable and sets the evaluation time appropriately for each field.

Calculation Property of Variables

Calculating values within report fields using the Calculation property of a parameter is often times much easier than creating a subreport to do the same thing via SQL. An additional parameter in the report easily accomplishes this. In the properties of the added parameter, change the Variable Class to **`java.math.BigDecimal`** and enter the >>

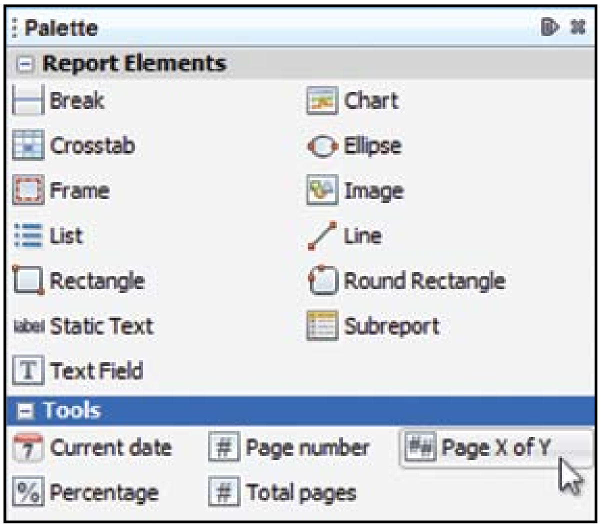
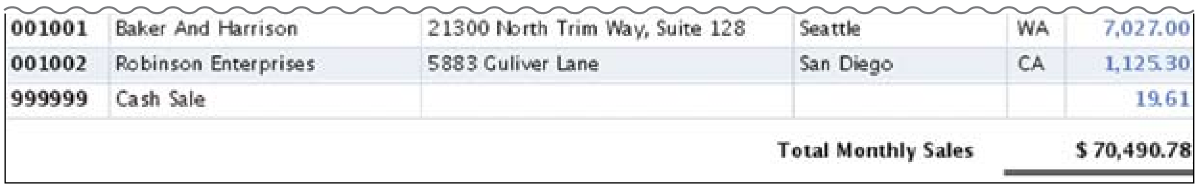


Figure 8. Page X of Y element from the Tools section of the Component Palette

calculation to perform in the Variable Expression property. The Calculation property contains several options including:

Nothing	No calculation is performed
Count	Returns the number of times the result is different from nu
Distinct Count	Returns the number of unique results returned
Sum	Adds each expression value to the current value
Average	Returns the average of all expressions
Lowest	Returns the lowest value received
Highest	Returns the highest value received

For example, to total all of the columns of the Invoice totals in a report, create a new variable called **TOTAL_SALES**. Modify the Variable Class and change the Variable Expression to the Invoice total field $F{INVOICE_TOTAL}. Since we are summing the results, change the calculation type to **Sum**. This variable will now return the total of all Invoices that we received in the report (see **Figure 9**).



001001	Baker And Harrison	21300 North Trim Way, Suite 128	Seattle	WA	7,027.00
001002	Robinson Enterprises	5883 Guliver Lane	San Diego	CA	1,125.30
999999	Cash Sale				19.61
			Total Monthly Sales		$ 70,490.78

Figure 9. Calculation example

Summary

iReport caters to all levels of users to quickly and easily develop new reports for their data. After creating a few reports with the built-in wizards, most report authors quickly discover that it whets their appetite for creating more advanced reports that utilize custom features and a more polished presentation. By implementing the techniques covered in this article, developers will have a head start in creating better looking and more functional reports.

For more information about iReport, check out the many books and online PDFs for sale such as www.amazon.com/Definitive-Guide-iReport-Experts-Voice/dp/1590599284

Database Query Analysis

As computers have progressed in their power and speed, so too have the expectations of software users for instant access to their data. While application tuning and hardware improvements can help, it is often necessary to find ways to retrieve information from the data files or database used by the application in a more efficient manner. Most experienced BBx® developers understand that keys or indexes play a major role in gaining quick access to required data. However, they can have difficulty determining user requirements for indexing, especially when using SQL and third party data access or reporting applications.

Powerful New Analysis Tool

BBj® version 9.0 introduced a powerful new tool to aid the developer or database administrator in discovering the types of queries that users are performing on the database. This new feature, “Query Analysis” (not to be confused with “Table Analysis” that determines the uniqueness of index segments in a table), analyzes the combinations of columns used in the WHERE clause of all SELECT, UPDATE, and DELETE statements. The database engine stores this information in a new data dictionary table called DD_INDEX_USAGE. Each time an application executes a query, the database engine adds information about the columns included in the WHERE clause. When the application closes the connection, it writes the collected information to the DD_INDEX_USAGE table for permanent storage.

How Does This Help?

Using Enterprise Manager (EM), the developer or administrator can see all of this query usage data and make decisions regarding adding useful indexes on tables in the database. **Figure 1** shows an example of the information shown in the Query Analysis tab for a database.

The grid of information on the Query Analysis tab shows a list of query data consisting of a table name, combination of columns used in the query, score, and whether there is currently an index on that combination of columns.

Look at the first row. This shows that a WHERE clause containing both COST_ACCOUNT and PROD_CAT was used seven times by users. Note that it is positioned at the top of the list, which means it is quite common and that it is not indexed. Row 2 shows that COST_ACCOUNT was used seven times as well (every permutation of the columns in a WHERE clause will get its own row in the table) and that it is not indexed either. Analyzing this information, the developer might seriously consider adding an index on the COST_ACCOUNT column because of its common use in queries. While this example is very simple, it shows the power and possibilities for improving query performance for end users.

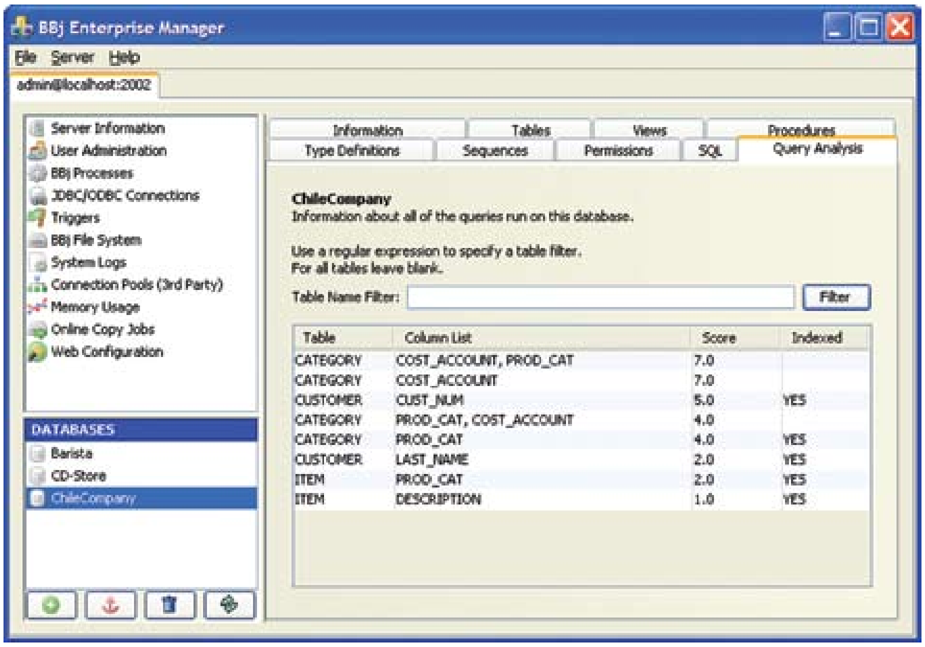
Real World Example

Here at BASIS, we used this new tool to improve our own intranet site and include such information as scheduled meetings and which employees are out on leave, vacation, or travel. This internal Web page generates its content dynamically based upon data stored in a BBj database. As more and more entries filled the “Out Today” and “Meetings” tables, there was a steady decline in performance. Using Query Analysis, we easily determined that the queries that the Web page ran were not using indexes in an optimal manner. Furthermore, we identified the necessary indexes and added them to the database using EM's built-in table definition capabilities, thereby greatly improving performance.

Summary

BBj 9.0 provides a powerful new feature to assist developers and administrators in further optimizing and fine tuning the performance of their databases. Using the Query Analysis tab in Enterprise Manager helps to quickly determine the index requirements and make the necessary changes to the database. Once again, BASIS provides another powerful tool to significantly improve the end user experience without changing a single line of BBj or other application code. ■

By Jeff Ash
Software Engineer



ChileCompany
Information about all of the queries run on this database.

Use a regular expression to specify a table filter.
For all tables leave blank.

Table Name Filter:

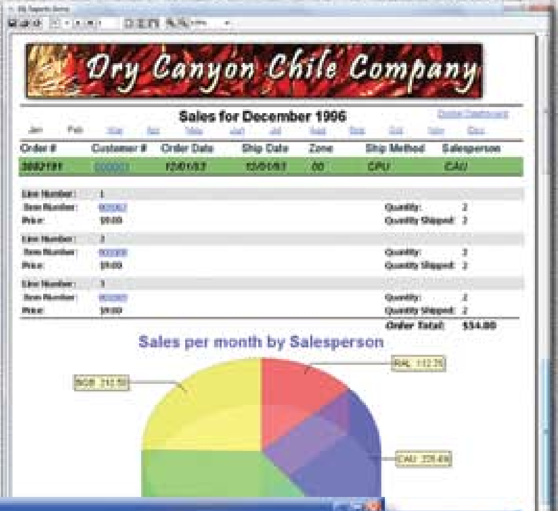
Table	Column List	Score	Indexed
CATEGORY	COST_ACCOUNT, PROD_CAT	7.0	
CATEGORY	COST_ACCOUNT	7.0	
CUSTOMER	CUST_NUM	5.0	YES
CATEGORY	PROD_CAT, COST_ACCOUNT	4.0	
CATEGORY	PROD_CAT	4.0	YES
CUSTOMER	LAST_NAME	2.0	YES
ITEM	PROD_CAT	2.0	YES
ITEM	DESCRIPTION	1.0	YES

Figure 1. The Query Analysis panel in EM

BASIS Utility Toolset

iReport and BBJasper

- Opens access to your normalized or non-normalized data with a BBj report viewer that integrates with the open-source iReport designer generated report



Dry Canyon Chile Company

Sales for December 1996

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Order #	Customer #	Order Date	Ship Date	Zone	Ship Method	Salesperson
3002191	[illegible]	12/01/93	12/01/93	00	CPU	CAU

Line Number: 1
Item Number: [illegible] Quantity: 2
Price: $9.00 Quantity Shipped: 2

Line Number: 2
Item Number: [illegible] Quantity: 2
Price: $9.00 Quantity Shipped: 2

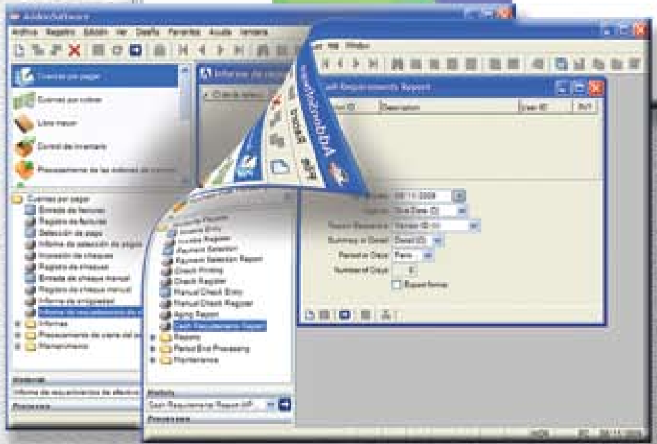
Line Number: 3
Item Number: [illegible] Quantity: 2
Price: $9.00 Quantity Shipped: 2

Order Total: $54.00

Sales per month by Salesperson

BBJabber and BBTranslator

- Provides Java resource bundle functionality for efficient translation assistance in creating multilingual programs



E-mail/Fax Output

- Facilitates the simple integration of your application's data with e-mail or fax output

LaunchDock

- Includes a customizable GUI launch pad for your applications

BASIS Update Service (BUS)

- Provides Web Service distribution of incremental update patches between released versions of your applications to your clients

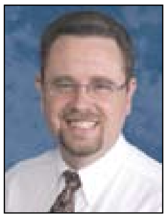
Jump on the BUS – BASIS Update Service!

All Windows users have seen it... that annoying little tool tip in the system tray that says something like "An update is now available." Now, your application can be annoying too! Of course, I am being facetious, but an application feature that automatically looks for updates can be very desirable and often essential.

Along your development journey, jump on the BUS (BASIS Update Service) and travel light so end users can get a desired feature or an essential fix quickly, without waiting for a full release of your product.

Here Comes the BUS

The need for a BUS came about with the Barista® Application Framework, the BASIS RAD tool. Barista is both a development framework and a run-time engine for Barista-developed applications. So, the need is two-fold as both the developer and the end-user communities will benefit from the BUS. Barista developers wanted the ability to update their applications via a mechanism that would not require a lot of work or be too intrusive for the users. Previously, Barista developers had to wait for the next full release to get a bug fix or the developers had to manually create a .zip or .tar file with the new versions of the programs and files for the customers to manually install. Combining input from the Barista developers with an awareness of the flexibility of Web Services development, BASIS engineers agreed the perfect solution would be Web-based.



By Brian Hipple
Quality Assurance Supervisor

Get On Board

Typically, implementing Web Services takes a great deal of time and effort. Coding the BBj® service application is the easy part; the time consuming part is completing the overall configuration as well as writing and debugging the ancillary Java code. BBj 9.0 introduced an embedded Jetty Web Server that easily deploys BBj applications as Web Services without a great deal of configuration effort or the need to write Java code. This feature really does remove the pain and effort, allowing developers to set up a Web Service faster than they could ever imagine.

The BUS is a BBj CustomObject that utilizes the new Web Services functionality. The BASIS DBMS provides the back end of the BUS (that's where all the cool people sat when I was a kid!) using ESQL files and their support for foreign key relationships to provide an extra level of data integrity.

To take advantage of the public interface, the application passes information to the BUS such as vendor, name, revision, and current update level. The BUS then notifies the client application whether an update is available. If an update is pending, the service provides the application with the new update level, date, and a description about what the update includes. If the application wants the update, the BUS provides the FTP location of the update from which the client can download the JAR and install its contents. BASIS' design decision to deliver updates via an FTP server was predicated on two facts: 1) FTP servers are a proven and reliable way to provide files to users 2) FTP servers are accessible to any FTP client. This is especially important if the application has no access to the internet (yes it is true!), users can still download the update on another machine with internet access, copy the update to the target machine via 'sneaker net' or a USB drive, and apply the update.

To accept this automatic update, the developer must configure the application to allow updates, similar to the way Barista has, shown in **Figure 1** and **Figure 2**.

Any client with proper credentials can add or remove an update from the BUS. For convenience, the BASIS Installation Suite installs the BUS Administration > >

module along with the BUS Web Service and database structure. The BUS Admin module requires a user name and password for validation, allowing authenticated clients to add a patch. To add a patch to the BUS service, the client must provide such application information as vendor, name, revision, location. Given this information, the BUS can then create and return the new update level. To remove an update, the developer must specify a vendor, application name, revision, and update level as shown in **Figure 3.**

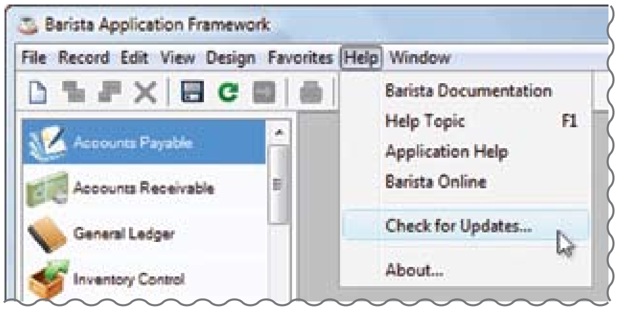
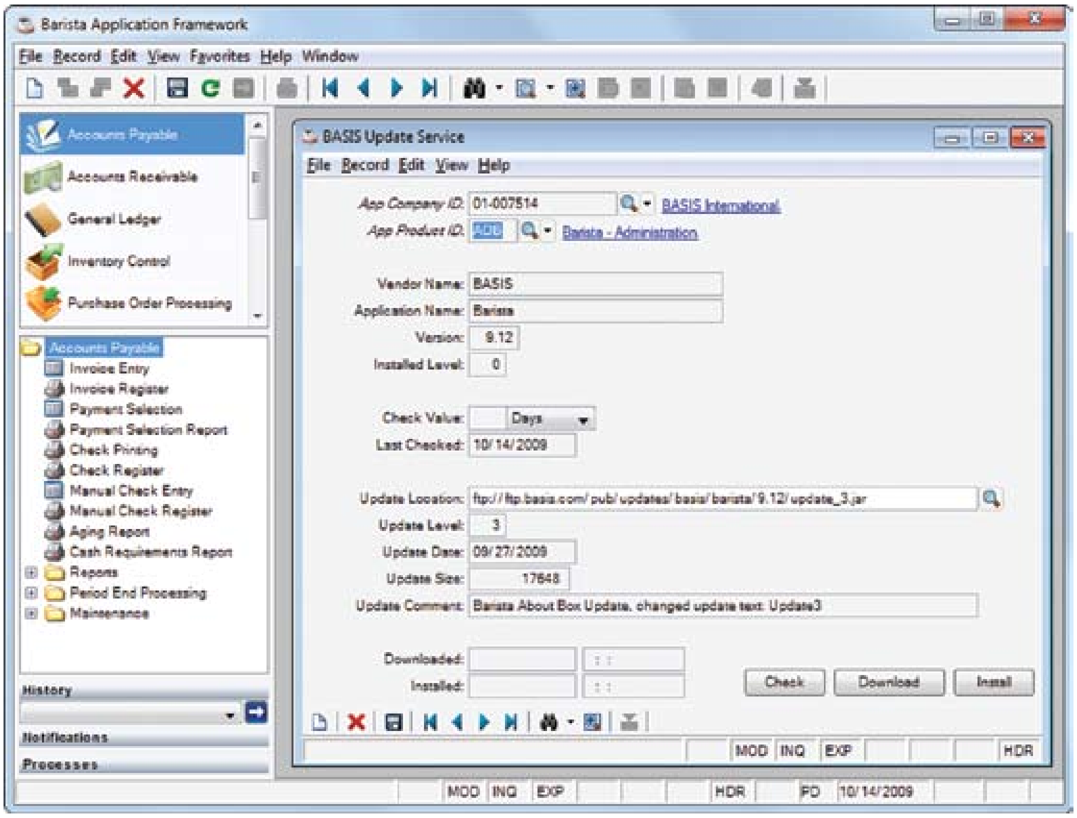


Figure 1. Barista's "Check for Updates" using the BUS



App Company ID: 01-007514 BASIS International
App Product ID: ADB Barista - Administration
Vendor Name: BASIS
Application Name: Barista
Version: 9.12
Installed Level: 0
Check Value: Days
Last Checked: 10/14/2009
Update Location: ftp://ftp.basis.com/pub/updates/basis/barista/9.12/update_3.jar
Update Level: 3
Update Date: 09/27/2009
Update Size: 17648
Update Comment: Barista About Box Update, changed update text: Update3
Downloaded:
Installed:

Figure 2. Barista using the BUS to provide updates

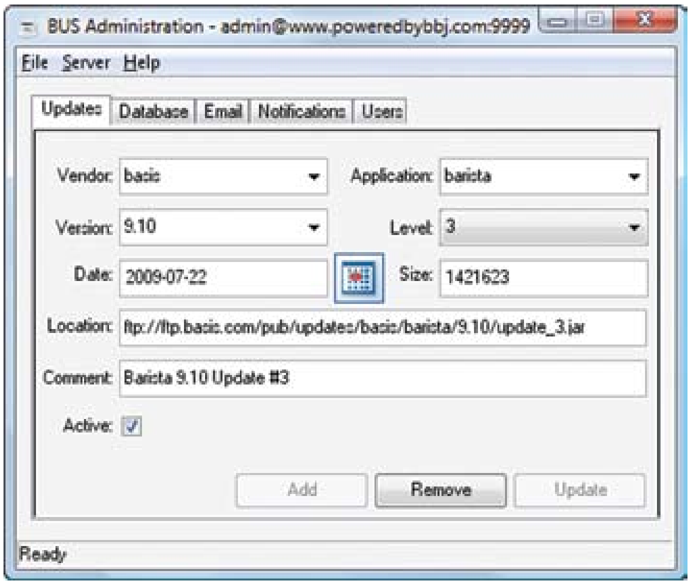
Since the BUS is implemented as a Web Service, the client can be a BBj, Java, Perl, C++ program, or any application that supports Web Services consumption. For example, the Barista development team chose a client Perl program to add updates. This Perl program runs on demand, checking files out from an update branch of the source code repository, and then adds the update via the BUS. A Barista developer only needs to check in the code to the appropriate SVN branch to be included in the next update. How easy is that?

Recent Barista enhancements make it possible for end users to check for updates via the BUS and install any pending updates. Alternatively, Barista can easily install an update downloaded manually from an FTP server.

Summary

Empowering end users with the ability to update their applications with the BUS eliminates the need to manually distribute updates or create a new revision of the product. Whether adding a 'check for updates' prompt at program startup or invoking it manually, easily updating an application reduces maintenance costs and gives end users the confidence that they can get updates in a timely and convenient manner improving the quality and value of your product and service offering. The BUS Utility is a BBj 10.0 feature available today for preview in 9.x.

Isn't it time for you to offer your customers a ride on the BUS? ■

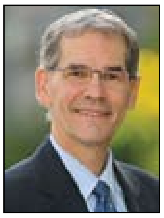

BUS Administration - admin@www.poweredbybbj.com:9999
File Server Help
Updates Database Email Notifications Users
Add Remove Update
Ready

Vendor: basis
Application: barista
Version: 9.10
Level: 3
Date: 2009-07-22
Size: 1421623
Location: ftp://ftp.basis.com/pub/updates/basis/barista/9.10/update_3.jar
Comment: Barista 9.10 Update #3
Active: ✓

Figure 3. BUS Admin module showing details about an available update

Barista Uses Your RDBMS of Choice

Barista® is an application development and runtime framework written in BBj® and, like thousands of applications developed with BASIS products, was initially created to utilize the extremely fast and efficient native BBx® file system. Used to create new GUI applications, refresh old GUI applications, or transform existing CUI applications to GUI, this data-driven framework takes over an enormous amount of the development effort by simply describing the data model in Barista's data dictionary. BASIS realized this metadata concept could be as revolutionary to the widespread market of SQL or relational database applications. To facilitate this, BASIS decided to "open up" Barista to allow any JDBC 2.0-compliant relational database to act as the user data repository! To accomplish this large goal, BASIS broke it down into smaller tasks.



By Ralph Lance
Software Engineer

The Road Ahead

The first challenge was for BASIS to change all data I/O from the native BBx file system to database independent SQL in order to work with any relational database. While this would otherwise be a sizable challenge for any software house, Barista's modular construction reduced it to a manageable project since most of the data I/O was reasonably contained.

The next challenge was retrieving all the required information from the foreign database. Since database architects have created their databases in SQL, it was important to use that metadata to build the necessary Barista table and data element descriptions while accommodating incompatibilities with the architecture of the BASIS data dictionary. For example, SQL databases may allow table and column naming conventions incompatible with Barista, so it was necessary to modify these names without losing track of their original values.

Integrating with third party databases also presented Barista with data layout challenges. BASIS modified Barista to translate the external data to and from the required standard BASIS record templates to resolve the data inconsistency. This solution made the least impact on Barista's existing code base.

One challenge was already satisfied by Barista. It has, since its debut, used SQL for the table inquiry function. The only additional need was to ensure that the SQL commands were standardized.

Lastly, Barista and the external database may both have their own authentication mechanisms, such as user name and password, so the connection information and user credentials needed to be stored in Barista tables.

Interfacing With the Third Party Database

BBj and the BASIS SQL Engine offer a myriad of possibilities and functionality targeted at solving specific SQL-related tasks, while abstracting the underlying complexities. This makes it easy for BBj developers to work with a RDBMS database either by using SQL commands or using objects like the BBjRecordSet. However, Barista needed to interface with databases at a deeper level than these constructs provided. For example, Barista needed to obtain metadata, control

>>

the volume of data being processed, and use result set navigation to avoid constructing individual database-specific SQL statements for every data I/O operation.

A new BBj API method came to the rescue – getJDBCConnection(). It returns a BBjCollapsableJDBCConnection object that represents a connection to the specified database. This object is an implementation of the **`java.sql.Connection`** interface that opens up access to the complete functionality of the JDBC API. This method is the cornerstone of a BBj custom class called BBjdbo (BBj JDBC Database Object) found in the Barista program **`bsq_bbjdbo.bbj`**. The BBjdbo class wraps many of the java.sql classes available and handles all of the interfacing to the SQL database via the JDBC 2.0-compliant driver.

Instantiation of the BBjdbo object occurs when connecting to a database where a native file open would normally have been performed. Connection and user credential information from Barista tables are used to make a connection to one or more databases. To keep the connection overhead under control, the class takes advantage of BBj's connection pooling feature to cache database connections. The class also handles result sets on a per table basis and manages their "channels" in a HashMap also stored in the group namespace.

When importing table description information from an existing database, metadata is interrogated for table, column, and index information. During the import, table and column names are modified to fit into Barista's data dictionary as needed. The original names, along with their catalog and schema, are stored in the Barista table and column records. Barista automatically generates data dictionary keys and their segments from the obtainable index information. The import also does its best to identify primary and foreign key relationships and sets up these relationships as validation tables in Barista's data element definitions. For more information about importing, see the link at the end of this article.

After describing tables and their data elements in Barista, the developer can easily generate the standard data maintenance forms and immediately use them to query and maintain the data in the external database.

Implementation Details

The actual data access either occurs from result set navigation (first(), last(), next(), previous()) or via BBjdbo methods like getRecordByKey(). Keep in mind that an SQL "key" translates to a WHERE condition that may be composed of more than one segment. To accomplish this, Barista constructs a HashMap of <column name>, <operator>, <value> conditions from the field-based key segments defined for a table such as **`cust_num, =, 000012`**. The column name and operator are used to build the WHERE clause for a prepared statement, e.g. **`WHERE cust_num=?`**. Values are passed to a method that sets the query parameters. The use of the Statement.setObject() method eliminates the burden of dealing with different data types.

The SELECT statement built by Barista for inquiries is simply passed to an execute() method that returns the SQL result set as a vector of BBjTemplatedStrings. Barista uses this vector, instead of the READ RECORD loop traditionally used to access the native file system. The maximum number of result set rows returned to the user has a configurable global default value to prevent memory (or user!) overload.

Dealing with Database Differences

Barista queries the database metadata for, among other things, the characters that wrap identifiers in SQL statements to avoid conflicts with reserved words. This is a good example of using what the database itself offers as metadata to avoid hard coding for idiosyncrasies in different databases. Although the JDBC 2.0 API is a "standard," it is up to the driver developers to implement the API interfaces and to determine to what extent they adhere to the standard. Many drivers also have database-specific extensions to this standard. The Barista SQL integration is based on the JDBC standard and in only a couple of places does the database product name play a part in the program logic.

External databases may or may not support SQL transactions and may have different capabilities for constraints, stored procedures, triggers, and the like. As a result, BASIS does not offer explicit support for these in Barista. Unsuccessful SQL operations caused by constraints will notify the user that Barista is unable to complete the process.

Handling large character and binary objects (CLOBs and BLOBs) are usually application-specific and currently handled as non-editable "attachments." Barista displays the first 400 bytes in forms and in the future, will make a call to an attachment viewer.

Summary

BASIS has tested the SQL integration with a number of the more popular relational databases; Oracle, MySQL, SQL Server, JavaDB/Derby, and BASIS' own ESQL relational database. Test for yourselves by checking out some of the sample demo databases and JDBC drivers available for download from the various vendors. Discover first-hand how SQL integration in Barista now gives more users more choices when using BASIS products. ■

For more information about importing, see the tutorials at
www.basis.com/products/devtools/barista/documentation

Read more about Java implementation at
java.sun.com/javase/6/docs/api/java/sql/Connection.html

Jazz up Your Applications – Seamlessly Embed JasperReports

It is easy to see why BBj® developers are jazzed about the iReport Designer – now they can create and preview professional looking reports against their live data with very little effort. The iReport Designer Wizard steps the developer through the report creation process to deliver useful and, in some cases, critical information. The next step, of course, is figuring out how to embed the newly-created report into an existing application.

So, how do you use JasperReports in BBj?

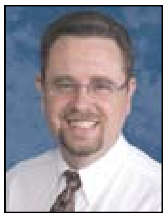
The answer is simple and the focus of this article...use BBJasper – the new utility from BASIS that makes it easy to integrate live reports into existing BBj applications!

BBJasper is a utility distributed in BBj 9.0 and written in BBj as a CustomObject that embeds Java code to wrap and extend the Jasper API. This fully supported, fully documented, and fully modifiable utility delivers a very easy, yet very powerful mechanism to utilize JasperReports in BBj. The BBJasper utility is comprised of three different modules or CustomObjects:

- BBJasperReport
- BBJasperViewerWindow
- BBJasperViewerControl

The BBJasperReport CustomObject allows for creating, displaying, printing, and saving a report from within a GUI, CUI, or background BBj application. To create a report, this API requires a .jrxml or .jasper file and a JDBC connect string. The iReport Designer produces the .jrxml/.jasper file and the JDBC connect string provides the information necessary to acquire the data for the report. Specifying optional report parameters and a locale for report creation provide further report criteria and localization. All of these inputs come together during the fill operation, which takes the design (.jasper or .jrxml) file, obtains the data via a JDBC connection as specified in the connect string, and applies the report parameter and localization. Voilà, a report is born! **Figure 1** is a code sample that displays a BBJasperReport in a BBj window.

BBJasper extends much-needed flexibility in printing the reports. Some common options and preferences include printing to the server or client, displaying an optional dialog to allow the user to change printers and printing preferences, and the ability to save the report in a myriad of formats including HTML, XML, PDF, or XLS. >>



By Brian Hipple
Quality Assurance Supervisor

iReport is a WYSIWYG visual report designer specifically designed for JasperReports. iReport gives administrators and report designers total control over the contents as well as the look and feel of every report. iReport allows users to build, test, and run reports from the desktop environment. iReport is available for Windows, Linux and Mac.

JasperReports is the world's most powerful and widely used embeddable open source Java reporting library for report designers and developers that can be used to generate reports in several formats including, amongst others, PDF, HTML, Microsoft Excel, RTF and XML files. Java applications can use JasperReports to generate dynamic content.

BBJasperViewer is the component that allows JasperReports to be seamlessly embedded and viewed in a GUI BBj Thin Client.

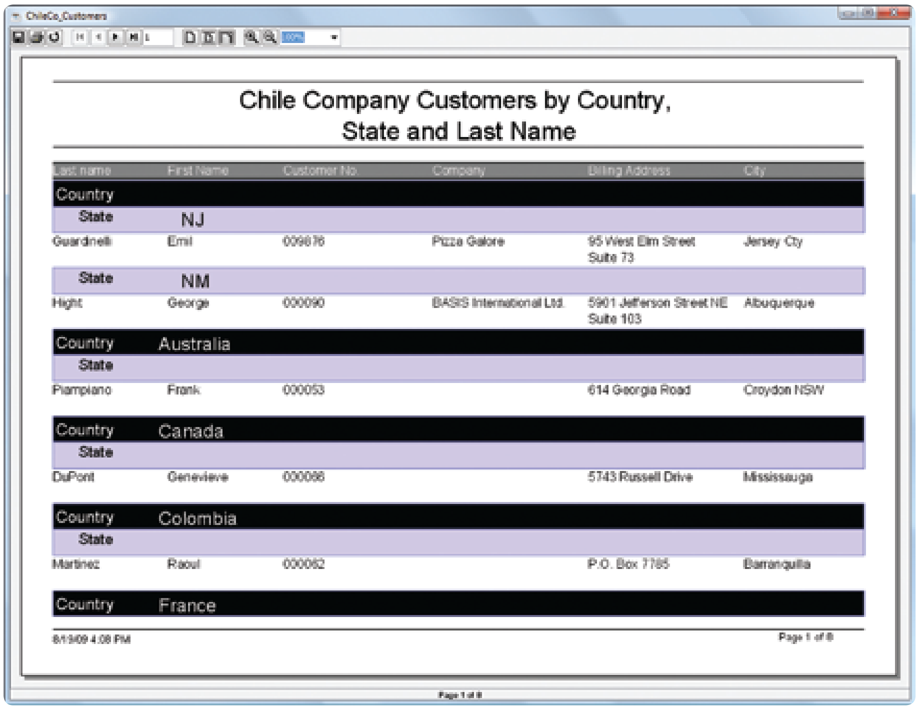
The BBJasperViewerWindow CustomObject provides an API to view the report in a window and is implemented as a BBjTopLevelWindow. As a result, all BBj API functions for a BBjTopLevelWindow are available on this window. Some of the more popular functions include setting the position of the window (x,y), the size of the window (width, height), and modification flags (modal, modeless, close box, resizable, etc). Refer to the online BBj documentation for all available BBjTopLevelWindow online functions.

The BBJasperViewerWindow contains a BBJasperViewerControl sized to the window constraints and is also accessible for further customization as shown in **Figure 2.** The BBJasperViewerControl CustomObject offers the ability to view the report in a control inside a pre-existing BBj window – a top level, child, or MDI. Because it is implemented as a BBjControl, all BBj API functions for a BBjControl are available on this control, including the ID of the control, position of the control (x,y), the size of the control (width, height), etc. (see the online BBj documentation for a full list of available BBjControl functions). The BBJasperViewerControl bestows the finest level of control over the viewed report. By default, it includes toolbuttons that enable users to save, print, reload, navigate pages, set viewing options, zoom in and out, and set the zoom ratio. To go beyond these features, developers can even add custom toolbuttons for extended functionality. The truly adventurous at heart can access the underlying Java control as a client object for absolute control manipulation.

Returning to the opening question, ”How do you use JasperReports in BBj?”, there is a more in-depth answer than simply “use BBJasper.” JasperReports does not actually require the BBJasper utility so developers can also use embedded Java and direct calls to the JasperReports Java API. This approach would take precious resources such as time, money, and effort that few in the IT industry can afford in these tough economic times. So the simple answer is the best answer. With very few resources, using the BBJasper utility unleashes the power of JasperReports. So, turn on your imagination and jazz up your application today!

```
REM USE Statements
USE ::bbjasper.bbj::BBJasperReport
USE ::bbjasper.bbj::BBJasperViewerWindow

REM Declares
declare BBjString reportFile$
declare BBjString connectString$
declare BBJasperReport report!
declare BBJasperViewerWindow reportWindow!

REM Set the report file and connect string
reportFile$ = "chileco_customers.jasper"
connectString$ = "jdbc:basis:localhost?database=ChileCompany"
connectString$ = connectString$ + "&user=admin&password=admin123"

REM Create and fill the report
report! = new BBJasperReport(reportFile$,connectString$)
report!.fill()

REM Create the report window and show the report
reportWindow! = new BBJasperViewerWindow(report!)
reportWindow!.show(BBjAPI().TRUE)
```

Figure 1. Code sample that displays a BBJasperReport



Chile Company Customers by Country, State and Last Name

Last name	First Name	Customer No.	Company	Billing Address	City
Country					
State	NJ				
Guardinelli	Emil	009876	Pizza Galore	95 West Elm Street Suite 73	Jersey City
State	NM				
Hight	George	000090	BASIS International Ltd.	5901 Jefferson Street NE Suite 103	Albuquerque
Country	Australia				
State					
Piampiano	Frank	000053		614 Georgia Road	Croydon NSW
Country	Canada				
State					
DuPont	Genevieve	000066		5743 Russell Drive	Mississauga
Country	Colombia				
State					
Martinez	Raoul	000062		P.O. Box 7785	Barranquilla
Country	France				

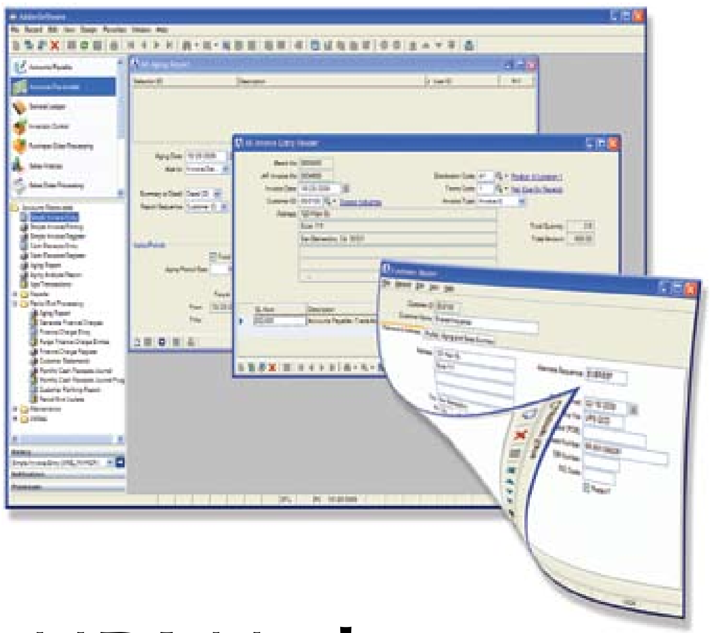
8/19/09 4:08 PM   Page 1 of 8

Figure 2. A BBJasperReport shown in a BBJasperViewerWindow

JasperReports
jasperforge.org/plugins/project/project_home.php?projectname=jasperreports

iReport
jasperforge.org/plugins/project/project_home.php?group_id=83

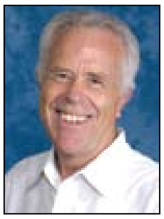
BASIS online documentation for BBJasperReport, BBJasperViewerWindow, BBJasperViewerControl, BBjTopLevelWindow, and BBjControl at www.basis.com/onlinedocs/documentation



MDI Modes – Make Mine a Single, With a Twist

Prior to BBj® 9.0, when a BBj program used an MDI (Multiple Document Interface) window, all programs that were SCALLed by that program (the MDI master process) appeared in the same MDI window and all the SCALLed programs (the MDI child processes) terminated when the master process terminated. BBj 9.0 introduced new API methods and a new command line parameter that allows the program to control whether the child processes will display within the MDI window and whether the child processes will be terminated when the master process is terminated.

Now, the developer controls whether a BBj program that uses an MDI window displays all programs SCALLed by that program in the same MDI window and terminates all the SCALLed programs when the master process finishes. This article introduces this feature with how-to tips on making an MDI application into an SDI (Single Document Interface) anytime it is needed.



By David Wallwork
Senior Software Architect

Default MDI Mode Methods

The new BBj 9.0 method BBjMDI::setDefaultMDIMode(), allows the program to control the behavior of all MDI child processes that are started after calling setDefaultMDIMode().

This method accepts three valid modes.

1. MDI – the traditional mode in which all MDI children appear within the master MDI window and are terminated when the MDI master is terminated.

2. SDI – displays each MDI child using a BBjWindow that is free to be outside the master MDI window but the children will be terminated when the MDI master is terminated.

3. DETACHED – causes the programs that are SCALLed by the MDI master to behave as though they had been launched as separate programs from the command line. They do not appear within the MDI master window, they will not be terminated when the MDI master is terminated, and they will not share the GroupNamespace of the MDI master; they are completely detached from the MDI master.

To call this new method, enter code such as:

```
mdi!=BBjAPI().getMDI()
mdi!.setDefaultMDIMode("SDI")
```

The corresponding method BBjMDI.getDefaultMDIMode() returns the current default MDI mode.

The MDI Command Line Parameter

When starting a new program from the command line, enter the new **-MDI** command line parameter to set the defaultMDIMode of the program. To start a program in "SDI" mode, use a command like the following:

```
BBJ -MDISDI program.src
```

Child Frame Mode Methods

To start a new BBjSession using BBjCommandLineObject, set the MDI mode for the new BBjSession by calling setChildFrameMode() with one of the string values MDI, SDI, or DETACHED. This setting will affect all BBjSessions started with BBjCommandLineObject; the value can be retrieved by calling getChildFrameMode() on the BBjCommandLineObject.

Summary

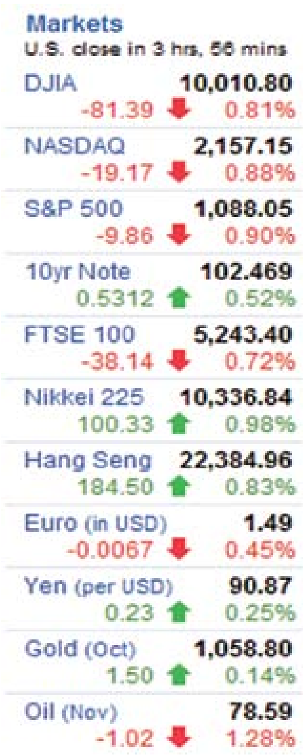
BBj 9.0 provides methods on the BBjAPI object for setting the default MDI mode that will be used when creating new child MDI processes. Similarly, the command line setting and the methods of the BBjCommandLineObject provide control of the mode of a single BBj process. Combining these various methods, the user has complete control of the MDI behavior of all BBjSessions.

Today, you can truly develop your application, any way you want! ■

From Legacy to Enterprise With BBj Web Services

Old is new again!

Have you ever used an application that receives stock quotes or the weather forecast or current news headlines? If so, it is likely you have encountered a Web Service.

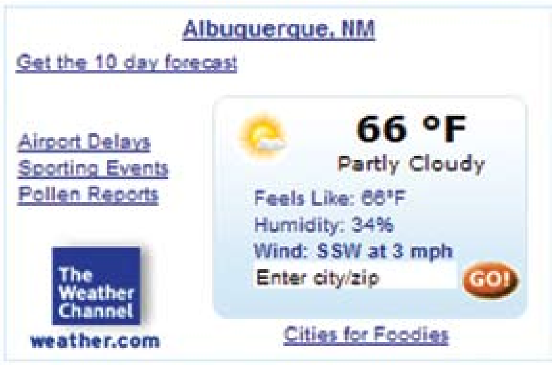


Markets
U.S. close in 3 hrs, 56 mins

DJIA		10,010.80
	-81.39 ⬇	0.81%
NASDAQ		2,157.15
	-19.17 ⬇	0.88%
S&P 500		1,088.05
	-9.86 ⬇	0.90%
10yr Note		102.469
	0.5312 ⬆	0.52%
FTSE 100		5,243.40
	-38.14 ⬇	0.72%
Nikkei 225		10,336.84
	100.33 ⬆	0.98%
Hang Seng		22,384.96
	184.50 ⬆	0.83%
Euro (in USD)		1.49
	-0.0067 ⬇	0.45%
Yen (per USD)		90.87
	0.23 ⬆	0.25%
Gold (Oct)		1,058.80
	1.50 ⬆	0.14%
Oil (Nov)		78.59
	-1.02 ⬇	1.28%

A Web Service is a server-side program that offers up an API in a document called a WSDL (Web Service Definition Language, pronounced: *whiz'-dull*) that a variety of client programs can use. Web Services allow a company to share essential information with a client application via HTTP without the need for a VPN or any other kind of constant connection to the company offering the Web Service. It is possible to create a sales server program that runs on a company's Web server and connect your sales force with robust clients on their laptops.

One example is Federal Express's Web Service that offers up an API for



Albuquerque, NM
Get the 10 day forecast
Airport Delays
Sporting Events
Pollen Reports
66 °F
Partly Cloudy
Feels Like: 66°F
Humidity: 34%
Wind: SSW at 3 mph
Enter city/zip GO!
The Weather Channel
weather.com
Cities for Foodies

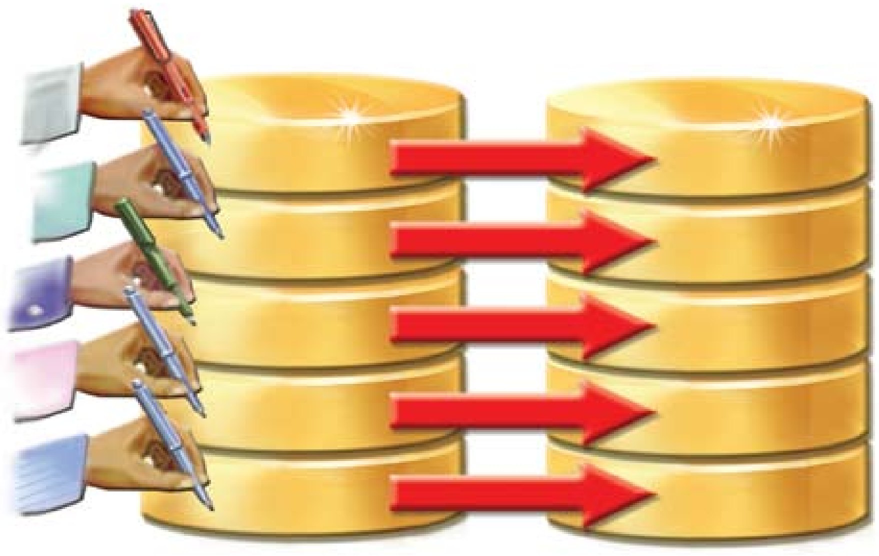
various companies to integrate into their software to track shipments. To see this in action, run the "CUI Package Tracking" and "GUI Package Tracking" demos. Like Federal Express, more and more companies are offering Web Services as a way to provide interactive "live" information to a variety of applications.

BBj® 9.0 showcases the new Web Services Configuration tool in Enterprise Manager (EM). A few modifications to your old code, a little configuration in EM, and you can offer up your legacy program as a Web Service that employees can use anywhere they have installed a client.

To see just how easy it is to create a Web Service, follow the online detailed instructions at the link below to configure and run the "Chile Company Web Service" demo. ■

There is more!
Continued at
www.basis.com/advantage/mag-v13n1/webservicesdemo2.pdf

By Shaun Haney
Quality Assurance Engineer

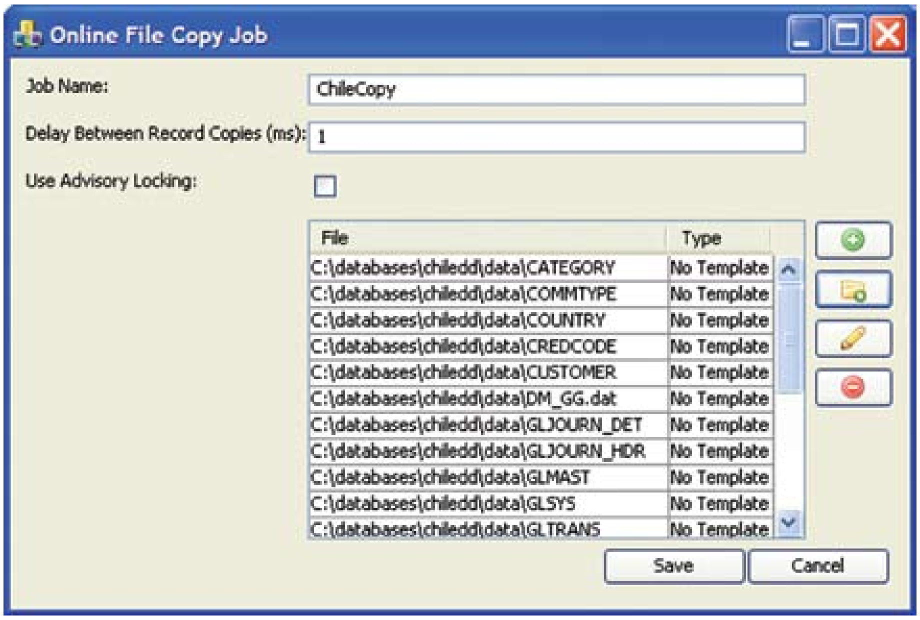


Online Copy Jobs

BBj® applications typically access data using two different methods: direct file access, and SQL. In either case, the system writes and reads data to and from individual data files. If it became necessary to make a backup copy or make structural changes to one or more data files, it was necessary to first notify all users that the file (or sometimes the entire system) is unavailable, then perform the necessary backup or file modifications. In many installations, this was and still is not an option due to such business requirements as 24/7 availability. The Online Copy Job feature provides the ability to make a backup copy of a file or changes to the structure of a file while users are still making changes to that file! This article discusses when, where, and how to use this new feature.

Online Backup Jobs

An online copy/backup consists of a definition file in XML format that contains a list of files to copy, the destination, and various parameters and settings. The beauty of using an Online Copy Job to perform a backup of one or more data files is that the backup can occur even while one or more BBj or ODBC/JDBC applications are reading or writing to those files. When the backup job begins, BBj Services creates a new file with the same structure as the original. It then begins copying records from the source file to the destination file. If any changes occur to the source file, those changes are propagated to the destination file. Once the contents of the destination file match the source file, synchronization continues until an administrator finishes the job or aborts the job by removing the destination file(s). Once the job completes, the destination file(s) no longer continues to synchronize with the original source file(s). >>



Job Name: ChileCopy

Delay Between Record Copies (ms): 1

Use Advisory Locking: ☐

File	Type
C:\databases\chiledd\data\CATEGORY	No Template
C:\databases\chiledd\data\COMMTYPE	No Template
C:\databases\chiledd\data\COUNTRY	No Template
C:\databases\chiledd\data\CREDCODE	No Template
C:\databases\chiledd\data\CUSTOMER	No Template
C:\databases\chiledd\data\DM_GG.dat	No Template
C:\databases\chiledd\data\GLJOURN_DET	No Template
C:\databases\chiledd\data\GLJOURN_HDR	No Template
C:\databases\chiledd\data\GLMAST	No Template
C:\databases\chiledd\data\GLSYS	No Template
C:\databases\chiledd\data\GLTRANS	No Template

Figure 1. Creating an Online Copy Job

By Jeff Ash
Software Engineer

Creating a backup job is quite simple, as shown in **Figure 1**.

1. Create a new Online Copy Job in Enterprise Manager.
2. Click the [Add Folder] button.
3. Select the folder of files to back up.

It is important to note that online copies can only occur on Mkeyed, Xkeyed, Vkeyed, Direct, Sort, and Serial files. Files of other types will be copied, however, the copy will not maintain synchronization of any changes made to the source file during the copy operation.

File Structure Modification Jobs

There are a number of instances where it might also be useful to modify the structure of one or more data files (record layout, record size, keys/indexes, file type, etc.) while they are in use by an application.

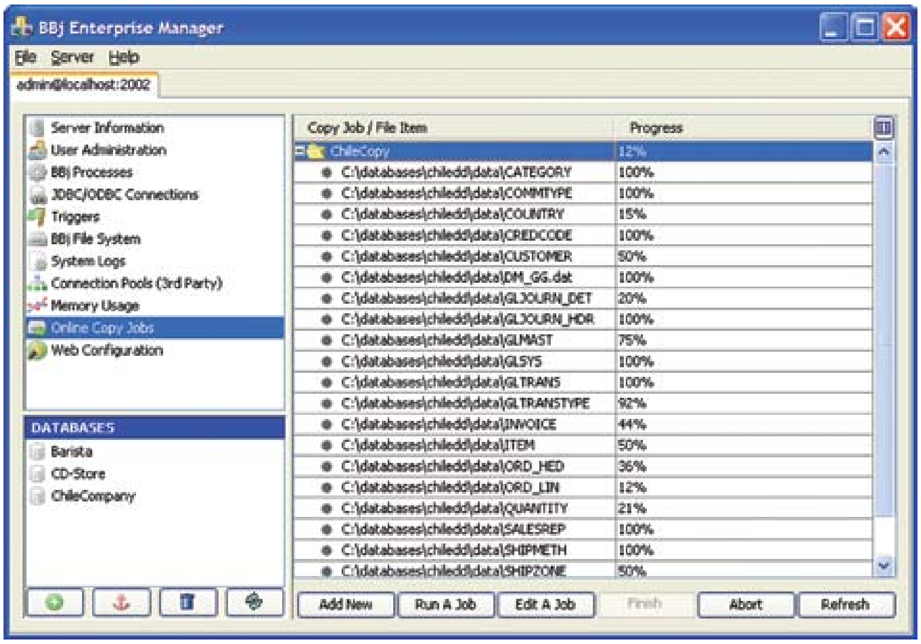
For example, it is very common for a major application upgrade to require additional fields and keys/indexes in one or more data files to accommodate new application functionality. In a large database, it might take hours or even days to update all of the necessary data files to the new structure. However, it may be undesirable or frankly impossible, due to business requirements, to prevent anyone from using the system during this time. Online Copy Jobs can make this upgrade process easy and pain-free.

The following steps outline the process for performing this type of upgrade:

1. Create a new Online Copy Job in the Enterprise Manager.
2. Add the file to upgrade to the job.
3. Specify the template used to describe the source file, and the template that describes the record layout of the destination file. Add any new columns to the destination template, and remove any columns that should not be included in the new destination file.
4. Specify the column mapping from source columns to destination columns.
5. Specify any key/index changes that should be made to the destination file.
6. Repeat steps 2 through 5 for each file to upgrade.

Running a Job

Once administrators or developers create a job, they can run the job immediately or at another time. Enterprise Manager shows a list of jobs currently running on the server (see **Figure 2**), along with a snapshot of the progress of any running jobs and the progress of each file in the job.



Copy Job / File Item	Progress
ChileCopy	12%
C:\databases\chiledd\data\CATEGORY	100%
C:\databases\chiledd\data\COMMTYPE	100%
C:\databases\chiledd\data\COUNTRY	15%
C:\databases\chiledd\data\CREDCODE	100%
C:\databases\chiledd\data\CUSTOMER	50%
C:\databases\chiledd\data\DM_GG.dat	100%
C:\databases\chiledd\data\GLJOURN_DET	20%
C:\databases\chiledd\data\GLJOURN_HDR	100%
C:\databases\chiledd\data\GLMAST	75%
C:\databases\chiledd\data\GLSYS	100%
C:\databases\chiledd\data\GLTRANS	100%
C:\databases\chiledd\data\GLTRANSTYPE	92%
C:\databases\chiledd\data\INVOICE	44%
C:\databases\chiledd\data\ITEM	50%
C:\databases\chiledd\data\ORD_HED	36%
C:\databases\chiledd\data\ORD_LIN	12%
C:\databases\chiledd\data\QUANTITY	21%
C:\databases\chiledd\data\SALESREP	100%
C:\databases\chiledd\data\SHIPMETH	100%
C:\databases\chiledd\data\SHIPZONE	50%

Figure 2. EM displaying a snapshot of an Online Copy Job in progress

When a backup job is 100% complete and the end users confirmed they have reached the desired 'known point' such as month-end processing, the administrator arranges for all users to exit the application, then clicks the [Finish] button to stop the synchronization. This effectively creates a “snapshot” of the files in the database at that point in time, like month-end. For modification jobs, once the job finishes and the users are out of the system, all that remains is to replace the original file(s) with the newly created file(s) and install the new software if necessary. For example, adding new keys to a Xkeyed, Vkeyed or Mkeyed file or converting from a single keyed Mkeyed file to a multi-keyed Mkeyed file would only require changes to utilities that re-created the files or examined the key structure. When the users log back in, they will do so with the upgraded application using the newly structured database without the traditional long downtimes previously associated with upgrades that required the restructuring of the database.

Summary

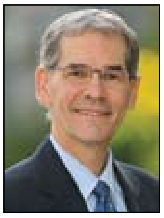
The Online Copy Jobs feature provides a powerful and flexible means for performing two very common tasks – creating a backup of files and upgrading data files while they are in use by an application. Developers and administrators will find that this powerful new feature in BBj makes the job of upgrading an application database or making numerous changes to one or more data files much less daunting. Implementing Online Copy Jobs means that end users are no longer inconvenienced by halting productive work during a major file upgrade, a 'known point' backup, or routine system backup. ■

Can Your App Speak to Your Customer?

"Sprechen ze Deutsch?"

BASIS International Ltd is, as its name suggests, an international company that does business either directly, or through its partners, in some 70 countries around the world. The installation and localization of the base product is delivered in seven languages. About a year ago, BASIS took on the challenge of translating or "localizing" text in the Barista® Application Framework and the Barista-built AddonSoftware® ERP suite. The task was to replace hard-coded strings found in the program code, selection lists, resource files, and messages with an efficient mechanism that would allow for translations to virtually any language.

Because this situation is not uncommon among the thousands of BASIS customers spread across the globe, BASIS decided to provide a generic solution in the form of a new set of translation utility classes based on Java resource bundle concepts and a GUI front end that helps developers identify and convert hard-coded strings in programs and resource files.



By Ralph Lance
Software Engineer

BBTranslator and BBTranslations

BBTranslator and BBTranslations are BBj® CustomObjects whose classes handle just about everything having to do with maintaining and finding translations contained in a *resource bundle*.

A resource bundle is a named set of property files. The property files are plain ASCII text files containing a key=value pair for each translatable phrase. Special characters are UTF-8 (Unicode) encoded and the en/decoding is handled by the translation classes automatically. One such line for a German translation for the key “Actual_Size” might look like the following:

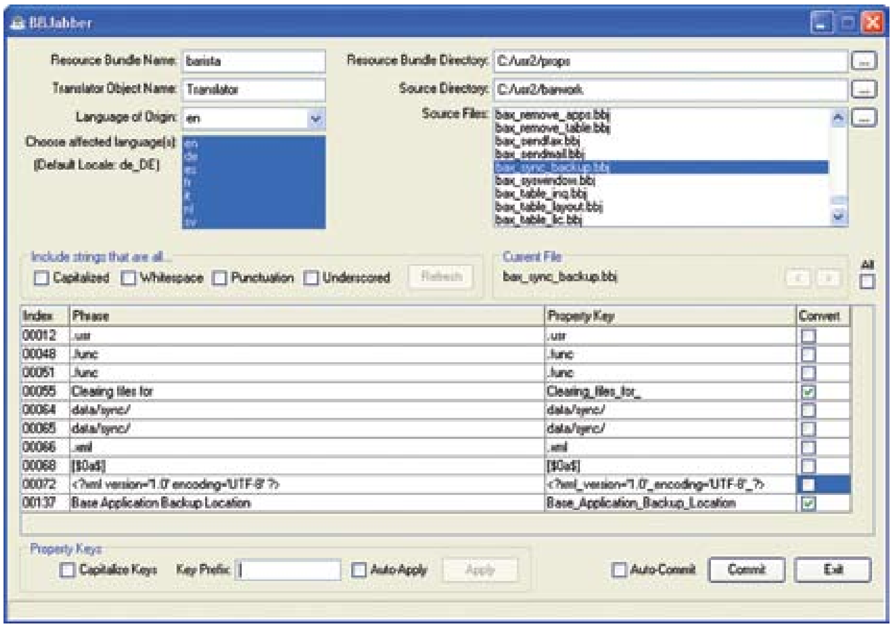
```
Actual_Size=Normale Gr\u00F6\u00DFe
```

In our example, the resource bundle is named “barista.” A default property file, **`barista.properties`**, always contains all translation keys and acts as the “fallback” property file, should a specific translation key not be found in one of the language specific property files. Language-specific property files are identified using the base bundle name plus the locale for a language. For example, generic German would be **`barista_de.properties`**. Locales can be dialect specific, if need be, by including a country and even a variant. For example, the locale **`de_AT`** would mean Austrian German. The translation search logic works its way from the most specific translation based on the target locale to the least specific and, as a last resort, to the entry in the default property file. Refer to the link that appears at the end of this article for more about locales.

Integrating the translation functionality into Barista is the same process for any BBj program and means having to:

1. Find hard-coded strings in the program code and ASCII resource files,
2. Build an appropriate translation key,
3. Store the key and its text in property files, and
4. Replace the string with a BBTranslator::getTranslation() method call in the programs, passing it the translation key to be used to find the translation. >>

For performance reasons, BASIS decided to generate language-specific sets of resource files for the Barista framework forms. Enter BBJabber!



BBJabber

Resource Bundle Name: barista
Translator Object Name: Translator
Language of Origin: en
Choose affected language(s): en, de, es, fr, it, nl, sv
(Default Locale: de_DE)
Resource Bundle Directory: C:/usr2/props
Source Directory: C:/usr2/barwork
Source Files: bax_remove_apps.bbj, bax_remove_table.bbj, bax_sendfax.bbj, bax_sendmail.bbj, bax_sync_backup.bbj, bax_syswindow.bbj, bax_table_inq.bbj, bax_table_layout.bbj, bax_table_lic.bbj

Include strings that are all...
☐ Capitalized ☐ Whitespace ☐ Punctuation ☐ Underscored
Refresh
Current File: bax_sync_backup.bbj
All ☐

Index	Phrase	Property Key	Convert
00012	.usr	.usr	☐
00048	.func	.func	☐
00051	.func	.func	☐
00055	Clearing files for	Clearing_files_for_	✓
00064	data/sync/	data/sync/	☐
00065	data/sync/	data/sync/	☐
00066	.xml	.xml	☐
00068	[$0a$]	[$0a$]	☐
00072	<?xml version='1.0' encoding='UTF-8' ?>	<?xml_version='1.0'_encoding='UTF-8'_?>	☐
00137	Base Application Backup Location	Base_Application_Backup_Location	✓

Property Keys
☐ Capitalize Keys Key Prefix: ☐ Auto-Apply Apply
☐ Auto-Commit Commit Exit

Figure 1. Managing phrases in the translation process

BBJabber

BBJabber is a utility that uses BBTranslator classes to assist developers in identifying strings in non-tokenized BBj programs and ASCII resource files that may need to be translated. Suggested translation keys are built automatically and presented before final commitment. Developers can modify translation keys and exclude individual phrases from the translation process as shown in **Figure 1**. The Translator Object Name is the object name to be used in the code and defaults to "Translator", if nothing is entered.

Committing changes for programs replaces the original string in the code with **`Translator!.getTranslation(<translation_key>)`** and stores the translation key and its text in the default property file and the property file of the "language of origin." The Language of Origin is simply the locale of the original text, e.g. **`en`** in the case of Barista.

BASIS ASCII Resource Files

Unlike BBj programs that are changed in place, committing changes for ASCII resource files will not change the ASCII resource file, but stores the translation keys and their text in the resource bundle. After translation, target language versions of the original ASCII resource files can then be generated. An example program, **`translateArcs.bbj`**, that performs this task appears in the online whitepaper *BBJabber Translation Utility*.

Because every program environment can be different, BBJabber does not include the code for the instantiation of the Translator Object. It is the responsibility of the programmer to provide for this, similar to the following code snippet:

```
rem --- Get Translator Object
use ::bbtranslator.bbj::BBTranslator
declare BBTranslator Translator!
Translator!=BBTranslator.getInstance("barista",stbl("+USER_LOCALE"),null(),dir("")+stbl("+DIR_SYR"))
```

Summary

To complete our localizing story, selection lists, messages, and elements used in building non-framework forms, including those belonging to vertical applications, are also functions that Barista writes to property files using keys built from Barista's own data dictionary information. BASIS Europe personnel has successfully and seamlessly translated the resulting 3,000 phrases in the property file to German, Italian and French. Translation into Spanish, Swedish, and Dutch was completed by a professional translation service. The result … Barista now speaks seven languages, thanks to these three new utilities! ■

For more information about locales, visit Sun Microsystems.

For more detailed information and a sample program that builds ASCII resource files in other languages, follow the online tutorial, *BBJabber Translation Utility*.

Object-oriented Performance

One of the traditional reasons for resisting the move from a legacy procedural programming style to an object-oriented programming style is runtime performance. Initially, refactoring monolithic procedural code into a group of small, individual classes and methods can introduce some performance regressions into a product. Object-oriented design principles allow programs to reuse more code, however. Reuse of code means fewer lines of code, which translates to shorter development time and fewer defects, allowing a greater focus on performance and new features. Furthermore, optimizations on a widely used method can benefit not only the original caller, but all callers that use it. This article reveals just how BBj® 9.0 used some of these very same principles in the BASIS codebase to provide significant performance enhancements to object-oriented BBj code.

Analysis

Amdahl's law is a theoretical statement about program optimization. One form of it argues that optimizing the portion of a computation that takes the longest amount of time produces the greatest performance gain.

Consider the implementation of some program operation. Each line of code in that implementation represents a potential optimization candidate. When a program spends all its time in one small section of code, the optimal candidate for optimization becomes obvious. But when there are so many lines of code or logical sections of code that each one only contributes a very small percentage of the total, the return on the effort required for any individual optimization project will seldom be compelling.

This is where object-oriented design can help. First, organizing operations into objects helps to limit the scope of an optimization. Optimizations can occur on an individual method or on a whole class, but properly decoupling unrelated objects prevents an optimization from "spilling over" into other code. Caching strategies can be hidden within the implementation of an object, and when the general codebase uses such an object and its methods often, optimizations to that object have wide applicability. Object-oriented design also provides the ability to view operations at different granularities to select the appropriate level to optimize. >>

By Adam Hawthorne
Software Engineer

This often requires restructuring some of the codebase. Cut and pasted code must be reabsorbed into a single method and reused. Functionality implemented slightly differently in different portions of the codebase must be factored into generally useful objects and then used everywhere. These all serve to increase the effect of any individual optimization as well as increasing maintainability and often testability.

Using a profiler to find regions of a program that run slowly is also critical to solving performance issues. This allows application of Amdahl's law by discovering those regions that take the most time first.

In the codebase for BBj Services, the code that makes object-oriented programming available to BBj developers is itself object-oriented. Previous releases of BBj had already addressed the easy performance gains, and profiling did not show any area that was significantly slower than any other. Applying some of the design principles above and refactoring the code to co-locate more of the implementation allowed the code profiler to show the critical obstacles to better performance in BBj.

This led to a number of proposed enhancements, and as the results show, better performance for BBj developers.

Results

Custom Object method calls saw the greatest improvement, but all method calls, both Java and Custom Object, are now faster than in previous releases. Overloaded parameter types and the number of methods that can match a given call site affect the optimization, but the most modest improvements are still 30% faster in BBj 9.0 than in BBj 8.31. In the best case, the time to perform a particular method call decreased by a factor of 134! The performance improvements are even more noticeable when the Custom Object definitions span several files. Notably, the time to invoke a method is now virtually the same as the time to perform a CALL when both invocations have the same number of parameters. Even the venerable CALL verb saw a small boost in performance, about 4%, in BBj 9.0.

There are even greater benefits available by leveraging some of the existing language tools. BBj 6.0 introduced the DECLARE verb to assign a type to a particular variable. The DECLARE verb instructs BBj to enforce static type checking rules on expressions involving variables used in DECLARE statement. In that release, DECLARE allowed for BBjCpl to warn about type checking errors via the -t command line option, and for the BASIS IDE to provide code completion on variables with DECLARE statements. BBj 9.0 further improves the benefit of using DECLARE statements. Method calls that only use variables specified by DECLARE statements (or expressions on those variables) can reach 90% improvement in execution time. Even the simplest method invocations are up to 10% faster on average than the same method call that includes an expression with an undeclared variable. DECLARE statements also allow BBj to infer guarantees about the object code in BBj programs that may lead to even further enhancements in the future.

Summary

Object-oriented code has the benefit of reorganizing a program to highlight similar pieces of code. This leads to refactoring opportunities, which reduces the number of lines of code. The fewer lines of code there are, the easier it is to find performance bottlenecks, and it is often easier to optimize them because of better organization. Using version 9.0, BBj developers benefit indirectly from the use of these principles, and directly, by writing decoupled, well-factored, object-oriented Business BASIC code themselves. ■

BASIS Goes Social!

Over the past few years, we have see an explosion of new technology in the world of social networking. BASIS now uses that technology to deliver the most up-to-date developments, news, and happenings. We will be posting breaking news from our developers in order to share a glimpse of our newest technology with our customers around the globe.

Follow us on Facebook or Twitter to be the first to get the Communiqués and event announcements. See demos of the newest technology on YouTube.

By Amer Child
Training and Sales Specialist

Click these links at www.basis.com

A Sneak Peek at BBj's Browser User Interface

Even though many of us pre-date the Internet and Web browsers, it is almost impossible to imagine our personal, professional, and social lives without them. The Internet has dramatically modified the way that we interact with businesses and colleagues, not to mention friends and family. We make travel and hotel reservations online, check our investments and bank accounts, research and purchase products, and more on the Internet via our trusty Web browser. It only seems natural to also run and access our business applications over the Internet via our Web browser. The new Browser User Interface (BUI) in BBj® 10.0 promises us exactly that!

Below are some BUI FAQs that inquiring minds may want to know.

Q. What is BUI?

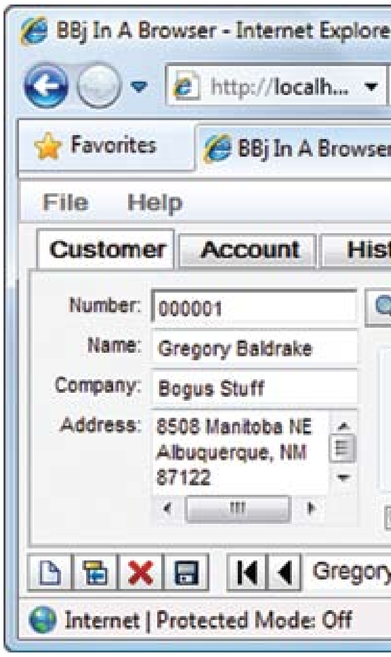
A. The concept behind BUI (pronounced: *boo'-ee*) is pretty simple - Visual PRO/5® and BBj developers can leverage BBj's integration of the Google Web Toolkit (GWT) to run their new and existing GUI applications in a browser. The GWT cross-compiles Java-based code, such as BBj, into optimized JavaScript that automatically works across all major browsers. That means that GUI Business BASIC applications now run in a variety of different browsers (Internet Explorer, Chrome, FireFox, Safari, etc.) on multiple platforms – even smart phones and other Web-capable mobile devices. Instead of the traditional Web Start/Thin Client prerequisite that end users must have a Java JVM installed, BUI bypasses the JVM requirement for many applications and merely requires a JavaScript-enabled Web browser. Certain technologies, like Java ClientObjects, will still require a JVM on the client but many applications can eschew this requirement.

By Nick Decker
Engineering Supervisor

Q. What does a BUI application look like?

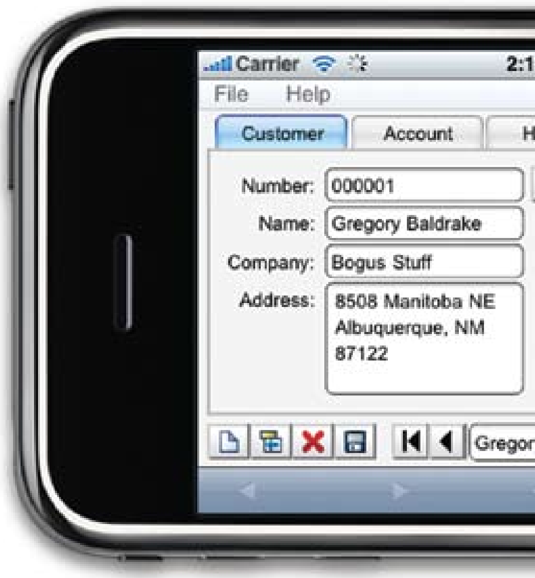
A. Since GWT maps traditional GUI controls such as buttons, text boxes, dropdown lists, etc. to native browser controls, existing BBj applications look remarkably similar to their SysGUI counterpart. However, because the application is running in a browser and different browsers sometimes render controls differently, your application's look and feel may be slightly different compared to the look and feel offered by the native operating system.

Additionally, some browser functionality such as File Open/Save dialogs, may not be as full-featured as those offered by an application running in SysGUI. On the other hand, running in a browser definitely improves the native support for HTML formatted content. BASIS plans to offer custom control styling in the future via Cascading Style Sheets (CSS). **Figures 1-3** show the same BBj GUI application running in three different environments – the traditional SysGUI, the new BUI in Internet Explorer, and BUI in Safari on an iPhone. >>



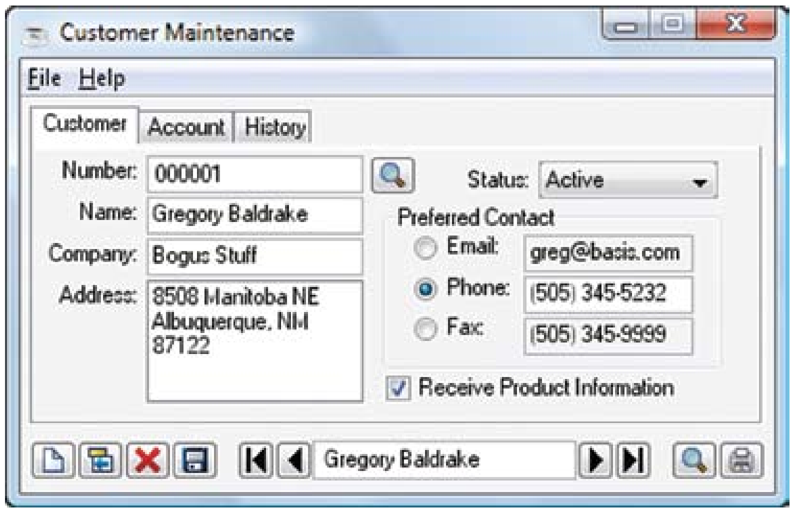
Number: 000001
Name: Gregory Baldrake
Company: Bogus Stuff
Address: 8508 Manitoba NE
Albuquerque, NM
87122
Gregory

Figure 2. The Customer Maintenance progran



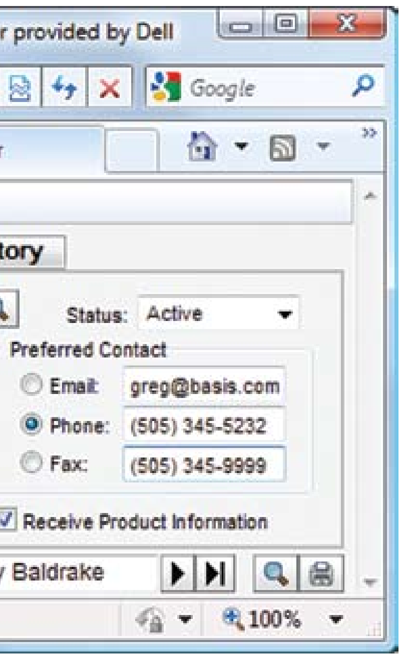
Number: 000001
Name: Gregory Baldrake
Company: Bogus Stuff
Address: 8508 Manitoba NE
Albuquerque, NM
87122
Gregor

Figure 3. The Customer Maintenance program running in BUI



Number: 000001
Status: Active
Name: Gregory Baldrake
Company: Bogus Stuff
Address: 8508 Manitoba NE
Albuquerque, NM
87122
Preferred Contact
Email: greg@basis.com
[✓ Phone:] (505) 345-5232
Fax: (505) 345-9999
✓ Receive Product Information
Gregory Baldrake

Figure 1. The Customer Maintenance program running in SysGUI

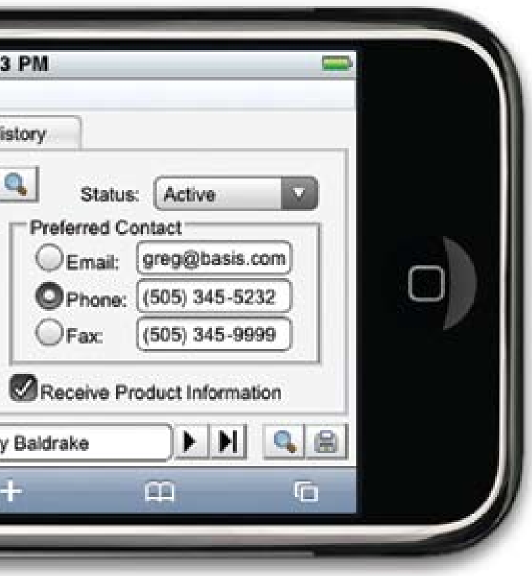


Status: Active

Preferred Contact

Email:	greg@basis.com
Phone:	(505) 345-5232
Fax:	(505) 345-9999

n running in BUI in Internet Explorer



Status: Active

Preferred Contact

Email:	greg@basis.com
Phone:	(505) 345-5232
Fax:	(505) 345-9999

in Safari on an iPhone

Q. What changes does BUI require in my Visual PRO/5 or BBj code?

A. Our goal for BBj 10.0 is to ensure that your BBj GUI application will require little or no code changes to run in BUI. We are striving to make the transition as seamless as possible, but there may be cases where a particular paradigm or function does not translate well into the browser environment. Additionally, you may want to reduce 'round trips' as much as possible for some architectural and performance considerations that apply to any Web-based application. For example, you may decide to replace field level validation with form level validation. For Visual PRO/5 code, you will need to port it to BBj, which is generally similar to previous efforts to port code from BBxPROGRESSION/4® to PRO/5®. For more on the process, refer to *Converting to BBj from Earlier Versions of BASIS Products* in the online documentation.

Q. Will Barista-developed applications run in BUI?

A. Yes, Barista applications are GUI BBj applications and we will be testing them extensively so that they will run as well in BUI as any other application.

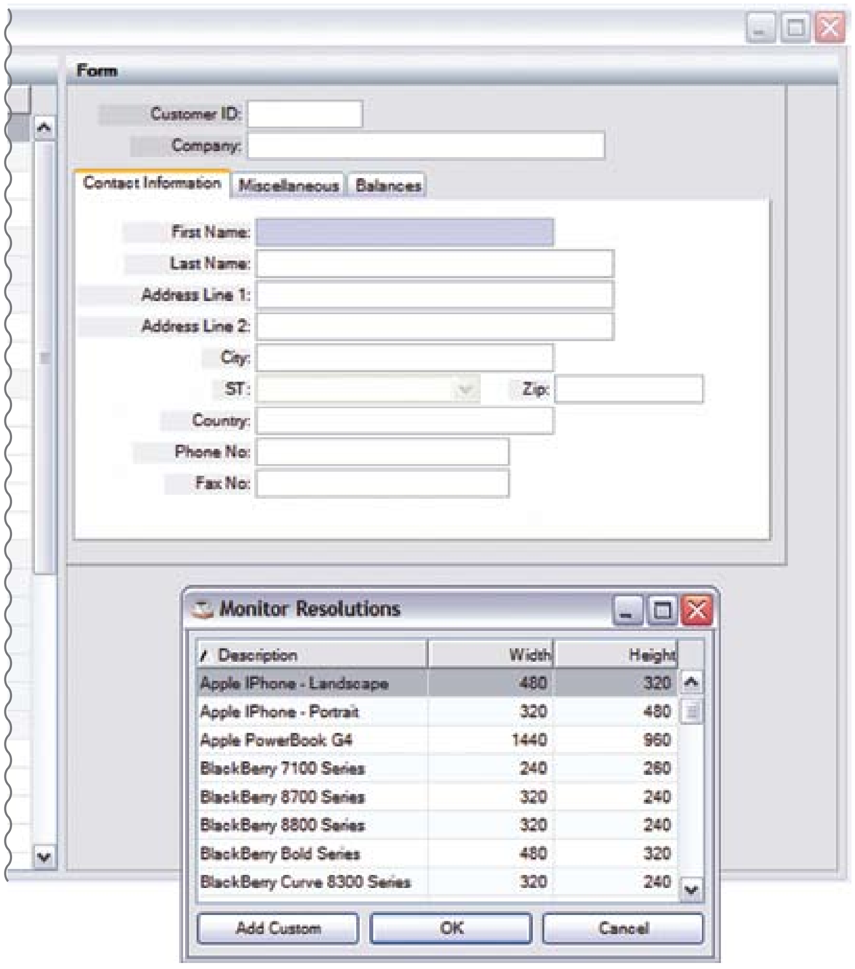
Q. When can I start coding my application for BUI?

A. There is no time like the present so code your application today in BBj via Barista, AppBuilder, or by hand coding in the IDE's syntax aware and code completion editor. The Barista Form Designer even offers various screen layout frames, as shown in **Figure 4**, to ensure that the form will fit on smaller form factor browsers such as a mobile device like an iPhone.

And the $64M Question....

Q. When can I begin playing with BUI?

A. (This was the number one question we received at TechCon2009 after demonstrating several BBj applications running in BUI, a true indication of the excitement and promise that BUI offers!) Formal release is scheduled for the end of Q1 2010 but developers were pleased to learn that it would be available for preview in the BBj nightly downloads at the beginning of 2010 and fully supported upon release of BBj 10.0. So anchors aweigh....get ready to float your boat with BUI! ■



Description	Width	Height
Apple IPhone - Landscape	480	320
Apple IPhone - Portrait	320	480
Apple PowerBook G4	1440	960
BlackBerry 7100 Series	240	260
BlackBerry 8700 Series	320	240
BlackBerry 8800 Series	320	240
BlackBerry Bold Series	480	320
BlackBerry Curve 8300 Series	320	240

Figure 4. Barista's Form Designer dropdown of various pre-configured and custom-built frames

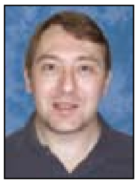
The BASIS Java Hat Trick

In ice hockey, a hat trick is a rare and exciting occurrence when a player scores three goals in one game. Similarly, BASIS scores with these three e-zine articles that collectively discuss how the extension of BBj® facilitates the use of Java and third party Java libraries.

Here is an overview of what you can read in greater depth online.

Calling Custom Java from BBj

Learn how to complete standard Java libraries to do the work for you



By Jim Douglas
Software Engineer

BBj is two languages in one. It combines the business focus and ease-of-use of Business BASIC with the power of Java. In *Quick and Easy Solutions With Free Java Libraries* (Part 1 and Part 2), we showed how to find and integrate third party Java components into your BBj application. This article shows how easy it is to integrate your own custom Java modules into BBj. ■

Follow along at
www.basis.com/advantage/mag-v13n1/customjava.pdf

Inheriting Java Types

Take Java code and extend it using BBj

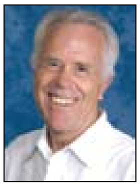
By Adam Hawthorne
Software Engineer

The introduction of Java syntax to BBj® gave developers a vast library of existing code to add to their toolboxes. Custom Objects further offered the ability to transform the way developers write code. BBj now bridges the gap between BBj and Java by allowing developers to extend Java classes and implement Java interfaces with BBj CustomObjects. ■

Follow along at
www.basis.com/advantage/mag-v13n1/javatypes.pdf

Session-specific Classpaths - Use Jars Dynamically

Add a new library without having to restart the application



By David Wallwork
Senior Software Architect

BBj 9.0 introduced session-specific classpath (SSCP) to allow developers to set the Java classpath for BBjSessions on a per-session basis. This capability is especially useful during the development of custom Java code or when the developer wishes to test a third party Java library without restarting BBj Services. This article explores how SSCP positively impacts the development cycle, and more! ■

Follow along at
www.basis.com/advantage/mag-v13n1/sscp.pdf

Regularly served to your browser @10:00 AM MT

java break with BASIS International

Nov 18, 2009	iReport, BBJasper, SPROC Auto Creation
Dec 02, 2009	Browser User Interface
Dec 16, 2009	Online Copy and Live Back-up
Dec 30, 2009	Barista Building an App - CUI to GUI
Jan 13, 2010	Java and BBj
Jan 27, 2010	Barista - Third Party RDBMS

Jetty Offers Legacy Programs via Web Services

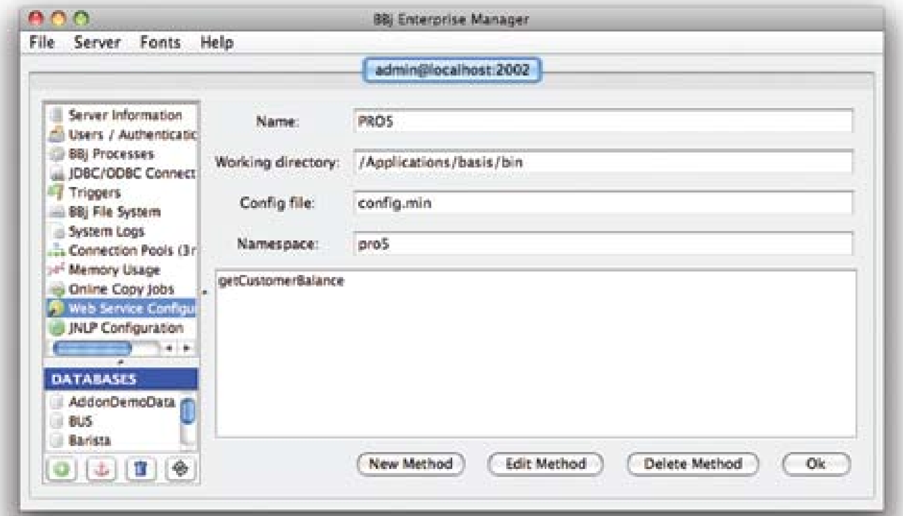
Enterprise Manager's new Web Services tool makes it easy to connect your application to BBj®'s new built-in Jetty Server, and ultimately to the world. The Jetty HTTP Server opens your application's API to clients anywhere on the Internet. Reliance on Internet standards such as HTTP and XML simplifies interoperability. Web Services help bridge the gap between BBj and other languages. If you need to get two systems working together to share data or business logic, whether they are across the room or across the globe, Web Services are a great solution. This article explores BBj's ability to turn an existing application into a Web Service with very little effort.

Create a Web Service

Enterprise Manager's new Web Service Configuration tool makes it easy to create a new Web Service. Selecting a name, working directory, config file, and namespace (see **Figure 1**) creates a new empty API, ready to populate with methods from your application.

BBj implements a Web Service method as a simple CALL to a BBx® program. Adding a new method to the API requires filling in only a few fields. First, select the public name of the method; the method name the Web Service clients will see in the

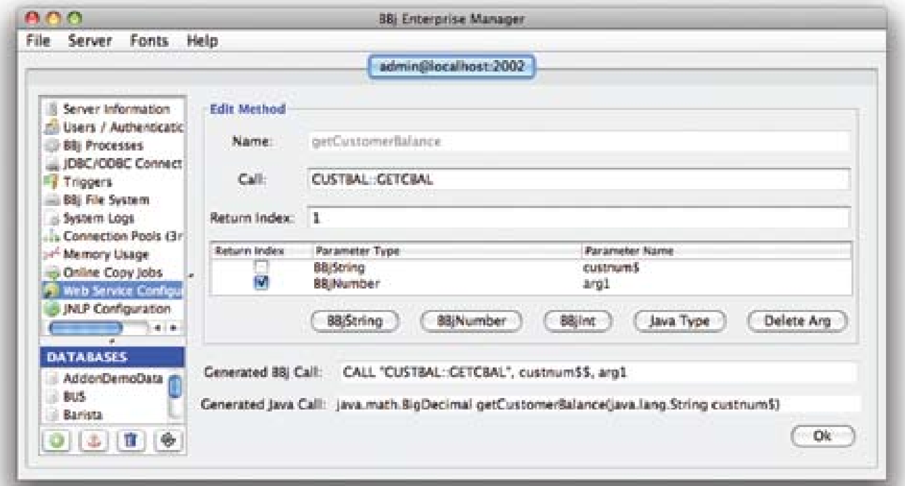
>>



Name: PRO5
Working directory: /Applications/basis/bin
Config file: config.min
Namespace: pro5

getCustomerBalance

Figure 1. A Web Service defined in Enterprise Manager



Name: getCustomerBalance
Call: CUSTBAL::GETCBAL
Return Index: 1

Return Index	Parameter Type	Parameter Name
☐	BBjString	custnum$
✓	BBjNumber	arg1

Generated BBj Call: CALL "CUSTBAL::GETCBAL", custnum$$, arg1
Generated Java Call: java.math.BigDecimal getCustomerBalance(java.lang.String custnum$)

Figure 2. The getCustomerBalance Web Service method

By Jason Foutz
Software Programmer

public API for your Web Service. This example uses "PRO5" as the name of the Web Service, and getCustomerBalance as the method name, a long name to help users of the service determine the purpose of the method. Next, provide the name of any BBx program that includes an ENTER statement. The CUSTBAL program looks up customer balances. Add the CALL arguments the client should provide. The GETCBAL entry point requires a BBjString to identify the customer, and a BBjNumber to report the current balance. In BBx, all the variables of an ENTER statement are visible to the calling program, but a Web Service method may only return one variable to the client. Selecting a check box next to one of the listed types will instruct the Web Service method to use the argument at that position as the return value, as shown in **Figure 2**.

The Web Service named "PRO5" in the configuration dialogs defines a getCustomerBalance method. Any valid BBx program can provide the implementation of this method. The CUSTBAL program defines an entry point, GETCBAL that looks up customer balances, as shown in **Figure 3**.

```
0010 GETCBAL:
0020 ENTER CUSTNUM$,BAL
0030 LET CHAN=UNT
0040 LET TMPLT$="CUST_NUM:C(6):LABEL=CUST_NUM:, FIRST_NAME
0050 OPEN (CHAN)"data/CUSTOMER"
0060 DIM RESULT$:TMPLT$
0070 READ RECORD(CHAN,KEY=CUSTNUM$,ERR=0090)RESULT$
0080 LET BAL=RESULT.CURRENT_BAL
0090 CLOSE (CHAN)
0100 EXIT
```

Figure 3. The getCustomerBalance method for the "PRO5" Web Service

```
- <definitions targetNamespace="pro5" name="PRO5Service">
    <import namespace="http://bbjgenerated/" location="http://localhost:8888/webservice/PRO5?wsdl=1"/>
  - <binding name="PRO5ServicePortBinding" type="ns1:PRO5">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    - <operation name="getCustomerBalance">
        <soap:operation soapAction=""/>
      - <input>
          <soap:body use="literal"/>
        </input>
      - <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
  - <service name="PRO5Service">
    - <port name="PRO5ServicePort" binding="tns:PRO5ServicePortBinding">
        <soap:address location="http://localhost:8888/webservice/PRO5"/>
      </port>
    </service>
  </definitions>
```

Figure 4. The WSDL served by BBj

```
wsimport -B-wsdl http://localhost:8888/webservice/PRO5?wsdl
```

Figure 5. Create Web Service clients

Publish the Web Service

A Web Service does not provide any value unless the world can reach it. To make the "PRO5" Web Service available, you must publish the Web Service. In BBj, publishing the service makes the API available using the embedded Jetty Web Server. An XML document called a WSDL document exposes the Web Service API and associated information to Web Services tools of any programming language. To retrieve the WSDL document, simply use a Web browser to navigate to http://localhost:8888/webservice/PRO5?wsdl. An example of the document generated for the "PRO5" service appears in **Figure 4**.

As mentioned above, Jetty, a small and efficient Web Server, makes the Web Service available. Because Jetty uses standard technologies, both system and network administrators understand how to configure their systems and networks to provide access to it. Developers could write a custom networking protocol, client, and server in BBj, but such an implementation is challenging at best. Using the standard tools, technologies, protocols, and automatic features provided by BBj reduces the difficulty both for developers and administrators.

Consume the Web Service in BBj

The Web Service is published and now waiting for a client to connect. The next logical step is to write a client that uses the WSDL document to provide an API for a Web Service client to use.

Java provides tools to automatically generate Java code to provide the Web Service API described by the WSDL document. Sun-derived JDK implementations provide this ability in a tool called wsimport, located in the `jdk/bin` directory. Issue the command in **Figure 5** to generate a set of Java classes.

These classes connect to your application, allowing BBj programs >>

```
rem  consume PRO5 web service

service! = new service.PRO5Service()
port! = service!.getPRO5ServicePort()
balance = port!.getCustomerBalance("000010")
print balance
exit
```

Figure 6. A sample BBj client to the "PRO5" Web Service

to use your methods. Such tools exist for many languages; the exact steps for constructing a client vary from language to language, but the core idea of using XML to represent the information being passed from the client to BBj and back remains the same. BBj requires a reference to the newly created classes. Therefore, it needs to be added to your classpath.

After Java has generated the Java classes to link a client to a Web Service and BBj is configured to use them, the code in **Figure 6** can be used to consume the Service.

Now, clients on any platform can call "PRO5" and get customer balances... even on their mobile device (see **Figure 7**)!

Summary

BBj provides all the tools necessary to open your application to the world. Enterprise Manager helps design an API that gives your clients the access they need to your applications. By leveraging HTTP and XML, configuration is far simpler than the alternatives. Web Services can even ease the pain of communicating with applications written in other languages. Make your application available to anyone anywhere with Web Services.

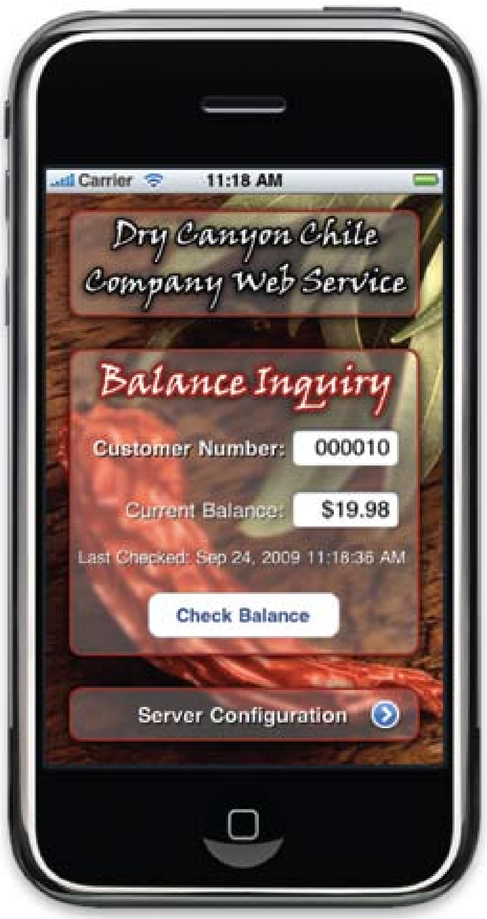


Figure 7. A mobile device using the "PRO5" Web Service

Turn a legacy (V)PRO/5 application routine into a Web Service

1. Install the latest version of BBj.

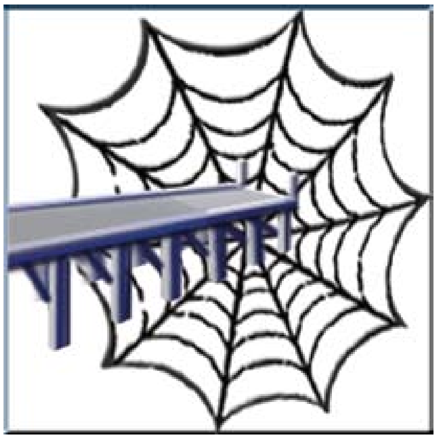
2. Follow the steps to search for reserved words in BBj using the (V)PRO/5 _keyword utility as documented in the online topic *Converting to BBj from Earlier Versions of BASIS Products* to ensure that the tokenized (V)PRO/5 application routine runs as expected in BBj.

3. Configure BBj to use Shared Locks so that it can simultaneously access data files that are used by your (V)PRO/5 application routine.

4. Create a new Web Service in Enterprise Manager.

5. Add a method to the Service that references your application routine.

6. Publish the Service.



Jetty Web Server for BBj, Java Style

Did you ever wish you had a Web Server at every site and knew how to get them configured? Or, do you know how to configure a Web Server, but wish you didn't have to spend the time? Well, now all your wishes are granted.

BBj® brings with it a built in Jetty Web Server to every installation. Before the integrated Jetty Server was available, setting up Web Start applications could be quite time consuming and a little challenging to the uninitiated. The new tools in Enterprise Manager help get your applications running in Web Start quickly and easily. In addition, several BASIS applications such as Enterprise Manager (EM), Barista® Application Framework, and AddonSoftware®, run in Web Start right from a standard installation.

For several years, BBj has been supporting thin client running in Web Start. It is a great way to get an application on a system without having to install BBjServices. Thin client in Web Start requires only a click on a Web page rather than downloading, installing, and configuring a full installation. In the past, setting up to run thin client in Web Start was difficult due to all the extra tools and configuration it involved.

BBj, Jetty, and EM make running applications in Web Start a breeze. The Web Server is ready to go as soon as the BBj installation is complete. Before BASIS integrated the Jetty Server, serving an application required installing a Web Server. Creating a new configuration in EM takes only minutes. One of the biggest stumbling blocks in any Web Start application is getting the jars signed and ready to deliver to the client. Jar signing is built into BBj's Jetty implementation so jars are signed automatically, when needed. Best of all, EM provides a sample security certificate so you can get up and running immediately.

The old way required a great deal of behind-the-scenes configuration. Developers would install a custom Web Server, copy and sign jars, and craft a custom .jnlp file for each application. Each step could take hours to complete. Now, in BBj, the whole process takes just a few minutes.

Enterprise Manager preconfigures Barista, AddonSoftware, and LaunchDock to run in Web Start. These are great examples of running full applications without requiring a BBj Services installation on each client's computer. BBj provides EM access via Web Start in a default installation, making it easier to administer BBj. Some scenarios are difficult to manage such as when administrators deploy BBj on machines that do not have GUI displays or when developers cannot log in to the server running BBj. Since BBj now provides EM access to any computer on the network via Web Start, developers can configure BBj from any available computer. There is no longer a need for a full installation of BBj on a local machine just to check memory usage on a server. Any computer with a Web browser and a network connection can launch EM.

BBj, with the integrated Jetty Web Server, makes running Web Start easier than ever and EM helps configure your application to run in Web Start. Behind the scenes, BBj's Jetty implementation takes care of jar signing and jnlp creation, saving hours of time. BBj delivers the tools you need to manage BBj, right from the integrated Web server.

Sometimes wishes do come true! ■

By Jason Foutz
Software Programmer

BASIS Generates SPROC Template Code

Now anyone can write a SPROC!

The SQL language got its start almost forty years ago and is now the standard language for querying, modifying, and managing a relational database. Developers have used SQL for years to manipulate their data, generate reports, and otherwise interact with their BASIS Databases. Despite the power of SQL, however, their data may not be structured for efficient SQL access, having been designed for the fast direct record access offered by the BBx® language syntax. The ability to easily lock data files, extract records, and do keyed reads are more appropriate for these data structures and are second nature to a BBx programmer. The SQL-counterparts are sometimes more complex, difficult to construct, or may not be available in legacy file formats. BASIS recognized this and took steps to aid developers in their quest to write BBx-driven database stored procedures – by providing an easy-to-use code generator that writes customized template code based on a file's data layout.

By Nick Decker
Engineering Supervisor

Stored Procedure Background

Stored procedures, or SPROCs for short, have become popular by helping developers open up their database to third party client access. It is possible to move all of the complex processing logic that previously existed in the primary application into a SPROC. Doing so allows a variety of new clients to access the database while retaining all of the requisite business logic and processing that the application formerly provided. Now that the SPROC is the central location for the business logic, disparate clients can access the database through the SPROC and take advantage of several years' worth of accumulated processing expertise without having to replicate that functionality in every client application.

Another strong case for using SPROCs is their ability to bypass the previous requirement that all SQL access to the database must be on normalized files to avoid an unbearable performance penalty. Many customers were initially pleased with SQL access, only to be disappointed later when queries to non-optimized and non-normalized legacy data files performed poorly. SPROCs in BBj®, with their ability to access record data with traditional READ statements, can use the same speedy routines as the legacy stand-alone applications to retrieve sought-after data, without using SQL or requiring a database overhaul and yet, deliver that data in structures acceptable to third party tools using the SQL language.

SPROCs – a CALL by Another Name

Business BASIC programmers have been using BBx's CALL statement for decades to access commonly used code and libraries of routines. In order to interface with the CALLed program effectively, a BBx program passes variables along in the CALL statement and the CALLed program uses an ENTER statement to retrieve these values. The following line of code should look familiar to most BBx programmers:

```
CALL "CUSTINFO", CUSTNUM, COMPANY$
```

In the same way that the CALL verb allows BBx programs to invoke another program, passing in variables and receiving information back, a SPROC does the same for client access to a database. In fact, even though we are now using SQL to talk to the database, the syntax is remarkably similar:

```
CALL CUSTINFO(CUSTNUM,'COMPANY')
```

>>

Not only does the syntax look strikingly similar, but developers use SPROCs in large part for the same reasons – launching a program that provides a commonly used service for several applications. You can also pass data into a SPROC in much the same way that you specify variables to pass to a public program. SPROCs are very flexible as they provide the traditional input and output variables along with advanced return types such as return codes and full-blown result sets, similar to a grid full of data.

Creating a new SPROC

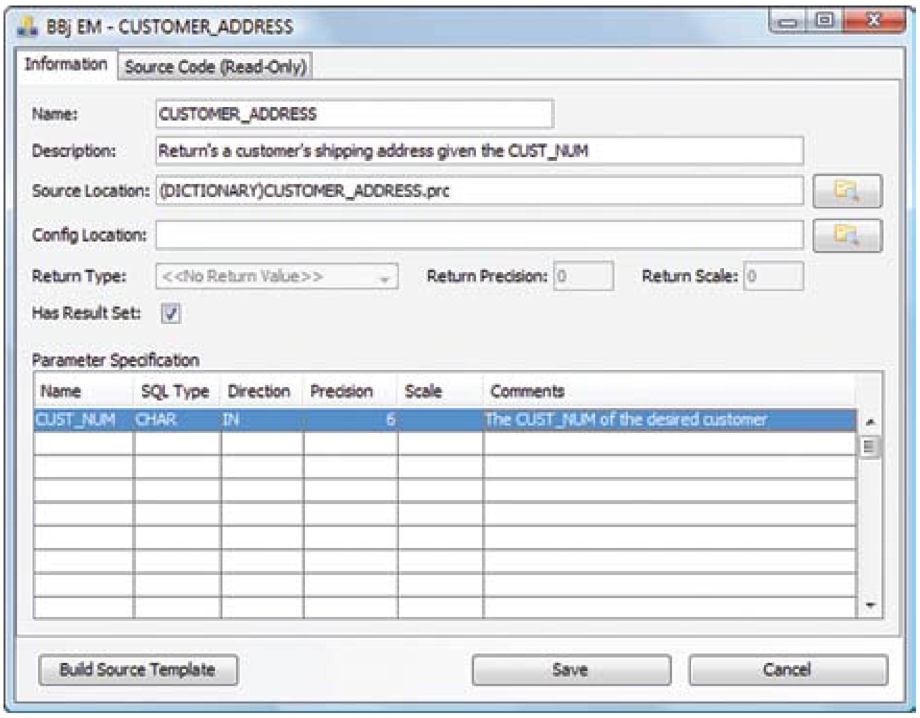
So now that everyone is sold on SPROCs and are reassured that the code is familiar, how does one go about creating one of these "called" routines/programs?

The first step is to determine what sort of database functionality the SPROC will provide, followed by the desired input and output data. In our example, we use the ChileCompany demo database and create a SPROC that returns a customer's billing/shipping address given their customer number.

Next, launch Enterprise Manager (EM) or load the EM module from within the BASIS IDE for a truly all-in-one experience. Then select the ChileCompany database entry from the list of databases in the bottom left panel. Click on the Procedures tab in the right-hand information pane for the ChileCompany, then the Plus button to add a new SPROC. A new window, as shown in **Figure 1**, appears in which you can enter all of the necessary information and full description of the new SPROC.

In this example, the SPROC's name is CUSTOMER_ADDRESS. The program file `CUSTOMER_ADDRESS.prc`, located in the Data Dictionary directory, runs when a client calls the SPROC. Selecting the "Has Result Set" checkbox indicates that it returns a result set back to the caller. The result set is a handy way to return data back to the client, especially if the SPROC might return multiple rows of data. In our example, the SPROC only returns a single row – the address for the specified customer – but the result set makes it easy to return several rows and columns worth of data.

Lastly, we have defined a single parameter called CUST_NUM that is a CHAR type with a direction of IN. This



Name: CUSTOMER_ADDRESS
Description: Return's a customer's shipping address given the CUST_NUM
Source Location: (DICTIONARY)CUSTOMER_ADDRESS.prc
Config Location:
Return Type: <<No Return Value>> Return Precision: 0 Return Scale: 0
Has Result Set: ✓

Parameter Specification

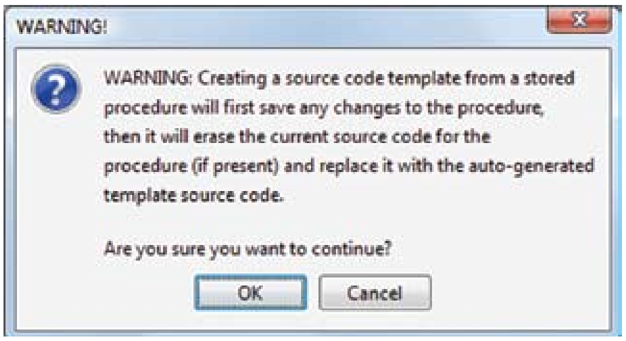
Name	SQL Type	Direction	Precision	Scale	Comments
CUST_NUM	CHAR	IN	6		The CUST_NUM of the desired customer

Figure 1. Adding a new stored procedure

means that the client must provide a string containing the customer number when calling the SPROC to specify which customer address they require.

EM Generates SPROC Code Template

The next step is where the magic happens. By clicking the [Build Source Template] button, the Enterprise Manager writes most of the SPROC code needed to make the program viable. Clicking the button results in the dialog box shown in **Figure 2** that warns it will overwrite the designated program file `(DICTIONARY)CUSTOMER_ADDRESS.prc` with a newly-generated template. If we had previously written the code for the SPROC and pushed the button by mistake, selecting [Cancel] would abort the process. Since we have not written the SPROC program, selecting [OK] causes EM to write the SPROC program file.



WARNING: Creating a source code template from a stored procedure will first save any changes to the procedure, then it will erase the current source code for the procedure (if present) and replace it with the auto-generated template source code.

Are you sure you want to continue?

Figure 2. EM's warning when creating a new template SPROC program

Customizing the SPROC's Output

Enterprise Manager gives us the opportunity to specify a string template to describe the SPROC's return result set. When the generator first created the SPROC, it only specified that the new procedure would return a result set of data back to the client. At that point in time, it was not critical to define what that result set would look like – only that the SPROC would use it to return data to the client. This makes sense, as a SPROC can be very flexible, returning different result sets back depending on which type of input parameters the client supplied. But now that the developer is getting down to the nitty-gritty of the SPROC code itself, it is time to figure out exactly what the return result set should look like.

For our example, the invoker of the SPROC needs the shipping address for the customer, so the data will be a subset of the appropriate record in the ChileCompany's CUSTOMER data

>>

file. To access the CUSTOMER table's string template, click on the Tables tab in EM then double-click the CUSTOMER entry in the table list. The resultant window brings up the properties for the CUSTOMER table and offers a button called [String Template] as shown in **Figure 3**. Click the button to show the string template for the data file in a text box and copy the address portion into the clipboard.

Equipped with the desired section of the customer table's string template, paste it into EM's dialog to define the columns that comprise a record in the return result set (see **Figure 4)**.

After clicking [OK], EM writes out the template SPROC code. Now save the fully defined SPROC to test it. Notice that the list of stored procedures in EM now contains the new SPROC at the top of the list as shown in **Figure 5**.

Testing the SPROC

Because EM wrote a full-blown program for the backend of the SPROC, it is possible to try it out right away. Obviously, it will not work exactly as needed, since EM cannot read minds and the developer did not tell it from which file to get the data. However, it did create a fully functioning program that will return sample data when CALLed by a client. To take it out for a test drive, execute an SQL CALL statement in the SQL tab. EM saves time here again, as the SPROC listing (as shown previously in **Figure 5**) also contains sample SQL code to invoke the SPROC. The simplest way to proceed is to copy the sample SQL code from the SPROC line, click over to the SQL tab, and then paste it into the SQL Statement box (see **Figure 6**).

Executing the SQL statement causes the BASIS DBMS to run the BBx program that defines the SPROC, returning a result set with the fields specified in the string template. After verifying that the new SPROC works without error, a fully functional SPROC is just around the corner, having already created a new SPROC, asked EM to write out a functional template program, and ensured that the SPROC actually works. The final step is

>>

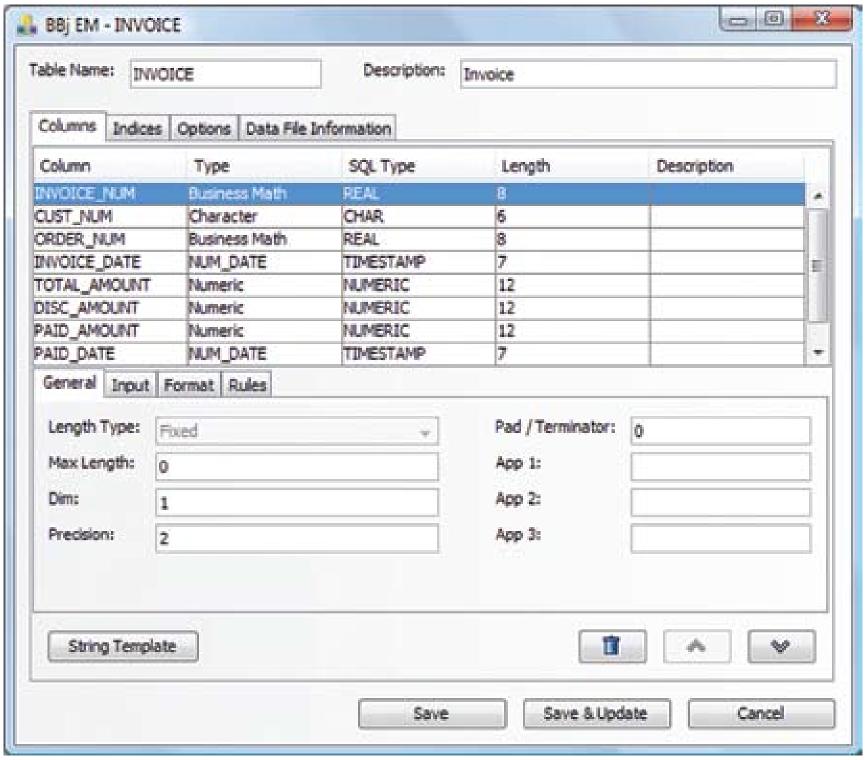
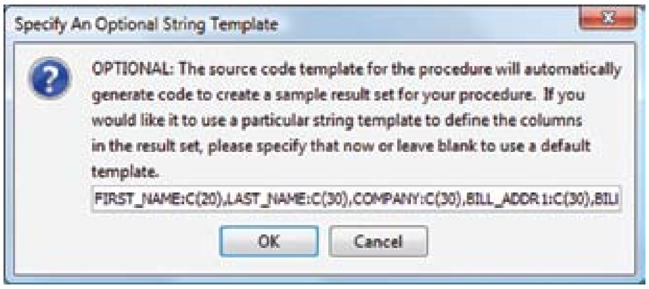


Table Name: INVOICE Description: Invoice

Column	Type	SQL Type	Length	Description
INVOICE_NUM	Business Math	REAL	8	
CUST_NUM	Character	CHAR	6	
ORDER_NUM	Business Math	REAL	8	
INVOICE_DATE	NUM_DATE	TIMESTAMP	7	
TOTAL_AMOUNT	Numeric	NUMERIC	12	
DISC_AMOUNT	Numeric	NUMERIC	12	
PAID_AMOUNT	Numeric	NUMERIC	12	
PAID_DATE	NUM_DATE	TIMESTAMP	7	

Length Type: Fixed Pad / Terminator: 0
Max Length: 0 App 1:
Dim: 1 App 2:
Precision: 2 App 3:

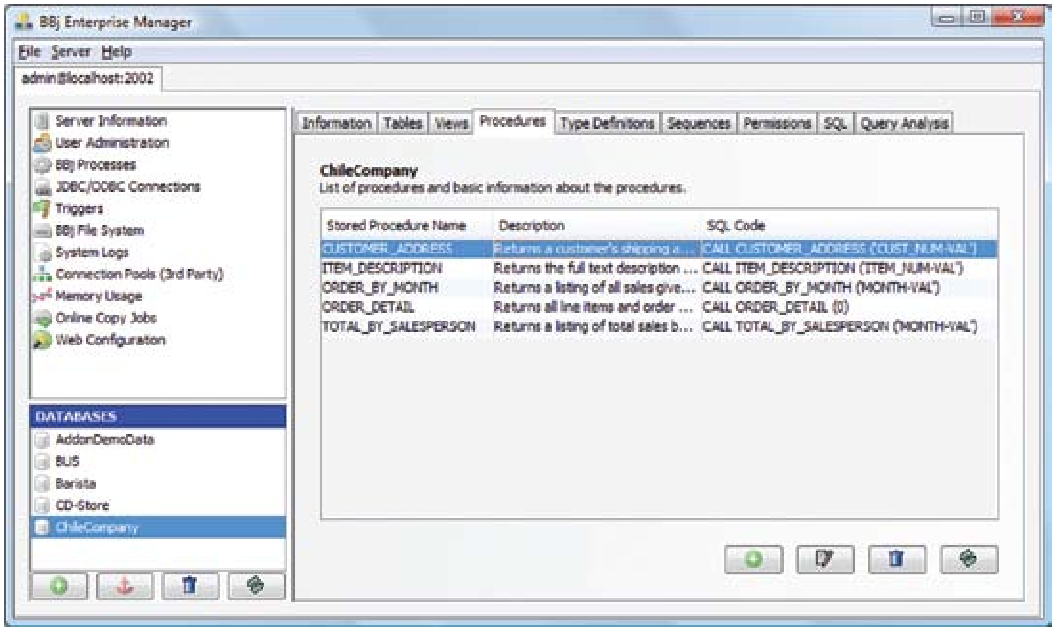
Figure 3. Accessing a data file's string template



OPTIONAL: The source code template for the procedure will automatically generate code to create a sample result set for your procedure. If you would like it to use a particular string template to define the columns in the result set, please specify that now or leave blank to use a default template.

FIRST_NAME:C(20),LAST_NAME:C(30),COMPANY:C(30),BILL_ADDR1:C(30),BILL

Figure 4. Specifying the string template for the result set



ChileCompany
List of procedures and basic information about the procedures.

Stored Procedure Name	Description	SQL Code
CUSTOMER_ADDRESS	Returns a customer's shipping a...	CALL CUSTOMER_ADDRESS ('CUST_NUM-VAL')
ITEM_DESCRIPTION	Returns the full text description ...	CALL ITEM_DESCRIPTION ('ITEM_NUM-VAL')
ORDER_BY_MONTH	Returns a listing of all sales give...	CALL ORDER_BY_MONTH ('MONTH-VAL')
ORDER_DETAIL	Returns all line items and order ...	CALL ORDER_DETAIL (0)
TOTAL_BY_SALESPERSON	Returns a listing of total sales b...	CALL TOTAL_BY_SALESPERSON ('MONTH-VAL')

Figure 5. The newly-defined SPROC in the list for the ChileCompany

to modify the EM-generated template program to return the right data – the actual data from the ChileCompany CUSTOMER data file.

Modifying the SPROC Program

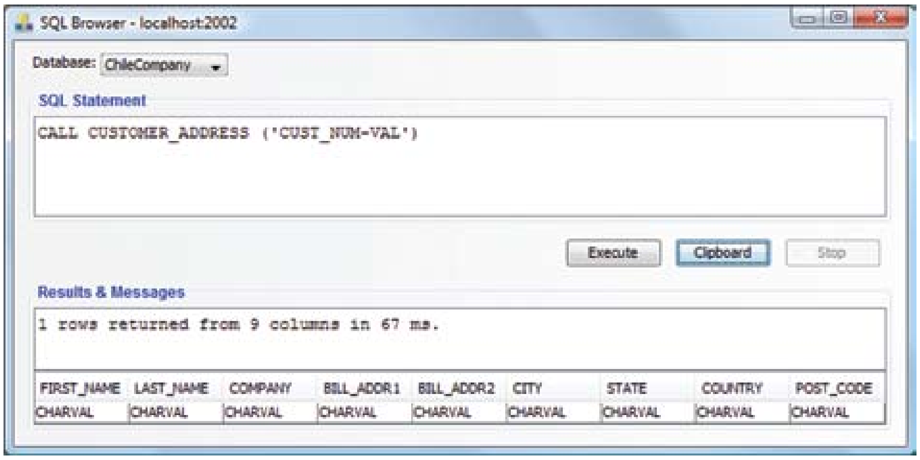
Open up the **`CUSTOMER_ADDRESS.prc`** file in the BASIS IDE and notice that the EM-generated program file already does most of the work. For example, the program takes care of getting the customer number that the client specified and loads it into a variable named CUST_NUM$. It also creates a memory record set, fills it with the appropriate address data, and sends it back to the client. The only change needed, is to remove the section of code that fills the record set with sample data and replace it with code that fills it with the real data. That job is easy too, as EM already wrote most of that code based on the supplied string template. In fact, just make a couple of small changes and the SPROC will retrieve data from the designated CUSTOMER data file.

Begin by removing the section of code that fills the record set with sample data, as shown in **Figure 7**.

Notice that the code filled every field in the record set with the string CHARVAL. This should look familiar as it is the same return result set for every column that occurred on the first test of the SPROC from the EM.

The next step is to enable the template code that fills the record set with the correct data from the CUSTOMER data file. After uncommenting the pre-generated code block, it looks like **Figure 8**.

The code is very close to what is ultimately desired, but it still needs a few changes. For starters, specify the path to the actual CUSTOMER data file instead of the MY_FILE placeholder in the code. Next, modify the rec$ template to match the full string template for the CUSTOMER file (right now it matches the return result set template which is just a subset of the full record template). Lastly, modify the READ RECORD routine. The routine is designed to read all the way through a data file, returning every record. The goal with the customer address is different, >>



```
CALL CUSTOMER_ADDRESS ('CUST_NUM-VAL')
```

```
1 rows returned from 9 columns in 67 ms.
```

FIRST_NAME	LAST_NAME	COMPANY	BILL_ADDR1	BILL_ADDR2	CITY	STATE	COUNTRY	POST_CODE
CHARVAL	CHARVAL	CHARVAL	CHARVAL	CHARVAL	CHARVAL	CHARVAL	CHARVAL	CHARVAL

Figure 6. Testing the newly created SPROC

```
REM Populate the record set with some sample data.
REM If you uncomment the code above, you will want to
REM make sure to remove this code.
data! = rs!.getEmptyRecordData()
data!.setFieldValue("FIRST_NAME", "CHARVAL")
data!.setFieldValue("LAST_NAME", "CHARVAL")
data!.setFieldValue("COMPANY", "CHARVAL")
data!.setFieldValue("BILL_ADDR1", "CHARVAL")
data!.setFieldValue("BILL_ADDR2", "CHARVAL")
data!.setFieldValue("CITY", "CHARVAL")
data!.setFieldValue("STATE", "CHARVAL")
data!.setFieldValue("COUNTRY", "CHARVAL")
data!.setFieldValue("POST_CODE", "CHARVAL")
rs!.insert(data!)
```

Figure 7. Filling the record set with sample data

```
REM ******************************************************
REM This portion is commented out.  Make necessary changes
REM to this section to make your procedure read records from
REM a BBj data file and populate the in-memory recordset
REM with the results.
REM
REM Create a template to hold the read record data.
DIM rec$:"FIRST_NAME:C(20),LAST_NAME:C(30),COMPANY:C(30),BILL_ADDR
dataFile$ = "../data/MY_FILE"
    chan = unt
    open(chan) dataFile$
    while 1
        read record(chan,err=*BREAK) rec$
        data! = rs!.getEmptyRecordData()
        data!.setFieldValue("FIRST_NAME", rec.FIRST_NAME$)
        data!.setFieldValue("LAST_NAME", rec.LAST_NAME$)
        data!.setFieldValue("COMPANY", rec.COMPANY$)
        data!.setFieldValue("BILL_ADDR1", rec.BILL_ADDR1$)
        data!.setFieldValue("BILL_ADDR2", rec.BILL_ADDR2$)
        data!.setFieldValue("CITY", rec.CITY$)
        data!.setFieldValue("STATE", rec.STATE$)
        data!.setFieldValue("COUNTRY", rec.COUNTRY$)
        data!.setFieldValue("POST_CODE", rec.POST_CODE$)
        rs!.insert(data!)
    wend
```

Figure 8. The template code that fills the record set

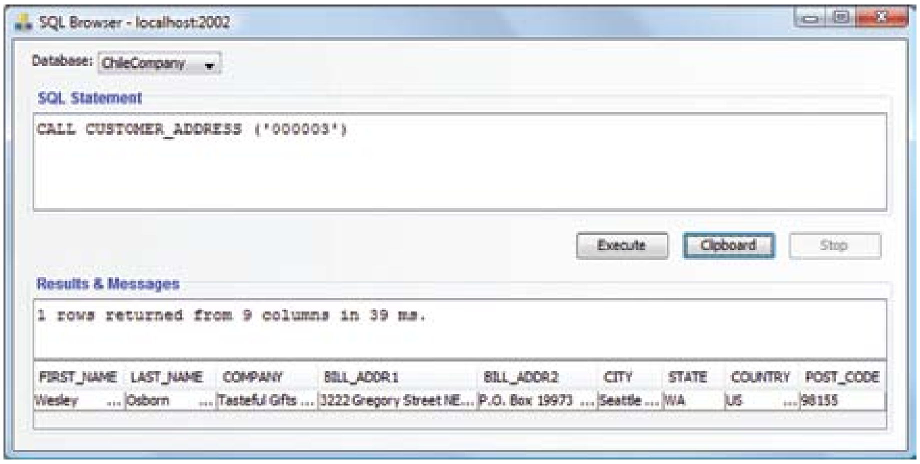
though, because it just needs to return a single address from the customer file. Since the SPROC has an input parameter for CUST_NUM, the caller can specify which customer address to return. Therefore, remove the WHILE/WEND loop and add a KEY= mode to the READ RECORD statement so that the code only reads the record the client requests. The resulting code appears in **Figure 9**.

```
REM *****************************************************
REM This portion is commented out.  Make necessary changes
REM to this section to make your procedure read records from
REM a BBj data file and populate the in-memory recordset
REM with the results.
REM Create a template to hold the read record data.
DIM rec$:"CUST_NUM:C(6),FIRST_NAME:C(20),LAST_NAME:C(30),COMPANY
dataFile$ = "../data/CUSTOMER"
    chan = unt
    open(chan) dataFile$
    read record(chan,key=CUST_NUM$,err=*BREAK) rec$
    data! = rs!.getEmptyRecordData()
    data!.setFieldValue("FIRST_NAME", rec.FIRST_NAME$)
    data!.setFieldValue("LAST_NAME", rec.LAST_NAME$)
    data!.setFieldValue("COMPANY", rec.COMPANY$)
    data!.setFieldValue("BILL_ADDR1", rec.BILL_ADDR1$)
    data!.setFieldValue("BILL_ADDR2", rec.BILL_ADDR2$)
    data!.setFieldValue("CITY", rec.CITY$)
    data!.setFieldValue("STATE", rec.STATE$)
    data!.setFieldValue("COUNTRY", rec.COUNTRY$)
    data!.setFieldValue("POST_CODE", rec.POST_CODE$)
    rs!.insert(data!)

REM Tell the stored procedure to return the result set.
sp!.setRecordSet(rs!)
```

Figure 9. The final READ RECORD code in the SPROC

Testing the Completed SPROC

With the final code changes in place, it is time to test the SPROC once again. This time supply an actual customer number for the input parameter and the SPROC will return live data from the ChileCompany Customer data file. In EM's SQL tab, run the query once again but this time specify a six-digit customer number as the input parameter. The SPROC will successfully retrieve the customer's shipping address from the database and display it as a result set in a grid as shown in **Figure 10.**



```
CALL CUSTOMER_ADDRESS ('000003')
```

```
1 rows returned from 9 columns in 39 ms.
```

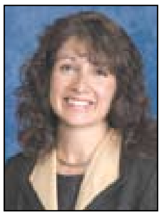
FIRST_NAME	LAST_NAME	COMPANY	BILL_ADDR1	BILL_ADDR2	CITY	STATE	COUNTRY	POST_CODE
Wesley ...	Osborn ...	Tasteful Gifts ...	3222 Gregory Street NE...	P.O. Box 19973 ...	Seattle ...	WA	US ...	98155

Figure 10. Testing the SPROC in EM

With a fully functional SPROC, a plethora of applications can access the customer data via SQL. The iReport Designer, covered in more detail in this issue's *Recipes for Successful Report Writing* on page 6, is a perfect example. Report authors traditionally use SQL queries to retrieve data for their reports, and a call to the new SPROC is just the ticket. After all, the report data can be the result of any query – whether it is accessing a table, view, or SPROC. In **Figure 11**, iReport's Services section of the IDE demonstrates this by dutifully offering a list of available tables, views, and SPROCs for each database connection.

The new SPROC, CUSTOMER_ADDRESS, shows up in the list of available procedures for the ChileCompany Database. Expanding the SPROC node reveals the list of its parameters. For each database connection, iReport talks to the back-end database and gathers metadata regarding the tables, views, and stored procedures. iReport takes this a step further by gathering details such as the description for each SPROC and the data types and comments for every parameter. The screenshot in **Figure 11** demonstrates this by displaying the input parameter and description for this new SPROC in its properties window. Notice that the Notes field in the properties window reflects the SPROC description that was filled in when originally creating the SPROC (see **Figure 1**). By including helpful comments and descriptions for SPROCs and their parameters, it makes it much easier for the end users to create a report because the SPROCs are self-explanatory. When report authors load up a tool like iReport, they have access to the right data without needing intimate knowledge of the database. All they have to do is peruse a list of available SPROCs and read the descriptions of each to find out which one suits their purpose. Supplied with the built-in descriptions and comments for each parameter, they are well on their way to creating a report in a flash. >>

Mining the Diamond Work Group



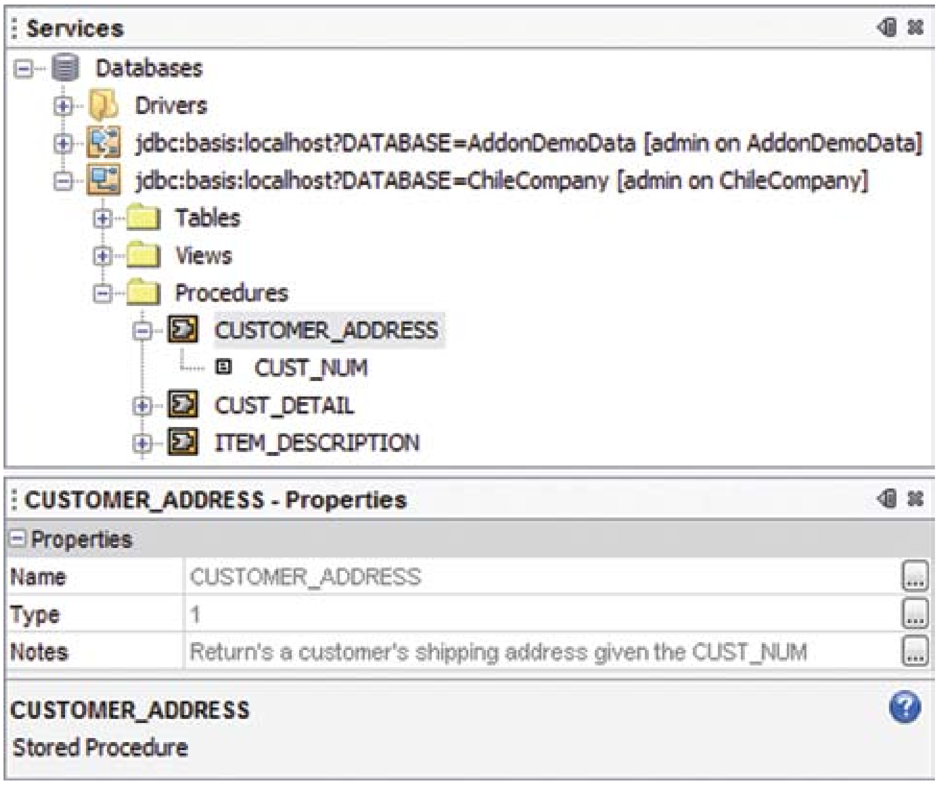
By Gale Robledo
Account Manager

In October 2008, Nico Spence and I attended the **Diamond Work Group Conference** in Alameda, CA. Established by a committee of Diamond 725 users, this conference is an annual opportunity to share and gain knowledge of how other users are managing information, enhancing their enterprise software, and employing new technology.

Nico presented many benefits of the current BBj® technology and how to take full advantage of its features and functions to improve overall productivity and performance. His excitement about BASIS tools clearly captured the attention of those in attendance. They quickly recognized the power they have with BBj and the enhancements they could make to improve their application.

Diamond 725 services the healthcare industry for health claims processing. Today, many Diamond users run PRO/5® and/or BBj. BASIS technology not only provides the tools they need to improve their application but also to stay competitive in the marketplace.

User groups like the Diamond Work Group Conference provide the opportunity to gather with peers and technology providers to stay abreast of the ever-changing IT world. BASIS welcomes the opportunity to partner with new resellers and users, and participate in strategy sessions and technology reviews. BASIS also provides professional services to help jump start BBj projects. Please call BASIS Sales if you require assistance in any of these areas. ■



Services
- Databases
  - Drivers
  - jdbc:basis:localhost?DATABASE=AddonDemoData [admin on AddonDemoData]
  - jdbc:basis:localhost?DATABASE=ChileCompany [admin on ChileCompany]
    - Tables
    - Views
    - Procedures
      - CUSTOMER_ADDRESS
        - CUST_NUM
      - CUST_DETAIL
      - ITEM_DESCRIPTION

CUSTOMER_ADDRESS - Properties

Properties

Name	CUSTOMER_ADDRESS
Type	1
Notes	Return's a customer's shipping address given the CUST_NUM

CUSTOMER_ADDRESS
Stored Procedure

Figure 11. iReport shows the available SPROCs, parameters, and descriptions

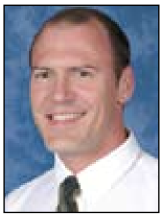
Summary

Anyone who followed these steps has completed their first stored procedure and it is ready for action. Not bad for a few minutes worth of work! Armed with Enterprise Manager's new SPROC template generation and the fact that BBj stored procedures can leverage existing legacy code and utilize standard BBx syntax like READ RECORDs, creating SPROCs are now a snap. Legacy programmers are empowered to expose their databases and business logic to other SQL applications safely, securely, and with the blazing speed of native access. ■

Debugging SPROCs and Triggers

Stored procedures (SPROCs) and triggers have proven themselves over the last few years as they provide developers with critical new capabilities and expand data access and connectivity options. After the developer has written, debugged, and deployed them, they run like a well-oiled machine and rarely require any maintenance or updates. Running silently in the background, stored procedures and triggers do their job behind the scenes, unbeknown to most users. Getting them to that point, however, may take a little more time and expertise compared to a typical 'foreground' application. The reason for this extra effort is because these types of programs must be able to run in the background without any user interaction and be prepared to handle any error condition or abnormality that occurs during its lifetime.

Therefore, the writing and debugging phase of trigger and stored procedure development is typically more challenging than writing a foreground program, however a new feature in BBj® 9.0 provides welcome relief!



By Nick Decker
Engineering Supervisor

Debugging Triggers and Stored Procedures – an Exercise in Frustration

If you have ever tried to debug a SPROC or a trigger, you are familiar with that initial feeling of helplessness that often overcomes new developers. After all, you only have control of the client application that causes the trigger or SPROC program to execute; you do not have direct control over that program's execution. Furthermore, attempting to execute the trigger or SPROC directly always results in failure. This is because they rely on constructs such as the BBjTriggerData or BBjStoredProcedureData objects in order to execute successfully – and these will not be available. So you are in a difficult position as you cannot run the trigger program directly. Instead, you must let the BBj file system execute it in response to designated file access. However, when BBj launches the program in the background, you have no way of interacting with it in order to debug problems.

A Clumsy Workaround

Before BBj 9.0, a programmer's only recourse was to log everything out to a file. Because it was not possible to dot-step through the trigger program interactively, developers would sprinkle the code with dozens of lines of debug code to print out program and variable state to a log, almost like debugging in a non-interpretive language. This workaround succeeded, but was clumsy and cluttered the program with large quantities of debug code. To add insult to injury, the developer would then remove the debug code, only to realize that it was necessary once again during a future debug session and required reinsertion.

The Elegant Solution

BASIS realized that this process was not optimal and sought a solution. The most straightforward solution – letting the developers interact with a background process – seemed unlikely at first because that has never been possible in the past. The benefits were so tempting, though, that the clever coders behind BBj figured out a way to do just that!

Taking the Plunge – Debugging a Trigger or SPROC

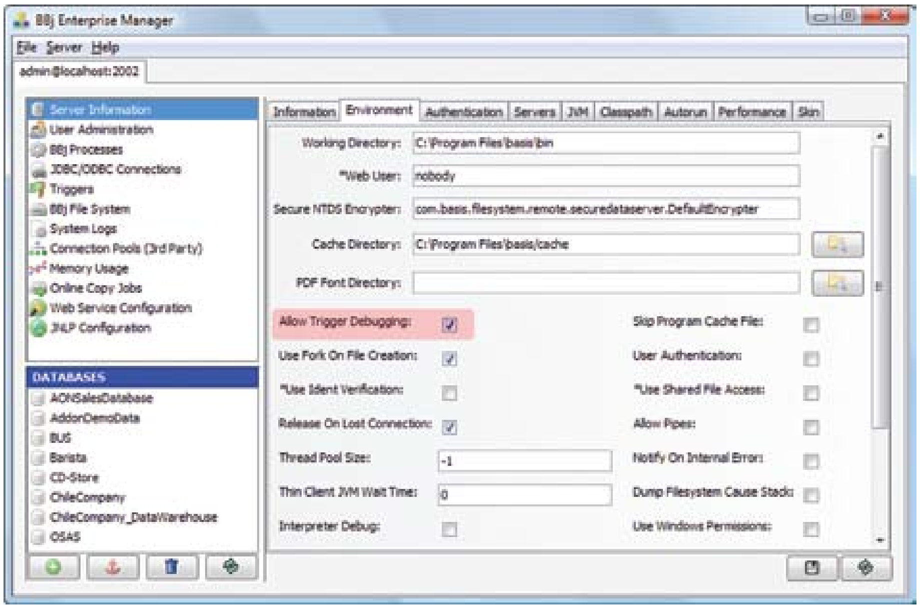
Before jumping into a trigger or SPROC debug session, there are a couple of prerequisites. First, you must turn on the ability to debug triggers and SPROCs in Enterprise Manager (EM). Interactive SPROC and trigger debugging is aimed primarily at developers, and because the option is a global setting for BBjServices, it is 'off' by default. **Figure 1** shows EM's Server Environment tab with the 'Allow Trigger Debugging' checkbox selected.

In order to debug a SPROC, set the similar option for the desired database. **Figure 2** shows the database's miscellaneous tab with the 'SPROC Debugging' checkbox selected. > >

After enabling interactive debugging, ensure that BBj Services is running in an environment that supports a graphical SysConsole. Finally, if BBj is running on MS Windows, BBj Services must not run as a service as it will not have sufficient access to the desktop to display a SysConsole.

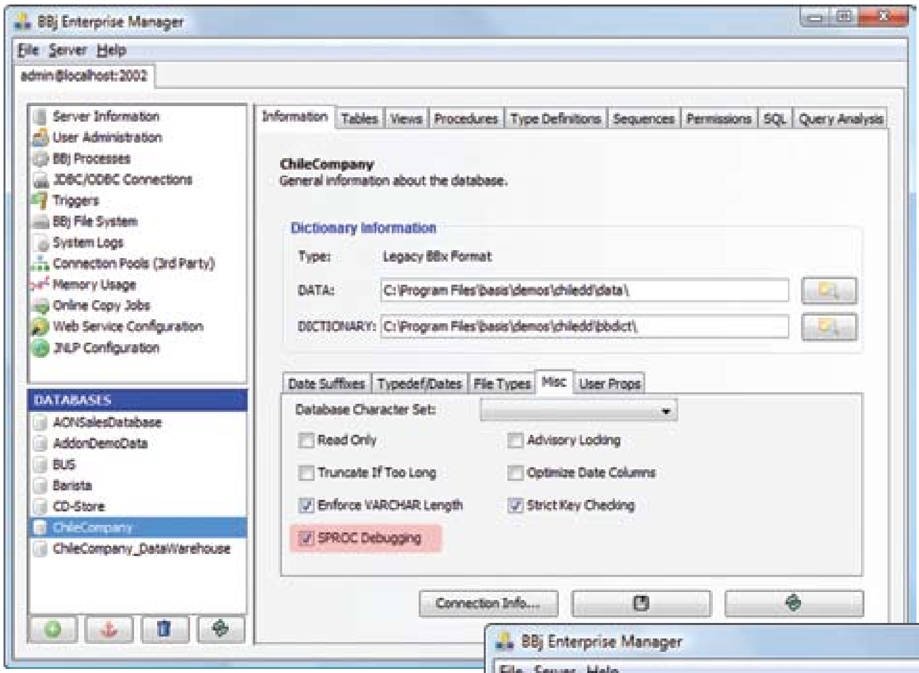
Once you have met all of the prerequisites, initiating a debug session is as simple as modifying the program code for the trigger or SPROC, usually by adding an ESCAPE. Then, force the program execution by calling the SPROC or accessing the file with the trigger.

Our example uses a trigger for the ChileCompany Customer data file. Begin by mounting the ChileCompany data directory, then adding the Customer file as shown in **Figure 3**. **> >**



Working Directory: C:\Program Files\basis\bin
*Web User: nobody
Secure NTDS Encrypter: com.basis.filesystem.remote.securedataserver.DefaultEncrypter
Cache Directory: C:\Program Files\basis\cache
PDF Font Directory:
Allow Trigger Debugging: ✓
Use Fork On File Creation: ✓
*Use Ident Verification:
Release On Lost Connection: ✓
Thread Pool Size: -1
Thin Client JVM Wait Time: 0
Interpreter Debug:
Skip Program Cache File:
User Authentication:
*Use Shared File Access:
Allow Pipes:
Notify On Internal Error:
Dump Filesystem Cause Stack:
Use Windows Permissions:

Figure 1. EM setting that allows trigger debugging

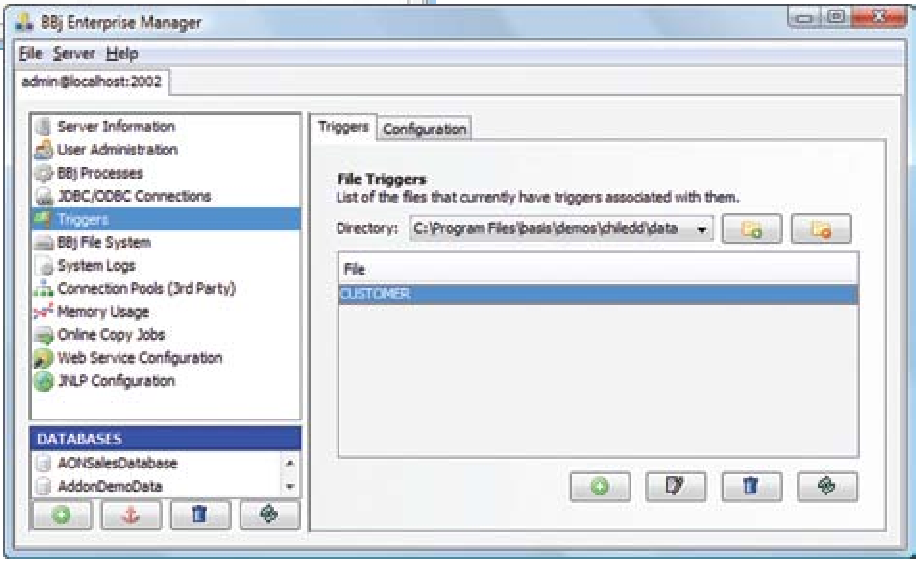


ChileCompany
General information about the database.

Dictionary Information
Type: Legacy BBx Format
DATA: C:\Program Files\basis\demos\chiledd\data\
DICTIONARY: C:\Program Files\basis\demos\chiledd\bbdict\

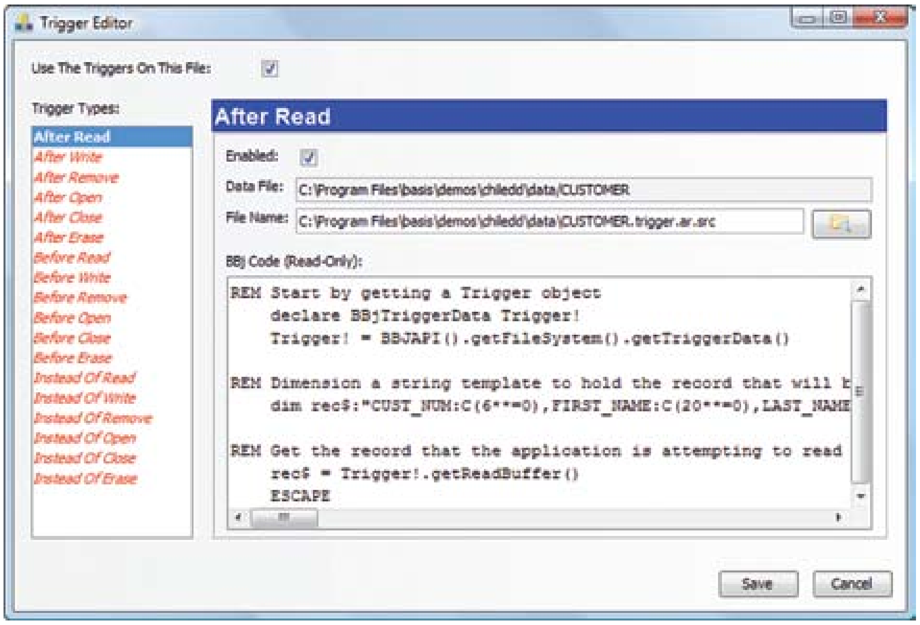
Database Character Set:
Read Only
Advisory Locking
Truncate If Too Long
Optimize Date Columns
✓ Enforce VARCHAR Length
✓ Strict Key Checking
✓ SPROC Debugging

Figure 2. EM setting that allows SPROC debugging



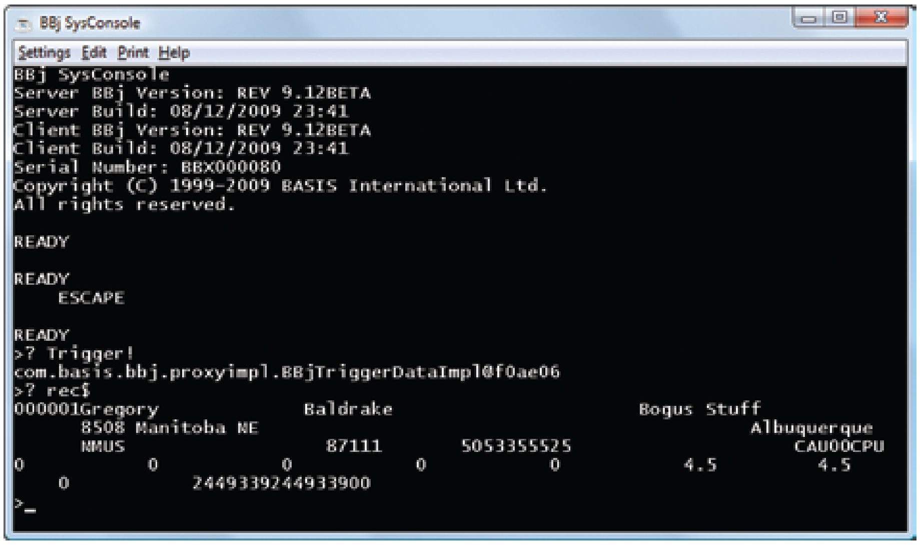
File Triggers
List of the files that currently have triggers associated with them.
Directory: C:\Program Files\basis\demos\chiledd\data
File
CUSTOMER

Figure 3. Mounting the data directory in Enterprise Manager



```
REM Start by getting a Trigger object
    declare BBjTriggerData Trigger!
    Trigger! = BBJAPI().getFileSystem().getTriggerData()

REM Dimension a string template to hold the record that will b
    dim rec$:"CUST_NUM:C(6**=0),FIRST_NAME:C(20**=0),LAST_NAME

REM Get the record that the application is attempting to read
    rec$ = Trigger!.getReadBuffer()
    ESCAPE
```

Figure 4. The After Read trigger definition



```
BBj SysConsole
Server BBj Version: REV 9.12BETA
Server Build: 08/12/2009 23:41
Client BBj Version: REV 9.12BETA
Client Build: 08/12/2009 23:41
Serial Number: BBX000080
Copyright (C) 1999-2009 BASIS International Ltd.
All rights reserved.

READY

READY
    ESCAPE

READY
>? Trigger!
com.basis.bbj.proxyimpl.BBjTriggerDataImpl@f0ae06
>? rec$
000001Gregory             Baldrake                      Bogus Stuff
      8508 Manitoba NE                                            Albuquerque
      NMUS                  87111       5053355525                    CAU00CPU
0           0           0           0           0           4.5         4.5
    0           2449339244933900
>_
```

Figure 5. Interactively debugging the trigger code

Next, select and enable the desired trigger. **Figure 4** shows that we have defined an 'After Read' trigger, meaning that the BBj trigger program executes after BBj reads the requested data from the disk.

The Trigger Editor in **Figure 4** also displays a read-only copy of the trigger program, allowing for easy identification and preview capabilities. Note the ESCAPE at the end of the code causes BBj to display an interactive SysConsole as soon as it executes that line. To test the trigger, simply perform an ordinary read on the customer data file. BBj will read the specified data as usual and then fire the After Read trigger. The trigger code executes until it hits the ESCAPE, as shown in **Figure 5**.

This also demonstrates that we are able to interact with the trigger program. After the trigger has paused execution following the ESCAPE, we are able to print out the Trigger! object and the contents of rec$, which is the trigger's read buffer (the data that was read from the disk and is destined for the client application). Because the session is completely interactive, it is possible for us to modify the contents of rec$ to return different information back to the original program or query that caused the trigger to fire.

After completing the debug session, type in **BYE** or **RELEASE** in the SysConsole to exit the trigger code and return the results to the client. Once the program runs to satisfaction, simply remove the ESCAPE and disable interactive debugging in Enterprise Manager to return everything to normal production mode.

Summary

In the past, determining what occurred in the inner workings of a SPROC or trigger program required a great deal of debug code and even more patience. BBj 9.0's new interactive debugging capability makes that a thing of the past as developers are free to debug their SPROCs and triggers in the same manner as all of their other application code.

The newfound ability to view and modify variable data, change program flow, and dot-step through the code ought to bring a sigh of relief to developers. We sure think so. ■

The Freedom to Grow®

AddonSoftware®

ACCOUNTING DISTRIBUTION MANUFACTURING

AddonSoftware has developed practical, affordable, feature-rich, and easy-to-use enterprise-wide software applications for businesses for more than 25 years.

Companies using AddonSoftware enjoy ***The Freedom to Grow®*** because of its lean efficiency, scalability, ease of customization, and ability to facilitate the preservation of existing customizations through the upgrade cycle.

- **Modern**
- **Productive**
- **Extensible**
- **Flexible**
- **Trusted**

Contact us to find out more.

1.800.370.9131 U.S. and Canada
+1.505.338.4188 International
info@addonsoftware.com
www.addonsoftware.com

A Friendlier 'AddonSoftware by Barista'

Developers at BASIS have been busy remodeling AddonSoftware®'s Accounting and Distribution bundles from their foundations up. The primary tool they used for this project was the Barista® Application Framework along with the proven functionality of BBj®. By any standard, this was a lofty undertaking resulting in many exciting enhancements to the product. Step into this article for a virtual tour of the most notable highlights of these user-friendly features.

Easy-to-use Menu System

Barista's easy-to-use menu system is revealed in the AddonSoftware menu on the left side of the multiple document interface (MDI). The top pane (see **Figure 1**) shows the individual modules and when a module is selected, the programs available in that module display in the bottom pane. The menu items themselves appear in a very consistent and standardized manner so that it is easy to find a task even when moving between modules. Additionally, the tasks appear so that the primary, daily processing tasks are always at the top and display in an expanded view to eliminate extraneous mouse clicks and movements.

Accommodating Multiple Screens

While the MDI delivers multiple windows in one screen, the AddonSoftware implementation allows you to launch a program in SDI mode. Users launch SDI programs by simply pressing the [Shift] key before making a menu selection. In stand-alone mode, the form has its own menu and status bars and is not constrained by the MDI. This flexibility is convenient for work stations with multiple screens. For more information about how to add this to your application, see *MDI MODES* on page 18 of this issue. When windows are launched within the MDI and sized and positioned by the user, AddonSoftware will remember the size and position when the window is launched again.

By Paul Yeomans
Vertical Market Account Manager

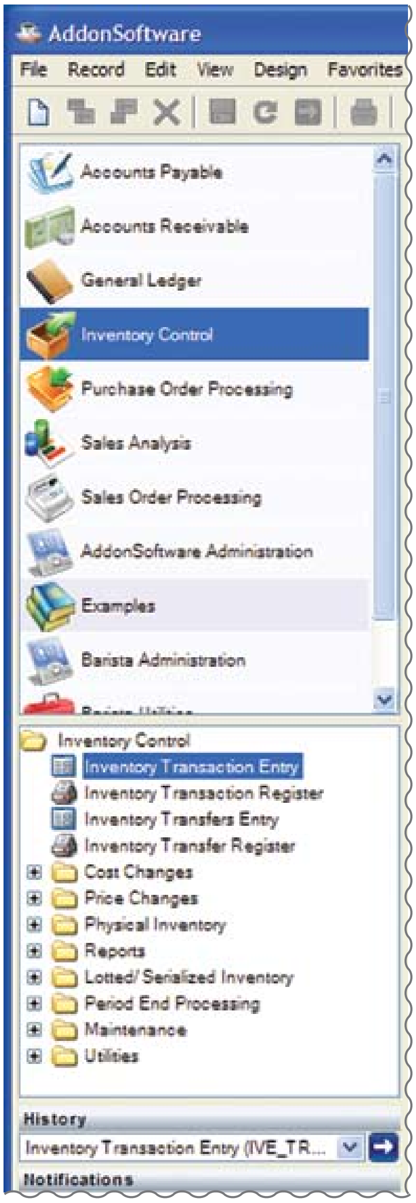


Figure 1. Sample menu with expanded view of tasks

Favorites

AddonSoftware also takes advantage of the Barista "Favorites" menu. Simply right-clicking on any menu item adds that item to users' favorites – a great place to save their frequently run processes across multiple modules and eliminate the time and effort clicking around searching for a commonly run task. Selecting Favorites from the top menu bar reveals the selected tasks, as shown in **Figure 2**.

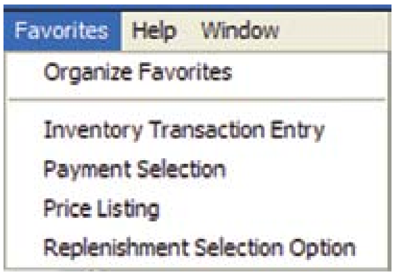


Figure 2. Favorites task shortcut

Navigation Choices

As a matter of preference, users can perform basic navigation with either the mouse or the keyboard. Users control the navigation buttons using the mouse while the mouse or keyboard gives them control of the movement around the form. For quick reference, each command on the menu bar drop downs includes the keyboard shortcut for that function. To return to a process that was run two or three steps ago, select that task from the History drop down (see **Figure 3**) to return to the task quickly. This is just another way that Barista provides shortcuts for ease of access and use.

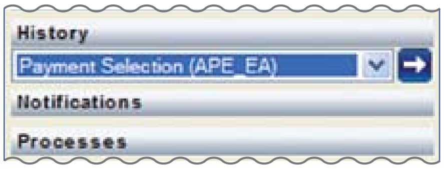


Figure 3. History process shortcut

Built-in Security and Auditing

Another versatile and powerful feature that the new AddonSoftware makes use of is Barista's built-in security. The role-based security structure secures access to data and business practices by defining user roles with permissions, access, or restrictions, and then assigning those roles to specific users. For example, tasks may or may not display in a given menu depending on the permissions of that user. At a more granular level, security enabled on specific fields in a table prevents users from editing or viewing the restricted data. Additionally, the audit logging feature records when and who made changes to the database. >>

Context Sensitive Help

Best of all, 'help' is just a click away. Right-clicking on an item from any menu displays the help topic for that function. Alternatively, users can launch Help by either pressing the [F1] key or by clicking on the Help drop down at the top of the screen. Once inside the Help topic, users can easily navigate to other help topics by clicking the title in the topic listing. The Help system provides a very quick and direct way for users to get assistance on the specific task they are running. Alternatively, the Help menu at the top menu bar displays the traditional Web-based Application Help.

Report Selections are Remembered

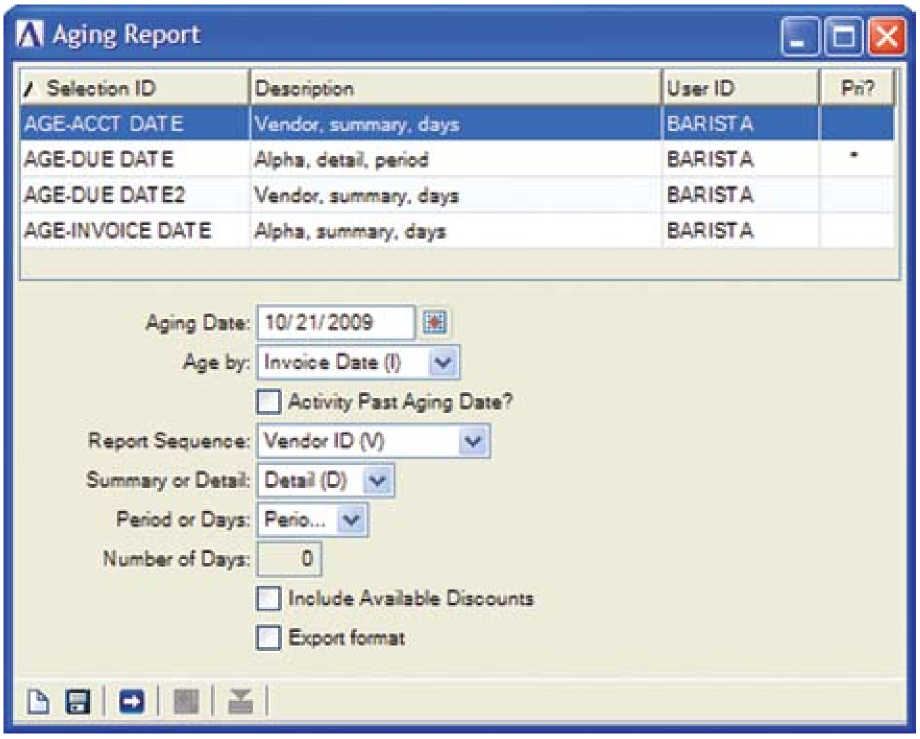
Running reports in AddonSoftware involves selecting multiple filters to access and display the data you require. To preserve the selections for frequently run reports, simply select the Save icon on the selection window and name the report (see **Figure 4**). Saved report selections can be made private for that specific user or created and used throughout the organization.

Document Management

AddonSoftware also incorporates the Barista document management system. This empowers users to search and quickly retrieve any document or report previously generated and saved in the document archive. Once retrieved, they can open it to print a traditional hard copy, send it electronically via e-mail or fax to a customer or send it within the organization for collaborative work.

Expresso Search

The new "Expresso" Search function greatly expands the traditional inquiry and drilldown capabilities. While the Barista inquiry system provides search, sort, and filtering capabilities at table and field level, Expresso Search provides users with predefined or easily customizable drilldown relationships based on table definitions and their relationships. Create custom search definitions and drilldown into master records with hyperlinks.



Aging Report

/ Selection ID	Description	User ID	Pri?
AGE-ACCT DATE	Vendor, summary, days	BARISTA	
AGE-DUE DATE	Alpha, detail, period	BARISTA	*
AGE-DUE DATE2	Vendor, summary, days	BARISTA	
AGE-INVOICE DATE	Alpha, summary, days	BARISTA	

Aging Date: 10/21/2009
Age by: Invoice Date (I)
☐ Activity Past Aging Date?
Report Sequence: Vendor ID (V)
Summary or Detail: Detail (D)
Period or Days: Perio...
Number of Days: 0
☐ Include Available Discounts
☐ Export format

Figure 4. Saved report selection defaults

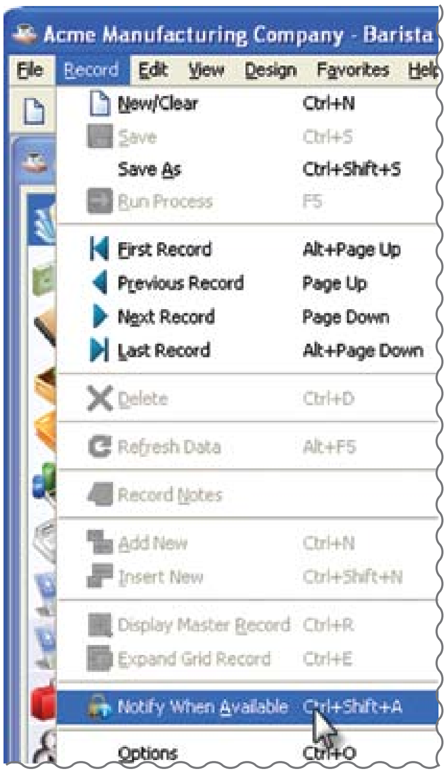


Figure 5. Notify when record is available

Traffic Cop for File Access

If a user attempts to edit a master file currently being edited by someone else, they are notified that the record is not accessible. For efficiency, that user may select an option that notifies them when the record is available for them to edit (see **Figure 5**). No wasted time checking back periodically for record access. The system will notify you automatically.

Summary

AddonSoftware by Barista has the core accounting and distribution functions one expects to find in a modern, scalable business solution. Many of these user experience features are easy to demonstrate and deliver value by increasing the productivity of your end users. ■

See a demonstration of many of these new 'AddonSoftware by Barista' features at tinyurl.com/yjxb9t3

On the Road in the Great Northeast

The opportunity to visit end user customers is so enlightening. A first-hand look at hardware, application, and database infrastructure is key to helping us understand our clients' real-world needs and helping them overcome any obstacles to achieving those goals.

Nico Spence and I had two such opportunities with Tufts Health Plan and Whelen Engineering.

TUFTS Health Plan

Our trip started in Watertown, Massachusetts at **Tufts Health Plan**, located in a beautiful business district near MIT. We spent the day with Jim Binney, MP Systems Support Manager, and his staff, learning about their infrastructure, 2009 hardware upgrade plans, and desire to migrate to BBj®. It was an eye-opening experience looking at the vast topology around BBx® that interfaces to many other applications and databases.

Jim presented a well-planned timeline for upgrading his hardware to the Hewlett Packard Itanium platform with over 500 users and making the move to BBj. Nico reviewed the benefits of BBj and gave Jim and his staff ideas on the potential enhancement opportunities available to them. Our goal was to give Tufts advice on their migration steps to insure a successful upgrade and to educate them on the tools they will need to have in order to make their development tasks easier.

The first migration step we recommended was to move to a BBj database, allowing them to use many of the BBj features alongside their PRO/5® applications, like enhanced file types such as J-Keyed, X-Keyed, V-Keyed, and ESQL. This would also provide them access to stored procedures, triggers, live backup, online copy job, and much more.

A feature that piqued their interest is triggers. Triggers will allow them to replace time-consuming batch jobs to update other databases in real time to keep their data synchronized. This is just one example of the many ideas that we explored together. Since our visit, we continue to guide Tufts along their upgrade path and make ourselves available to them as needed. Tufts' goal is to completely migrate their applications to BBj and use its powerful functionality to make enhancements and create new applications.

WHELEN

After visiting Tufts, we embarked on a road trip through scenic Connecticut to **Whelen Engineering**. At one point Nico was certain I was lost; it was the beautiful countryside that threw him off. But in no time I found Whelen Engineering, located in picturesque rural Chester.

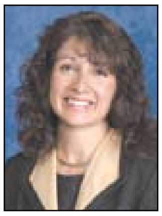
Whelen manufactures sirens, alarms, beacons, and many other related products and relies on PRO/5 to keep their enterprise software running for several hundred users. Tom Englart and Nora Lund, Senior Software Analysts, gave us an overview of their infrastructure and the applications they currently maintain. Although PRO/5 has always been a solid and reliable product, it became apparent that there were many ways BBj could make their job easier, give users better access to data, and increase productivity.

We spent the afternoon giving Tom and Nora a technology overview of all areas of the language and offered advice on how to improve their database, administration duties, and development tools. It was a productive day sharing ways they can take advantage of BBj features to run their department more efficiently.

We try very hard to communicate to the world about all of the exciting new BBj features and functions through our communiqués, videos, and tutorials, yet it is so important to share face-to-face time to understand the customer needs. It is also a great opportunity to address those needs with the cutting edge technology that BASIS offers and continues to enhance.

Our goal is to reach out to our customers that have the aspiration to take their enterprise software to the next level. Software developers and end users alike can benefit from the many venues we offer for product knowledge. Stay up-to-date with our Advantage magazine articles, upcoming Java Break with BASIS Webinar series, ongoing training classes, and technical conferences. In addition, visit our Web site often – it has an abundance of information at your finger tips.

We look forward to the opportunity for face-to-face time with you and to have technology and strategy discussions. The BASIS Sales Team is ready to hit the road and bring guidance and direction to you. Contact your BASIS Account Manager to set up an onsite visit. ■



By Gale Robledo
Account Manager

BARISTA

GUI Application Framework

Brews

GUI Applications

- Creates new applications from scratch
- Updates existing GUI applications
- Renovates legacy CUI applications

With

Framework Built on BASIS Technology

- Delivers cross-platform solutions
- Offers choice of multi-tiered architecture
- Significantly reduces program code

Data Dictionary-based Structure

- Maintains standard table and key definitions
- Supports entry validation, business rules, and interface formatting

Form Manager and Designer

- Displays current status of all tables defined in Barista
- Creates a consistent look and feel across applications

Callpoint System

- Provides custom interaction with Barista-based forms
- Seamlessly calls customized programs via runtime engine

Help System

- Context sensitive and on-line help

Inquiry System

- Displays data in any Barista-defined table
- Allows sorting and filtering for all columns

DocOut and iReport

- Renders customizable reports within the MDI

Legacy Report and Update Compatibility

- Reuses existing update and report programs
- Allows creation of forms that capture user input

Menu System

- Inherits role-based security
- Launches any CUI or GUI application

Expresso Search

- Leveraging relationships of column and table information to generate drill downs

MDI/SDI Deployment

- Run in a multiple document interface or as a standalone application

Delivers

Sarbanes-Oxley Compliance Capability

- Audit logging - records who made database changes
- Role-based security structure - secures access to data and business processes

Contemporary-looking Applications

- Handles the vast majority of the standard application programming

Drilldown Functionality

- Access your data with user configurable defaults

Integration with BASIS and Third Party RDBMS

- Choose the back-end database that works best for you

Multifold Productivity Gains

- Saves time and money

Document Output System

- Efficiently deliver professional looking reports
- Access normalized or non-normalized data
- View, fax, e-mail and save multiple output report formats
- Archiving capabilities

Incremental Modernization Capability

- Create new GUI modules that seamlessly integrate with exisiting applications in MDI or SDI mode

BASIS International®

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

Tech Resource Zone

Getting to Know More About the BBj

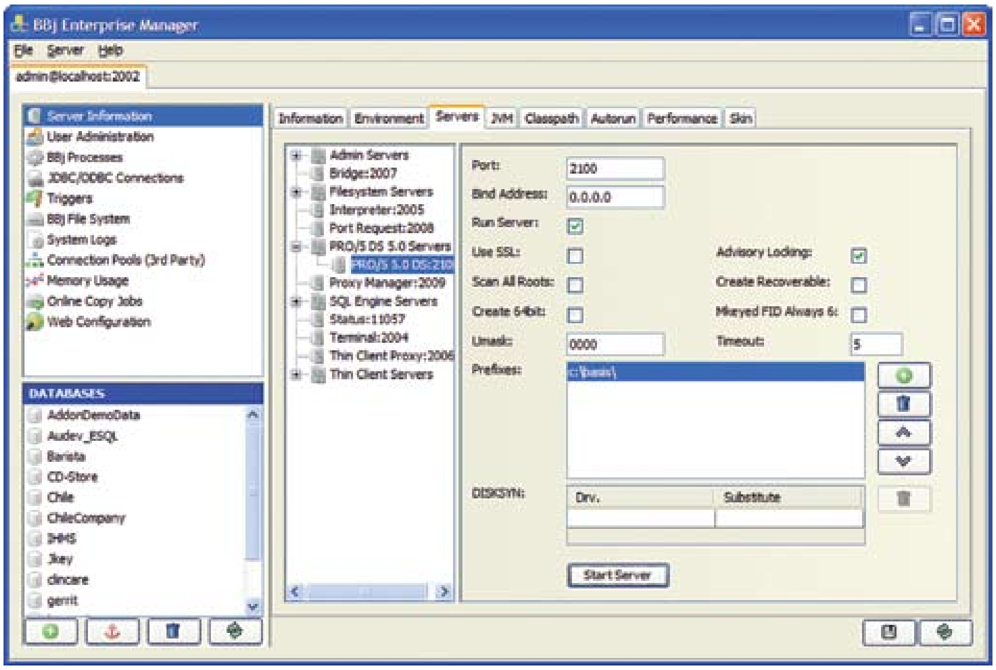
Even though the BBj® PRO/5 5.0 Data Server® (BBj P5 DS) has been part of the BBj product since nearly the beginning, we still often receive questions in Tech Support about how to use the BBj PRO/5 DS. It is a database management component that enables PRO/5® or Visual PRO/5® clients to access the BBj-based BASIS DBMS and all its features. It's a very viable component to the BBj solution that you may not have considered. Read on to discover more about how this might fit into your configuration.

Q. ***What is the difference between the BBj PRO/5 DS and the PRO/5 5.0 Data Server? Why would I use one over the other?***

A. The main difference is whose control the data is under, PRO/5 or BBj. The PRO/5 (P5) Data Server is a standalone product that controls PRO/5 data access that serves data to (V)PRO/5 or BBj clients, while the BBj P5 DS is one of the BBj Services DBMS components that permits access to the BASIS DBMS from (V)PRO/5 clients. One performance advantage to using the BBj Services DBMS with BBj P5 DS for file access from PRO/5 clients is that less stress is placed on the operating system resources on large user networks compared to the PRO/5 5.0 Data Server. This is especially true when the server is a Windows machine. It also enables Triggers, Stored Procedures, and access to additional BASIS file types. For a more in depth look at the two products and determining which one is best for your configuration, read *Lock, Lock, Who's Got The Lock? Is it Visual PRO/5, PRO/5, PRO/5 Data Server, or BBj?*

Q. ***How do I install the BBj P5 DS?***

A. The BBj P5 DS installs automatically when installing the standard default components of BBj. You can configure it in Enterprise Manager under the Servers Tab as published in the online documentation for Enterprise Manager - Server Information.



Port: 2100
Bind Address: 0.0.0.0
Run Server: ✓
Use SSL: ☐
Scan All Roots: ☐
Create 64bit: ☐
Umask: 0000
Prefixes: c:\basis\
DISKSYN: Drv. Substitute
Advisory Locking: ✓
Create Recoverable: ☐
Mkeyed FID Always 6: ☐
Timeout: 5

Figure 1. Configuring the BBj P5 DS in Enterprise Manager

By Janet Smith
Technical Support Supervisor

PRO/5 5.0 Data Server

Q. ***Where is the config.bbx file located that the BBj P5 DS uses?***

A. Actually, the BBj P5 DS does not use a config.bbx file. Instead, set the configuration settings and information pertaining to the BBj P5 DS in Enterprise Manager as shown in **Figure 1**.

Q. ***How do I access files via the BBj P5 DS?***

A. To enable data server access in the interpreter, set the setopts bit byte 4 to `$20$: setopts $00000020$`

The code syntax for accessing files is the same for both BBj P5 DS and standalone PRO/5 DS except for the port specified: `open(1)"/<server,port=2100>\path\filename"`.

By default, the standalone P5 DS uses port 1100 and BBj P5 DS uses port 2100, but you can change the BBj P5 DS port in the Enterprise Manager. The same network troubleshooting tips regarding hostname resolution and .rhosts user authentication apply to both products. For more information, read knowledge base articles KB 964 *TCP/IP Hostname Resolution and the BASIS License Manager (BLM)* and KB589 *TCB(10)=-10061 or RUSEROK Failure in Data Server Log.*

Q. ***Why am I getting an !ERROR=69 "User count or validation error" when I try to access the BBj P5 DS?***

A. This error occurs when you are attempting to access the BBj P5 DS from a BBj client. The BBj P5 DS is intended for use by (V)PRO/5 clients only as explained in KB1095 *ERROR=69 Returned When BBj PRO5 Data Server Accessed Incorrectly*. It can also occur if you do not have an Enterprise License feature in your BBx product license as explained in the Advantage article *BBj: One License, One Install*. ■

ASK Cost Saving Products for BBx Applications

✓ How much are you spending annually on pre-printed forms?

✓ Are you still storing and retrieving client files manually?

✓ How much time and money is wasted by stuffing and mailing your invoices?

Despite your company's best efforts to transition from hard-copy to electronic files, most operations are still relying on costly paper, requiring a labor intensive manual storage and retrieval process. Implementing a paperless reporting system is an effective way to reduce these costs and improve employee productivity, in turn having a positive impact on your bottom line.

ASK provides software that will allow you to send any selected report or document as an email attachment.

Select any recipient from a predefined email address field. Attachments can be in a MS Word, MS Excel, RTF, PDF or html format. All emailed reports fit on standard 8 1/2" x 11" sheet paper.

Benefits Include:

- Eliminate costly pre-printed documents
- Reduce paper document storage with cold storage document archiving
- Reduce time customer service inquiries with immediate access to client documents
- Improve cash flow by eliminating delays in billing
- Delivery, storage and retrieval all within your BBx application
- Reduce paper and storage costs through a centralized electronic database

ASK technologies, inc.
www.asktech.com

7 Bala Avenue, Ste. 201 Bala Cynwyd, PA 19004
Tel +1 (610) 617-0300 Fax +1 (610) 617-0307
www.asktech.com

Switched on Design

BASIS lights up its look and feel

In the spirit of TechCon2009, BASIS designed the *BASIS International Advantage* magazine anew.

Last year, BASIS introduced the first European issue of the *BASIS International Advantage* magazine in a new layout and modified design. The format is now slightly narrower than the 8.5" North American/US paper size standards, which allows BASIS Europe to mail the US English edition in their standard international-sized envelopes. BASIS also introduced new visual elements to create a fresh, new viewing experience.

Good design integrates strong elements of color, line, light, space, shape, texture, and form. It is equally as important as content; one supports the other to communicate a message. Titles and headers are critical to effective user-friendly communication in printed collateral layout and Web design. It is human nature to want to know something quickly. Titles and headers help to grasp the major points and ideas of an article, and tell the reader what to expect from the remaining content. Color aids in identifying similar items.

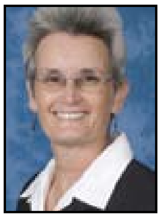
Since "a picture is worth a thousand words," graphic elements and screen shots are excellent resources for enhancing attractive and interesting design and for succinct communication. Images often communicate a message faster, more clearly, and more emphatically than text. Icons, as small graphic elements representing an application, file, or hardware resource, are also very useful images in communicating a message without the need for text. When readers see a familiar icon, such as a floppy disk 💾 that represents [Save File] or a blue underlined word that means a hyperlink to other pages, they immediately know what the item represents and what to do.

How BASIS Turned on the Light

The BASIS Advantage magazine outside cover now sports a new masthead and graphically calls out our feature articles. Inside the magazine, we introduce articles with large visual representations and align BASIS product component color tabs Language/Interpreter to the top of the page. Articles are further color-coded to the product component with the I nitial capital letter of the lead paragraph, **Subheads**, the continued arrows **>>**, the square symbol ■ indicating the article end, the outline of the code samples, the 'For More Information' box, and the page numbers 7.

Finally, by introducing a 2 and 3-column format with a sleek sans serif font, we added visual flexibility and more white space for a lighter, more accessible, easier-to-read publication.

BASIS always strives to enhance their presentations to the reader, so the reader can make the information actionable in the most efficient way. The *BASIS International Advantage* magazine continues to present the very latest in BASIS products, features, sales strategy, support, partnership, news, events, applications, utilities, and much more. We hope that you enjoy the new look and feel. ■



By Patricia Catlett
Art Director

Have You Missed an Issue?

Find all of these articles and more on our Web site www.basis.com/advantage

Partnership

2008	#1	Starwood's Web Service Shines Brighter
2007	#1	RCG Uses Marketing Skills to Achieve OSAS "Top Dog" Status
		Marex Returns Home to BASIS – A Personal Journey
2006	#1	Borealis Press Chooses Mac Platform
		BASIS Tours Europe With New Partnership Program
2005	#2	OSI Solidifies OSAS Investment with BBj
		QA Memos Web Service
2005	#1	BBx Generations – Our Progression in Time
		MARK SYSTEMS – A Revenue Stream Runs Through It
		Enhancing the Bouquet with BBj – KMK Software AG
2004	#1	Fit For the Future? – Midata Service GmbH
		BBj Sparks High Productivity at Heilind Electronics
2003	Q4	b'Using b-commerce
		BASIS Introduces Online Tutorials
		Maharam: BASIS Customer Profile
		Technical Resources on the BASIS Web Site
	Q2	SAM Plan – Managing Your Software Assets

Language / Interpreter

2008	#1	Dishing up Forms
		T Minus - BASIS LaunchDock Blasts Off
		Blowing the Doors Wide Open With ClientsObjects
		Quick and Easy Solutions With Java Libraries - Part 1
		Quick and Easy Solutions With Java Libraries - Part 2
		BASIS Adds Mapping Functionality to BBjRecordSets
		More on Client File Access
		Inner Types Exposed
		Socket to Me
		Deliver a Package Tracking System
		Real-time Web Reports
2007	#1	Spin, Slide, or View Your Data
		If You Have Choices, We Have Choosers
		BBj Spell Checker is all the Buzz
		A Tour of the BBjCharts API
		Leap from 'Cut and Paste' to 'Drag and Drop'
		Desktop Data Delivered
		Catching the XML Wave
		A Deeper Voyage Into the BBjCharts API
		TRZ: Jars, Jars, and More Jars
2006	#1	A Primer for Using BBj Custom Objects
		Confessions of a Language Polygamist
		Applying Custom Objects to Existing Code
		The Scoop on 64-Bit Computing
		Visual PRO/5 6.0 with an XP or Vista L& F
		Type Checking With "bbjcpl"
2005	#2	Grid to Order
		Safe and Retrievable Intellectual Property
2005	#1	Input Validation: Veto Power
		Connection Pooling – Part 1
		Connection Pooling – Part 2
2004	#1	The Secrets of Secure Communication
		Why Use the BBjRecordSet?
2003	Q4	BBj Databound Control
		BBj Form Printing
		GUI Date Input With the BBj INPUTD Control
		Using the BBjRecordSet
	Q2	New Language Features Give Programmers More Choices
		Writing a Web Service in BBj
	Q1	BBj and Web Services
2002	Q2	Running BBj Code From Within Java; The JavaBBjBridge
	Q1	Performance Enhancements in BBj 2.0 - Print Batching
2001	Q2	From BBx to BBj
		The GUI Enhancements of BBj

Database Management System

2008	#1	Databound Grid - Data Abstraction Extended
		Take the Plunge Into SQL-Transactions
		All Aboard the REGEXP
		Data Warehousing II
2007	#1	Unleashing the Power of SPROCs Without SQL
		Solving the Data Warehousing Dilemma
2006	#1	Using Triggers to Maintain Database Integrity
		Using Stored Procedures to Add Business Logic to the Database
		ESQL Files: Constraining Your Data to Guarantee Integrity
2005	#2	PDF Now Also Means Perfectly Displayed Forms
		OpenOffice.org – a Sweet New Suite
		VKEYED Files – New Features for a New File Type
2005	#1	The BASIS DBMS – New 5.0 Features
2004	#1	Secure Your Future, Encrypt Your Data
		The Great Connection
2003	Q2	BASIS International Offers a Choice of Databases
		Capitalizing on BASIS's Open Architecture
2002	Q1	Implementing Journaled Files and Transaction Tracking in BBj 2.0
		Route Data to Oracle & More with Plug-Ins
2001	Q2	Next Generation File Format: XKEYED Files

Development Tools

2008	#1	From the BASIS DD to a Barista App in a Flash
		From the BASIS DD to a Barista App in a Flash Part 2
		Goodbye Event Loop, Hello Process Events
2007	#1	Watch the Form Gen Wizard Trans"form" Data
2006	#1	AppBuilder: The BASIS IDE Gets RAD GUI Development Integration
		Freedom of Choice: Using Object Code Completion in the IDE
		BASIS Puts Triggers to Work
2005	#2	Tuning the Performance Analyzer
		The Sate of the IDE
2005	#1	FormBuilder: BASIS IDE's Better Cross-Platform Resource Builder
		TRZ: Logs, Logs, and More Logs
2004	#1	Inside BASIS: Software Product Version Control
2003	Q4	Managing Databases/Data Dictionaries with the BASIS IDE
		Viewing Data Files in the BASIS IDE
	Q2	Speeding Up the Development Process – The BASIS IDE
2002	Q3	Putting The BBj 2.0 Online Documentation to Work For You
		Using The Netbeans-Based BBjIDE to Develop/Debug (Part One)

Applications

2008	#1	Freshly Brewed - Barista Caffeinates Addon Accounting Applications!
		BBj Thrives on the Web

System Administration

2008	#1	The Extreme Enterprise Manager Makeover
2006	#1	Solving the Locked Record 'Whodunit'
		TRZ: The FAQ's About 64-Bit BASIS Products
		Gain the Advantage – Use the Advantage Resource
2005	#2	Choices, Choices, Choices
		The Response File Responds Silently
		Safe and Retrievable Intellectual Property
		TRZ: Lock, Lock, Who's Got the Lock?
2005	#1	New Install Puts You in the Driver's Seat
		Taking Out The Garbage
2004	#1	Give Your Software a New Lease on Life
		Tech Resource Zone: Licenses and BLMs
		Without a Trace: The Silent Installation
2003	Q2	Enterprise Manager – Administration Made Easier
		Web Deployment Options With BASIS
2002	Q2	BBj Memory and Process Management
		BBj Thin Client Performance Tuning
2001	Q2	BBj: One License, One Install

The Freedom to Grow® Your Partnership

AddonSoftware®

Our Partner Program – Delivering Building Blocks for Your Vertical Solutions

For over 25 years, AddonSoftware has developed practical, affordable, feature-rich, easy-to-use, enterprise-wide, and platform independent software applications for businesses across a variety of industries.

The new AddonSoftware Partner Program combines the best elements of the open source community development model with a commercial partnership. This innovative program rewards our partners for ongoing AddonSoftware-directed contributions to the product. We have leveraged the collective creative energy and expertise of our partners to continuously advance the application. Using AddonSoftware's core business applications as building blocks, our partners can remain focused on developing their own customized vertical solutions.

- **Join**
- **Build**
- **Share**
- **Gain**
- **Focus**

Sound interesting?
Join the AddonSoftware community today to make the most of your expertise while maximizing your revenue opportunities.

Contact us to find out more.

1.800.370.9131 U.S. and Canada
+1.505.338.4188 International
info@addonsoftware.com
www.addonsoftware.com