Partnership
Language/Interpreter
DBMS
Development Tools
System Administration
Applications

Freshly Brewed – *Barista Caffeinates Addon Accounting Applications!*

By Chris Hawkins

By now, just about everyone has heard about BASIS' exciting new rapid application development framework – Barista® – as well as the recent release of AddonSoftware™ 8.0, the first major application developed in the Barista Application Framework.

AddonSoftware has a 25-year history as a cost-effective, modular, robust, and fully-integrated enterprise resource planning solution. A strictly character-based application for most of that time, AddonSoftware eventually began the transition to a graphical user interface to take advantage of technological advancements and keep abreast of competitors.

AddonSoftware 7 gave users the opportunity to run in either graphical or character modes. While it provided the more modern look and feel that many users sought, Version 7 was complicated to develop, maintain, support, and especially customize, particularly in terms of the user interface.

With the advent of Barista, AddonSoftware decided to abandon the character mode and "go GUI" only. This was not a trivial undertaking, but considering that it meant remaining in a Business BASIC environment and preserving a large amount of legacy code, it was a desirable option compared to others.

This article reviews the process of creating AddonSoftware 8 and shares some ideas about how to brew a legacy application into an exciting new creation!

Application Analysis

Developing AddonSoftware within the Barista Application Framework offered substantial productivity gains over other alternatives, but the team still had some unique challenges to face. In analyzing the legacy versions, the developers had to determine the best way to take full advantage of the advancements offered by Barista and BBj®, but also preserve tried and true legacy "back-end" code where desirable, and modernize outdated data structures – all while providing an upgrade path for those using the older versions.

The legacy AddonSoftware code fell into roughly five categories: user interface, reports, updates, utilities, and public programs. Barista could handle the most complex of these – the user interface – and would also provide additional functionality in terms of inquiry, drilldowns, and document manipulation. That left the developers' with the manual tasks of porting desired back-end code into a format compatible with Barista and normalizing and converting data files.

At the Front End

Developing the User Interface in Barista

The user interface tends to be the most time-consuming component of any application to develop and maintain. These 'new' GUI user interface design skills often have to be provided by additional hires to the development team or require a substantial divergence of valuable resources from the already expensive resource pool. Using Barista to develop the user interface for AddonSoftware 8 not only enabled developers to create forms faster – several times faster in many cases – but also supplied a consistent look and feel throughout the application without the developers need to acquire a new skill set. In addition, developers were able to take advantage of Barista's inquiry and drilldown capabilities, all without writing so much as a line of code.

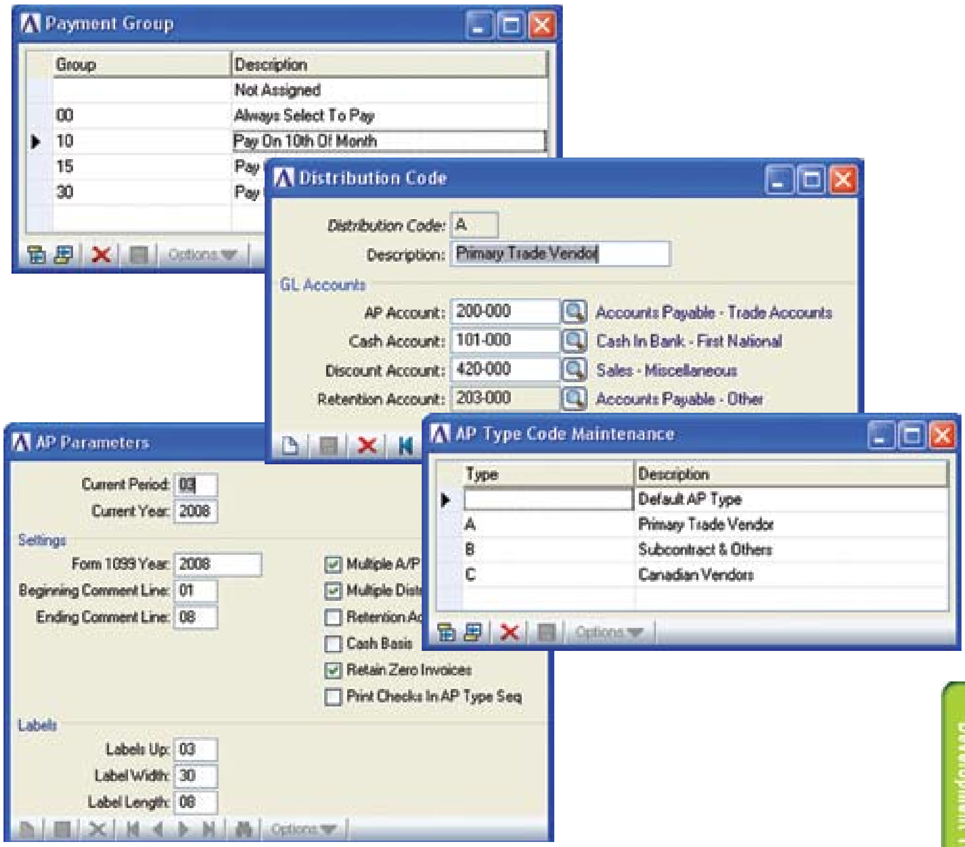
Compared to working in other graphical development environments, Barista's "bottom-up" approach seemed a bit foreign at first. But developers soon discovered the power behind the Barista design. The key to development in this new framework is to define the database elements, and then define tables using those elements. Given a table definition, Barista builds a fully functional form. Really, it can be just that simple, and just that fast.

Chris Hawkins
Software Engineer

continued...

AddonSoftware developers did not have to port any of the legacy user interface programs to Version 8; they developed all of the forms using the Barista framework. As a result, they significantly reduced the amount of code because Barista uses the same handful of programs to run every form. Once the core set of Barista programs is cached, forms run much faster. Also, changes and enhancements to the Barista framework, such as adapting to the constantly evolving GUI standards, are available at once in all forms, and do not have to be propagated manually or by means of a specially written utility.

Forms are easy to alter in the Barista Form Designer. Many of the AddonSoftware forms underwent some amount of cosmetic modification to place fields on tabs, insert group headings, add additional display fields, etc. Most of this work takes just a few minutes to complete. **Figure 1** shows some forms that developers created very quickly, requiring little or no custom code.



Payment Group

Group	Description
	Not Assigned
00	Always Select To Pay
10	Pay On 10th Of Month
15	Pay
30	Pay

Distribution Code

Distribution Code: A
Description: Primary Trade Vendor

GL Accounts

AP Account:	200-000	Accounts Payable - Trade Accounts
Cash Account:	101-000	Cash In Bank - First National
Discount Account:	420-000	Sales - Miscellaneous
Retention Account:	203-000	Accounts Payable - Other

AP Parameters

Current Period: 03
Current Year: 2008

Settings

Form 1099 Year: 2008
Beginning Comment Line: 01
Ending Comment Line: 08

[✓] Multiple A/P
[✓] Multiple Dist
[ ] Retention Ac
[ ] Cash Basis
[✓] Retain Zero Invoices
[ ] Print Checks In AP Type Seq

Labels

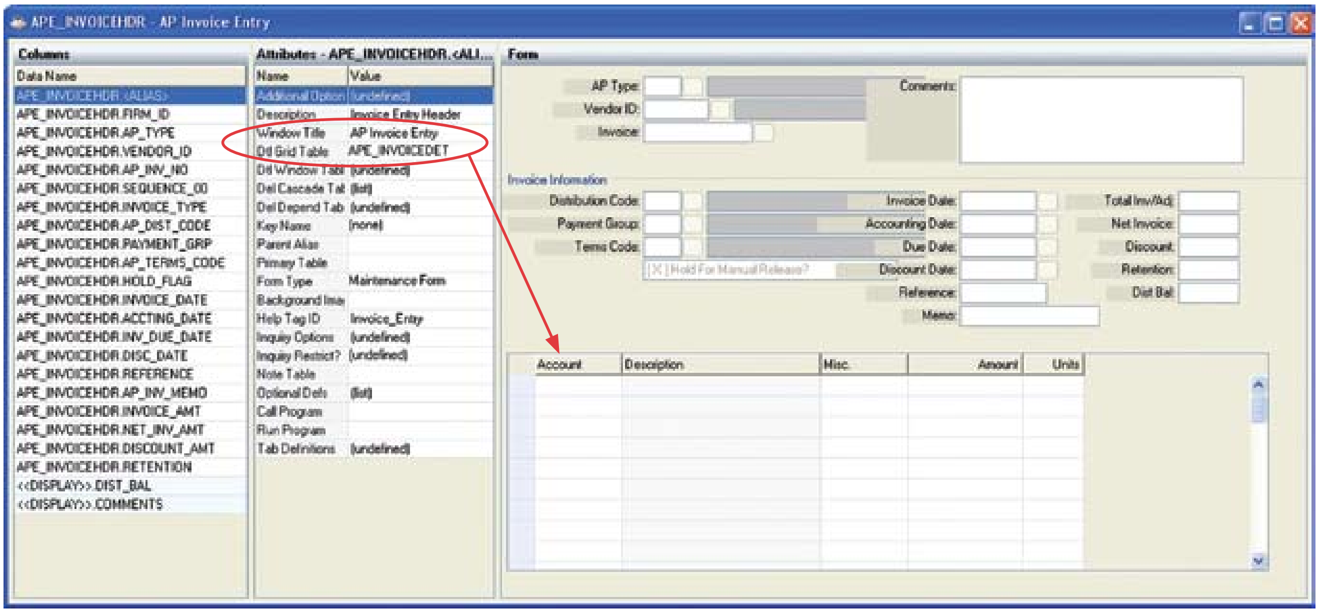
Labels Up: 03
Label Width: 30
Label Length: 08

AP Type Code Maintenance

Type	Description
	Default AP Type
A	Primary Trade Vendor
B	Subcontract & Others
C	Canadian Vendors

Figure 1. Barista enables rapid development of many of an application's less complex forms

Even more powerful is the ability to link forms based on their key structures. AddonSoftware makes use of several forms structured as "header/detail" or "one-to-many." With the header and detail tables already defined, the Form Designer makes it easy to specify that a particular header form contains a detail grid or detail window table. Specifying a detail grid provides a classic one-to-many user interface, with the header information at the top of the form, and the detail information in a grid below. **Figure 2** shows the table definitions and linking mechanism used for Accounts Payable Invoice Entry.



APE_INVOICEHDR - AP Invoice Entry

Columns

Data Name
APE_INVOICEHDR.<ALIAS>
APE_INVOICEHDR.FIRM_ID
APE_INVOICEHDR.AP_TYPE
APE_INVOICEHDR.VENDOR_ID
APE_INVOICEHDR.AP_INV_NO
APE_INVOICEHDR.SEQUENCE_00
APE_INVOICEHDR.INVOICE_TYPE
APE_INVOICEHDR.AP_DIST_CODE
APE_INVOICEHDR.PAYMENT_GRP
APE_INVOICEHDR.AP_TERMS_CODE
APE_INVOICEHDR.HOLD_FLAG
APE_INVOICEHDR.INVOICE_DATE
APE_INVOICEHDR.ACCTING_DATE
APE_INVOICEHDR.INV_DUE_DATE
APE_INVOICEHDR.DISC_DATE
APE_INVOICEHDR.REFERENCE
APE_INVOICEHDR.AP_INV_MEMO
APE_INVOICEHDR.INVOICE_AMT
APE_INVOICEHDR.NET_INV_AMT
APE_INVOICEHDR.DISCOUNT_AMT
APE_INVOICEHDR.RETENTION
<<DISPLAY>>.DIST_BAL
<<DISPLAY>>.COMMENTS

Attributes - APE_INVOICEHDR.<ALI...

Name	Value
Additional Option	(undefined)
Description	Invoice Entry Header
Window Title	AP Invoice Entry
Dtl Grid Table	APE_INVOICEDET
Dtl Window Tabl	(undefined)
Del Cascade Tal	(list)
Del Depend Tab	(undefined)
Key Name	(none)
Parent Alias	
Primary Table	
Form Type	Maintenance Form
Background Ima	
Help Tag ID	Invoice_Entry
Inquiry Options	(undefined)
Inquiry Restrict?	(undefined)
Note Table	
Optional Defs	(list)
Call Program	
Run Program	
Tab Definitions	(undefined)

Form

AP Type:
Vendor ID:
Invoice:
Comments:

Invoice Information

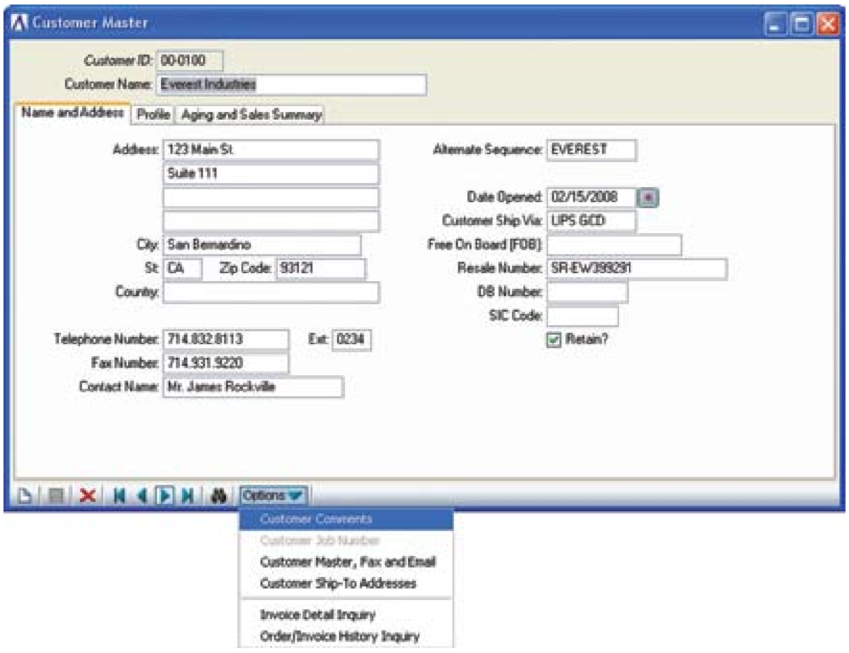
Distribution Code:
Payment Group:
Terms Code:
[X] Hold For Manual Release?
Invoice Date:
Accounting Date:
Due Date:
Discount Date:
Reference:
Memo:
Total Inv/Adj:
Net Invoice:
Discount:
Retention:
Dist Bal:

Account	Description	Misc.	Amount	Units

Figure 2. Header and detail tables easily linked into a one-to-many arrangement based on key structure

continued...

A detail window table is set up in much the same way, but when invoked from the main form through an options menu, it comes up as a separate modal form in Barista. **Figure 3** shows this structure in the Customer Maintenance form.



Customer ID: 00-0100
Customer Name: Everest Industries

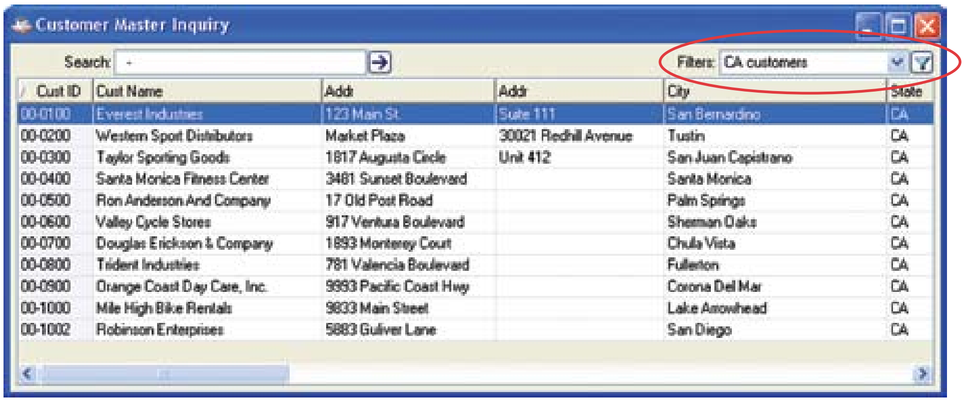
Address: 123 Main St
Suite 111
City: San Bernardino
St: CA Zip Code: 93121
Country:
Telephone Number: 714.832.8113 Ext: 0234
Fax Number: 714.931.9220
Contact Name: Mr. James Rockville

Alternate Sequence: EVEREST
Date Opened: 02/15/2008
Customer Ship Via: UPS GCD
Free On Board (FOB):
Resale Number: SR-EW399291
DB Number:
SIC Code:
[✓] Retain?

Figure 3. Customer Master form with an Options Menu containing several detail window tables

Barista's Inquiry Subsystem

One of the biggest time savers in Barista is the inquiry subsystem. Barista inquiries make use of the BASIS SQL engine and any table-bound form or field has access to the inquiry subsystem. Legacy AddonSoftware contained an extensive set of field and record lookups, all of them the result of code written by a developer. By using Barista to develop the new AddonSoftware forms, developers were able to leave behind all of that old code. Once they defined a table, the running form could query the underlying table(s) for either the form or fields on the form, sort and filter, and even save often-used filters. **Figure 4** illustrates a saved filter for the Customer form that shows only those customers with California addresses. Users can even copy and paste information in the inquiry grid to other applications, such as a word processor or spreadsheet.



Search: -
Filters: CA customers

Cust ID	Cust Name	Addr	Addr	City	State
00-0100	Everest Industries	123 Main St	Suite 111	San Bernardino	CA
00-0200	Western Sport Distributors	Market Plaza	30021 Redhill Avenue	Tustin	CA
00-0300	Taylor Sporting Goods	1817 Augusta Circle	Unit 412	San Juan Capistrano	CA
00-0400	Santa Monica Fitness Center	3481 Sunset Boulevard		Santa Monica	CA
00-0500	Ron Anderson And Company	17 Old Post Road		Palm Springs	CA
00-0600	Valley Cycle Stores	917 Ventura Boulevard		Sherman Oaks	CA
00-0700	Douglas Erickson & Company	1893 Monterey Court		Chula Vista	CA
00-0800	Trident Industries	781 Valencia Boulevard		Fullerton	CA
00-0900	Orange Coast Day Care, Inc.	9993 Pacific Coast Hwy		Corona Del Mar	CA
00-1000	Mile High Bike Rentals	9833 Main Street		Lake Arrowhead	CA
00-1002	Robinson Enterprises	5883 Gulliver Lane		San Diego	CA

Figure 4. Quickly get to data with stored filters

The inquiry system, in conjunction with Barista's Document Output utility, also eliminates developer code by automatically generating listings of the various application code tables. In legacy AddonSoftware, there were separate programs to print the content of these kinds of tables. None of that code is necessary when utilizing Barista.

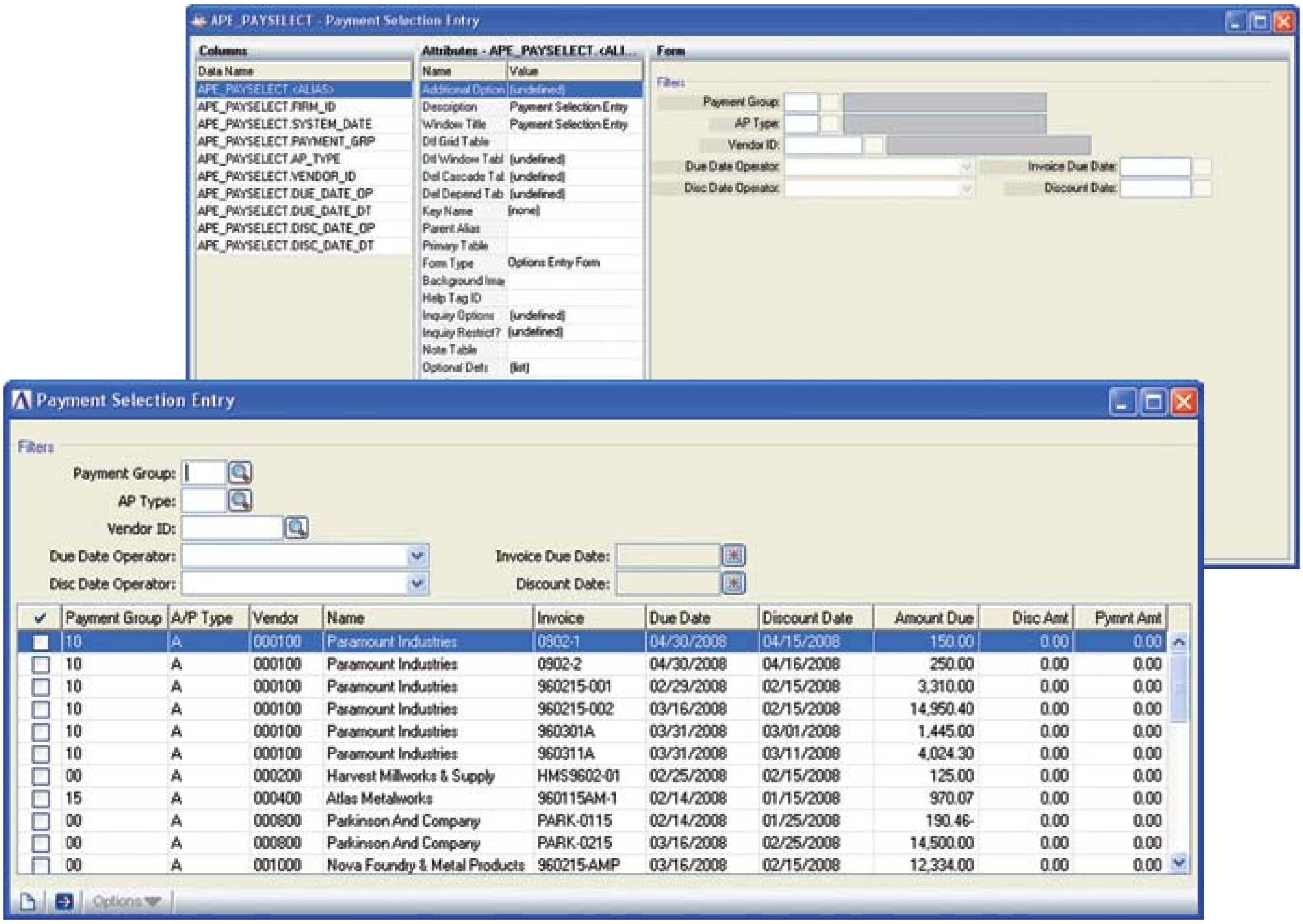
Barista Callpoints

Wondering how to implement old code? Whether the form is simple or complex, "you gotta get under the hood at some point," as they say. That is where Barista callpoints come into play. Barista callpoint code provides a gateway to supplement the Barista application with all that BBj and Java have to offer, thereby delivering flexibility to the framework's structure.

continued...

Barista callpoints are a series of event “hooks” associated with a Barista form, where a developer may want to place custom code. A few of the commonly used callpoints are “Before Form Show,” “After Display,” “Before Write,” and “After Validation.” The names themselves suggest their use. For example, inserting code in the “Before Write” callpoint of an invoice entry program would be a good way to make sure that amounts in the detail lines match the total invoice amount.

Like other applications, AddonSoftware contains many forms that are fairly simple and a few that are quite complex. Using Barista, the developers created many of the simpler forms in very little time, often with just a small amount of callpoint code for additional validation, parameter checking, etc. Some forms, however, needed functionality that is more complex. For example, the form in which the user selects invoices for payment (**Figure 5**) uses a combination of Barista-defined elements and a custom BBjGrid. Callpoint code handles creating and filling the grid, filtering the grid contents, setting callbacks and processing grid events, and writing the final grid results to a payment selection table. Callpoints provide the developer with all of the functionality of BBj, including BBj Custom Objects and Java classes. Consider incorporating other graphical tools into forms – a Java spinner, maybe – or launch the user’s e-mail client or browser. Barista callpoints provide the portal for all of the additional code.



Data Name
APE_PAYSELECT.<ALIAS>
APE_PAYSELECT.FIRM_ID
APE_PAYSELECT.SYSTEM_DATE
APE_PAYSELECT.PAYMENT_GRP
APE_PAYSELECT.AP_TYPE
APE_PAYSELECT.VENDOR_ID
APE_PAYSELECT.DUE_DATE_OP
APE_PAYSELECT.DUE_DATE_DT
APE_PAYSELECT.DISC_DATE_OP
APE_PAYSELECT.DISC_DATE_DT

Name	Value
Additional Option	(undefined)
Description	Payment Selection Entry
Window Title	Payment Selection Entry
Dtl Grid Table	
Dtl Window Tabl	(undefined)
Dtl Cascade Tal	(undefined)
Dtl Depend Tab	(undefined)
Key Name	(none)
Parent Alias	
Primary Table	
Form Type	Options Entry Form
Background Ima	
Help Tag ID	
Inquiry Options	(undefined)
Inquiry Restrict?	(undefined)
Note Table	
Optional Dets	(list)

Payment Group: |
AP Type:
Vendor ID:
Due Date Operator:
Disc Date Operator:
Invoice Due Date:
Discount Date:

✓	Payment Group	A/P Type	Vendor	Name	Invoice	Due Date	Discount Date	Amount Due	Disc Amt	Pymnt Amt
☐	10	A	000100	Paramount Industries	0902-1	04/30/2008	04/15/2008	150.00	0.00	0.00
☐	10	A	000100	Paramount Industries	0902-2	04/30/2008	04/16/2008	250.00	0.00	0.00
☐	10	A	000100	Paramount Industries	960215-001	02/29/2008	02/15/2008	3,310.00	0.00	0.00
☐	10	A	000100	Paramount Industries	960215-002	03/16/2008	02/15/2008	14,950.40	0.00	0.00
☐	10	A	000100	Paramount Industries	960301A	03/31/2008	03/01/2008	1,445.00	0.00	0.00
☐	10	A	000100	Paramount Industries	960311A	03/31/2008	03/11/2008	4,024.30	0.00	0.00
☐	00	A	000200	Harvest Millworks & Supply	HMS9602-01	02/25/2008	02/15/2008	125.00	0.00	0.00
☐	15	A	000400	Atlas Metalworks	960115AM-1	02/14/2008	01/15/2008	970.07	0.00	0.00
☐	00	A	000800	Parkinson And Company	PARK-0115	02/14/2008	01/25/2008	190.46-	0.00	0.00
☐	00	A	000800	Parkinson And Company	PARK-0215	03/16/2008	02/25/2008	14,500.00	0.00	0.00
☐	00	A	001000	Nova Foundry & Metal Products	960215-AMP	03/16/2008	02/15/2008	12,334.00	0.00	0.00

Figure 5. The grid shows on the running form, built and managed entirely with callpoint code

Going Totally Custom

Alternatively, developers can go completely outside the Barista form and callpoint structure to write their own custom BBj programs and then access them in Barista. These programs run seamlessly with the rest of the application inside the Barista MDI. The AddonSoftware Executive Summary is a case in point. Created with the BASIS IDE, it uses BBjTree, BBjGrid, and BBjChart controls to present graphical representations of various AddonSoftware data file compilations.

At the Back End

Codeport

The various back-end programs in legacy AddonSoftware were in need of some rejuvenating. Some of the changes were necessary and others, while not essential, transformed the code into a more modern, structured format, facilitating future maintenance tasks. The Codeport program is an AddonSoftware-specific utility that converts legacy AddonSoftware programs into source files compatible with Barista. Codeport takes advantage of the fact that this legacy code is highly standardized so it is possible to reliably locate and alter or replace code. What Codeport cannot do automatically, developers can do manually with the BASIS IDE.

Continue on page 44

Freshly Brewed – *Barista Caffeinates Addon Accounting Applications!*

...Continued from page 9

Codeport performs functions such as removing line numbers, formatting code with indents, removing BEGIN statements and IOLISTS, replacing calls to legacy public programs with their more intuitive Version 8 names, and replacing common functions and routines with Version 8 counterparts. The developer makes further improvements by replacing GOTO's with structured code, using symbolic labels (e.g., `err=*next`) wherever possible, replacing references to obsolete sort files with alternate keys on the source files, and replacing IOLIST variable names with version 8 string template names, etc.

Dataport
The legacy AddonSoftware data was deficient in two major areas. Firstly, it carried the majority of the dates in a packed format. Secondly, it contained several non-normalized tables (one table containing multiple record types/formats). In addition, the legacy data dictionary contained several field names that were duplicated or were not intuitive. The Dataport utility is another AddonSoftware-specific tool intended to address these issues. Dataport unpacks all dates into YYYYMMDD format, separates old combined city/state address lines, replaces non-normalized files with individual files containing a single consistent record type, and performs several other conversions on AddonSoftware data. Dataport also makes use of several plain text files to specify file and/or field name changes between the legacy version and Version 8.

Import from BASIS Data Dictionary
Although Codeport and Dataport are tools developed specifically for upgrading AddonSoftware, Barista also includes a non-AddonSoftware specific tool for creating a Barista database. Developers who use the BASIS Data Dictionary (DD) in their applications are one step ahead in converting their applications to Barista. The Import from BASIS DD program will bring the BASIS DD across to Barista and, thereafter, any changes to the Barista dictionary will automatically be applied to the BASIS DD.

The import utility uses a configurable settings file. It reads the BASIS DD and incorporates these settings, creating Barista elements and table definitions. Again, with table definitions in place, a functional set of forms can result. In no time, the foundation of the application takes shape, running in Barista. Read *From the BASIS DD to a Barista App in a Flash* on page 12 of this issue.

Additional Barista Features

Document Management
After running the legacy report programs through the Codeport process and completing the post-Codeport cleanup, AddonSoftware developers had a functional set of reports. While these reports were adequate, they were limited just to the selected output device. If output was directed to the screen, the report appeared in the standard print preview window outside of the MDI. The developers decided to make another pass through those reports and convert them to use Barista's Document Output and Management System.

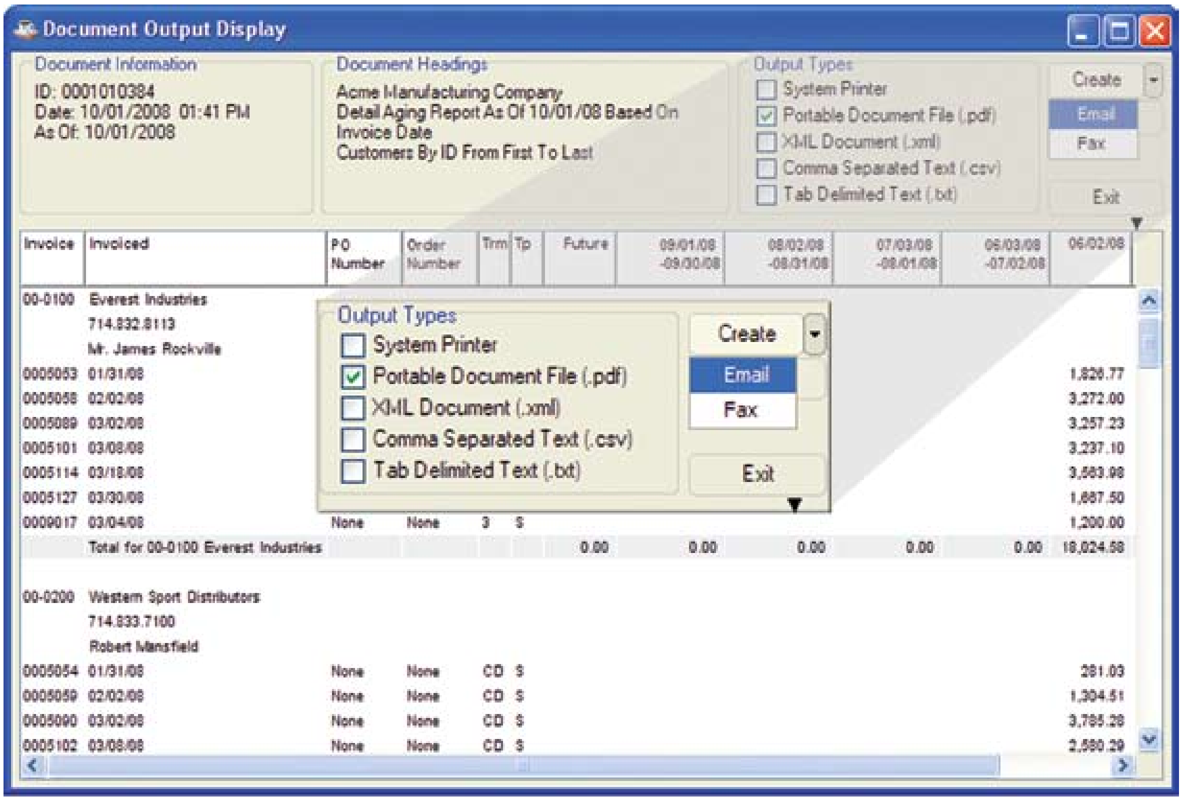
The Document Output and Management System, a.k.a. "DocOut," is another powerful component of Barista that offers several advantages over traditional printing. DocOut renders reports in a window within the MDI. The body of the report displays in a grid, the columns of which are user-adjustable. All reports rendered in DocOut, therefore, have a standardized look and feel. Once the document is created, users can opt to save the report in one or more output formats, as **Figure 6** illustrates.

Document settings are customizable as well, so users can alter the font size and orientation, change the archive location, or create a report in one or more of the available formats without ever displaying the DocOut window. This last option is great for reports like registers or daily statistics because it does not require any intervention from the user, which eliminates the age-old problem of running an update, only to realize the printer jammed, or the report was lost!

Fax and E-mail
Barista also offers integrated fax and e-mail capability, made possible by BBj's ability to incorporate third party Java fax libraries. From inside the DocOut window and after selecting an output type such as PDF, simply generate the file or open a supplemental window to enter fax or e-mail information.

It was important to include the ability in AddonSoftware 8 to fax and/or e-mail such documents as invoices, statements, or

continued...



Document Information

ID: 0001010384
Date: 10/01/2008 01:41 PM
As Of: 10/01/2008

Document Headings

Acme Manufacturing Company
Detail Aging Report As Of 10/01/08 Based On
Invoice Date
Customers By ID From First To Last

Output Types

- [ ] System Printer
- [✓] Portable Document File (.pdf)
- [ ] XML Document (.xml)
- [ ] Comma Separated Text (.csv)
- [ ] Tab Delimited Text (.txt)

Invoice	Invoiced	PO Number	Order Number	Trm	Tp	Future	09/01/08 -09/30/08	08/02/08 -08/31/08	07/03/08 -08/01/08	06/03/08 -07/02/08	06/02/08
00-0100	Everest Industries										
	714.832.8113										
	Mr. James Rockville										
0005053	01/31/08										1,826.77
0005058	02/02/08										3,272.00
0005089	03/02/08										3,257.23
0005101	03/08/08										3,237.10
0005114	03/18/08										3,563.98
0005127	03/30/08										1,687.50
0009017	03/04/08	None	None	3	$						1,200.00
	Total for 00-0100 Everest Industries					0.00	0.00	0.00	0.00	0.00	18,024.58
00-0200	Western Sport Distributors										
	714.833.7100										
	Robert Mansfield										
0005054	01/31/08	None	None	CD	$						281.03
0005059	02/02/08	None	None	CD	$						1,304.51
0005090	03/02/08	None	None	CD	$						3,785.28
0005102	03/08/08	None	None	CD	$						2,580.29

Output Types

- [ ] System Printer
- [✓] Portable Document File (.pdf)
- [ ] XML Document (.xml)
- [ ] Comma Separated Text (.csv)
- [ ] Tab Delimited Text (.txt)

Figure 6. The DocOut display window provides several choices for output type and fax or e-mail delivery method

sales orders, without requiring a third party solution. The Barista Fax/E-mail Queue does just that. The user can update this queue in a batch fashion to, for example, generate statements or update it interactively by browsing for and adding any previously archived document.

Barista Security

Many legacy applications fall short in their security features so the AddonSoftware developers decided to abandon the old security system and take advantage of Barista's role-based security. In Barista, developers can create users and security roles, and then assign one or more roles to each user; each security role provides application, form, and field-level security. Using the audit mechanism, developers can also log additions, deletions, or modifications to specified tables, a common requirement of the Sarbanes-Oxley Act.

Proof-of-the-pudding

Thanks to the Barista Application Framework, AddonSoftware 8 is now available in a platform-independent graphical interface. Users will find that they can navigate through AddonSoftware forms in a way very similar to other graphical applications while appreciating the care taken to maintain keyboard-oriented, "heads-down" data entry functionality for data-centric forms. Barista has delivered on its productivity gains promise and then some!

The AddonSoftware reseller community is excited about the recent release of the AddonSoftware Accounting bundle that includes the Administrative, Accounts Payable, Accounts Receivable, and General Ledger components. Development is brewing away on the Distribution modules – Inventory, Purchase Orders, and Sales Orders – with Manufacturing and Payroll applications waiting to go into the frother.

Refer to the related Barista tutorials and documentation at
www.basis.com/products/devtools/barista/documentation

For more information about the AddonSoftware product or Partner Program, call 1.800.370.9131 or visit www.addonsoftware.com

Download AddonSoftware and Barista from www.basis.com/products/bbj/download.html
Choose Release Type "Current Release," select your platform, and check Optional Files "AddonSoftware 8.0" and "Barista." In the "Register for a BASIS License" window, select the "Requesting a 30-day evaluation license" checkbox if you are not already running a BASIS DVK License.