AddonSoftware's Digital Dashboard Takes Off

By Carla Johnson
Software Developer

By Christine Hawkins
Software Developer

Digital dashboards provide a graphical view of business data, allowing anyone, especially owners and management executives, to quickly and easily make better informed decisions. To capitalize on the value of visual data presentation, the AddonSoftware® development team undertook the task of offering the value-added reseller (VAR) community a representative, well-rounded sampling of ERP widgets that would both pique prospects' interest as well as supply a prototype that VARs could use for their own vertical development. To that end, AddonSoftware by Barista® version 14.0 not only debuts a fully functional dashboard (**Figure 1**), but also provides a solid foundation for customization.

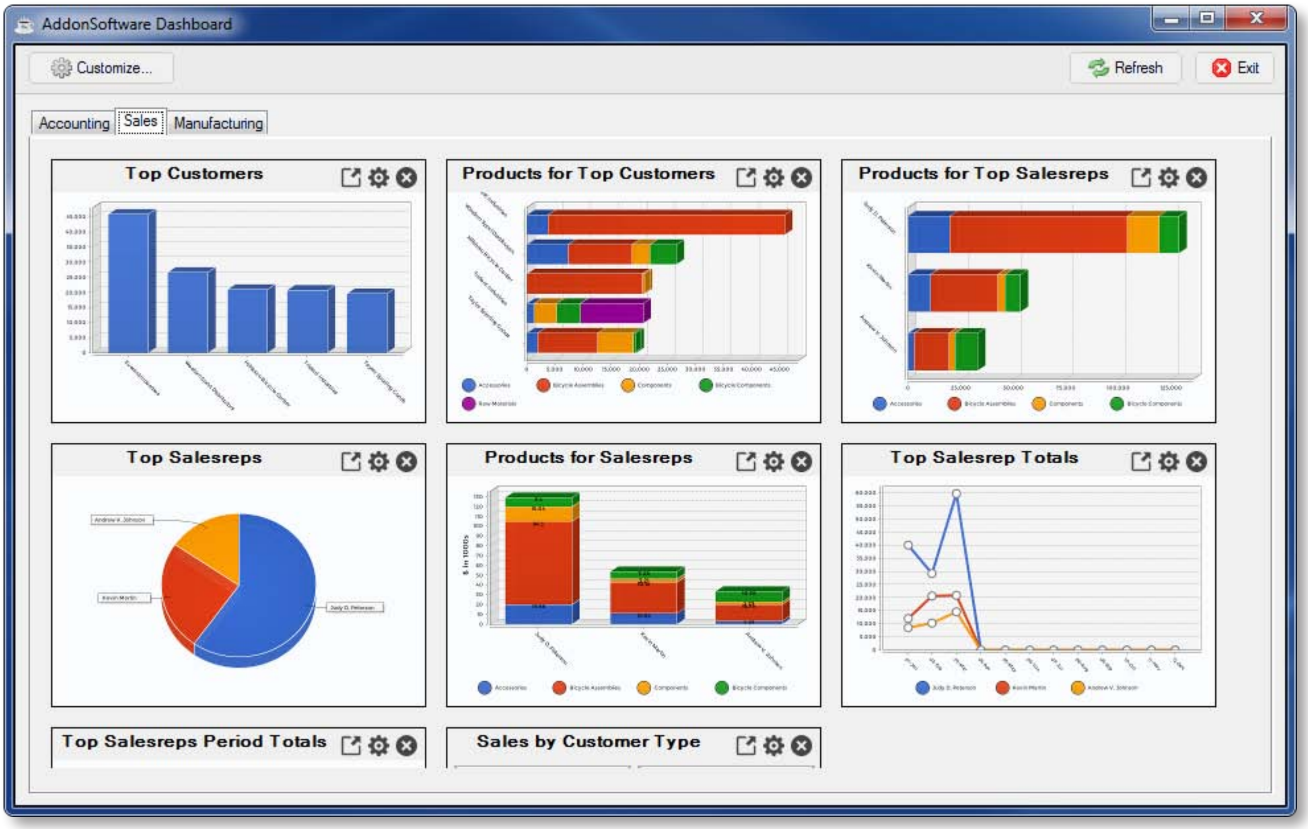
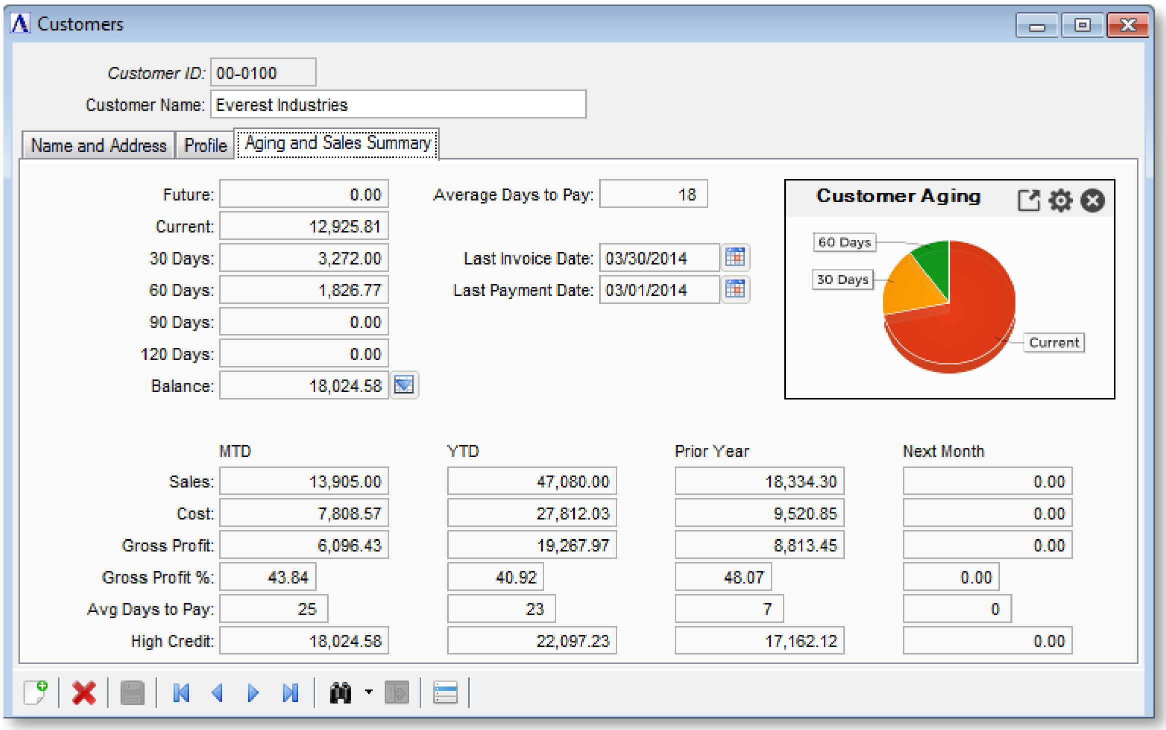


Figure 1. AddonSoftware Dashboard showing widgets in the sales category

The AddonSoftware Dashboard includes examples of most of the new BASIS Dashboard Utility's widget types, graphically displaying key data in an easily digestible manner. In addition, version 14.0 showcases the dashboard's flexibility with an embedded widget (**Figure 2**) on the Accounts Receivable Customer form. In the AddonSoftware tradition, all the code that makes this magic happen is available to the VAR community.


Customers

Customer ID: 00-0100
Customer Name: Everest Industries

Name and Address | Profile | Aging and Sales Summary

Future:	0.00
Current:	12,925.81
30 Days:	3,272.00
60 Days:	1,826.77
90 Days:	0.00
120 Days:	0.00
Balance:	18,024.58

Average Days to Pay: 18
Last Invoice Date: 03/30/2014
Last Payment Date: 03/01/2014

	MTD	YTD	Prior Year	Next Month
Sales:	13,905.00	47,080.00	18,334.30	0.00
Cost:	7,808.57	27,812.03	9,520.85	0.00
Gross Profit:	6,096.43	19,267.97	8,813.45	0.00
Gross Profit %:	43.84	40.92	48.07	0.00
Avg Days to Pay:	25	23	7	0
High Credit:	18,024.58	22,097.23	17,162.12	0.00

Figure 2. Customer form contains an embedded pie chart showing aged balances

This article unearths what we learned during our development cycle. Read about our design process and the aspects of development that were both easy and challenging, and get a peek "under the hood" at the technical details.

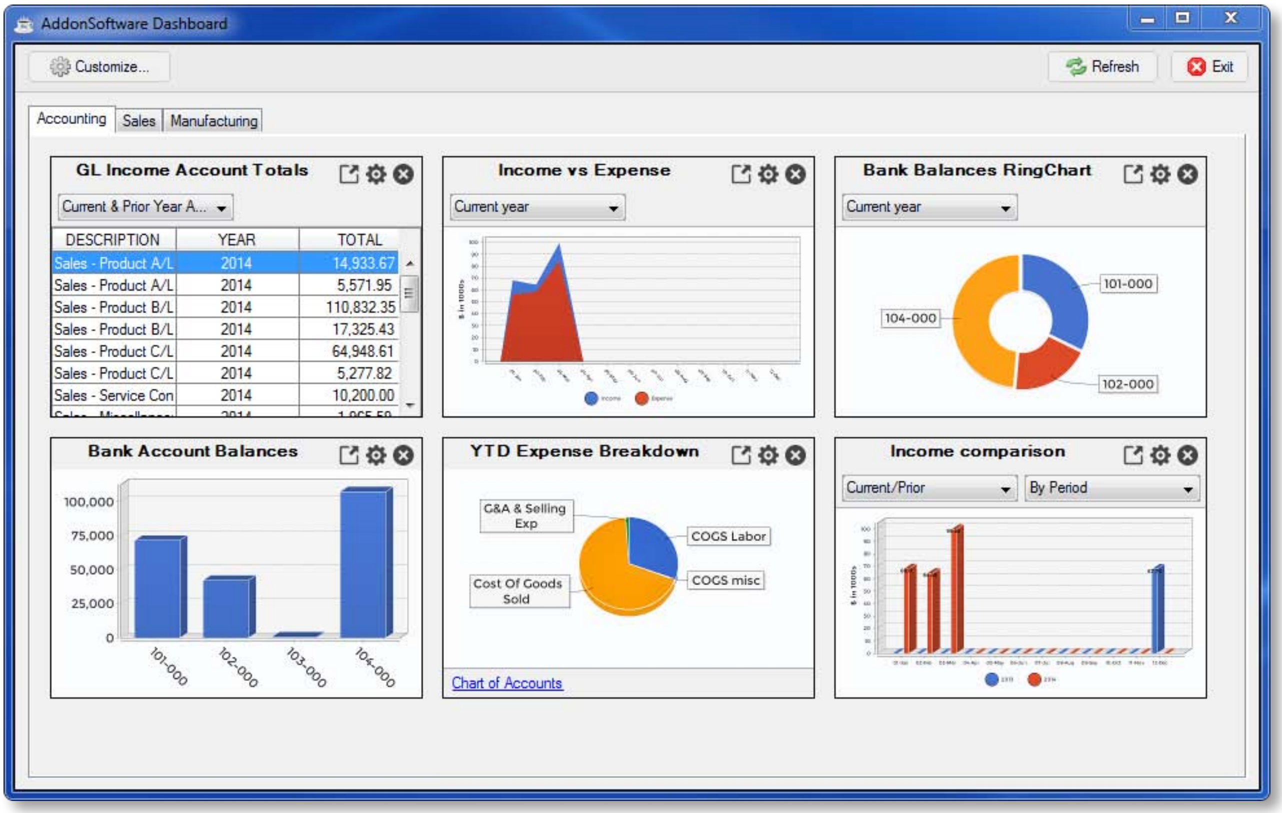
Getting Started

The AddonSoftware team was excited to participate in the development of the new BASIS Dashboard Utility and have the opportunity to work closely with BBj® engineers to help shape the dashboard's functionality and direction. As the BBj team developed the utility, we molded the mechanics to AddonSoftware – always keeping in mind our goal of assisting both VAR developers and resellers. AddonSoftware inherits all the functionality of the Dashboard Utility, including the ability to pop out individual widgets for a zoomed-in view, save or email a widget image, manage refresh options, and customize the dashboard layout.

With so much inherited functionality, we were able to focus on how to make the best use of the Dashboard Utility within AddonSoftware and the Barista Application Framework. We started by brainstorming ideas on which widgets to include in our initial release. To meet our goal of delivering a dashboard that is both customer-facing and a prototype for VARs, we identified some initial requirements. Widgets needed to

- Offer an eye-catching display
- Provide an effective presentation of data relevant to a prospect's business
- Reference data that would lend itself to a graphical depiction
- Be appropriate for the data being collected

As we considered these requirements, AddonSoftware's General Ledger and Sales Analysis data tables came to the fore as a "widget-rich" environment. Our design plan included a broad selection of widgets to benefit both prospects and VARs as shown in **Figure 3**.



DESCRIPTION	YEAR	TOTAL
Sales - Product A/L	2014	14,933.67
Sales - Product A/L	2014	5,571.95
Sales - Product B/L	2014	110,832.35
Sales - Product B/L	2014	17,325.43
Sales - Product C/L	2014	64,948.61
Sales - Product C/L	2014	5,277.82
Sales - Service Con	2014	10,200.00

Figure 3. AddonSoftware Dashboard showing widgets in the 'Accounting' category

Once we defined the core set of widgets, we turned our efforts toward designing a process to launch the dashboard in accordance with the Barista security model so the availability of widgets would tie to the user's security role. We also considered AddonSoftware's modular structure so that the dashboard would only create widgets for installed modules.

Understanding the Dashboard Utility

Just as the AddonSoftware Dashboard has laid the foundation for VARs and resellers to demo the new dashboard and customize or expand on it, the Dashboard Utility paved the way for the AddonSoftware team. Not only could we refer to the Dashboard Utility Overview in the online Help to learn about the various aspects of the utility, but we also got a boost by having a functional demo dashboard with access to the source code. And all of these resources – the online documentation, the BBj Demo Dashboards, and the Addon Dashboard – are available to anyone developing with BBj!

We began by taking the demo dashboard for a test drive to acquaint ourselves with its function, capabilities, and available widget styles. We analyzed the underlying code, and found that we could leverage pivotal routines – specifically, the logic for constructing the dashboard, its categories, and its individual widgets.

The methods for constructing the various widgets and setting their properties are consistent and intuitive, so the code was easy to understand and propagate. Furthermore, we found that because the Dashboard Utility provides defaults for colors, fonts, etc., we could create a widget very quickly with just a few parameters. We could set and/or change many additional properties if we chose, but didn't need to worry about a myriad of details to get up and running.

Implementing the Dashboard in AddonSoftware

Since we could borrow the core code from the demo dashboard to get the basics for AddonSoftware's dashboard in place, we were free to focus on the application-specific challenges. Some of these challenges were specific to widgets while others applied to the overall design and, once solved, would be in place for the benefit of others doing dashboard development.

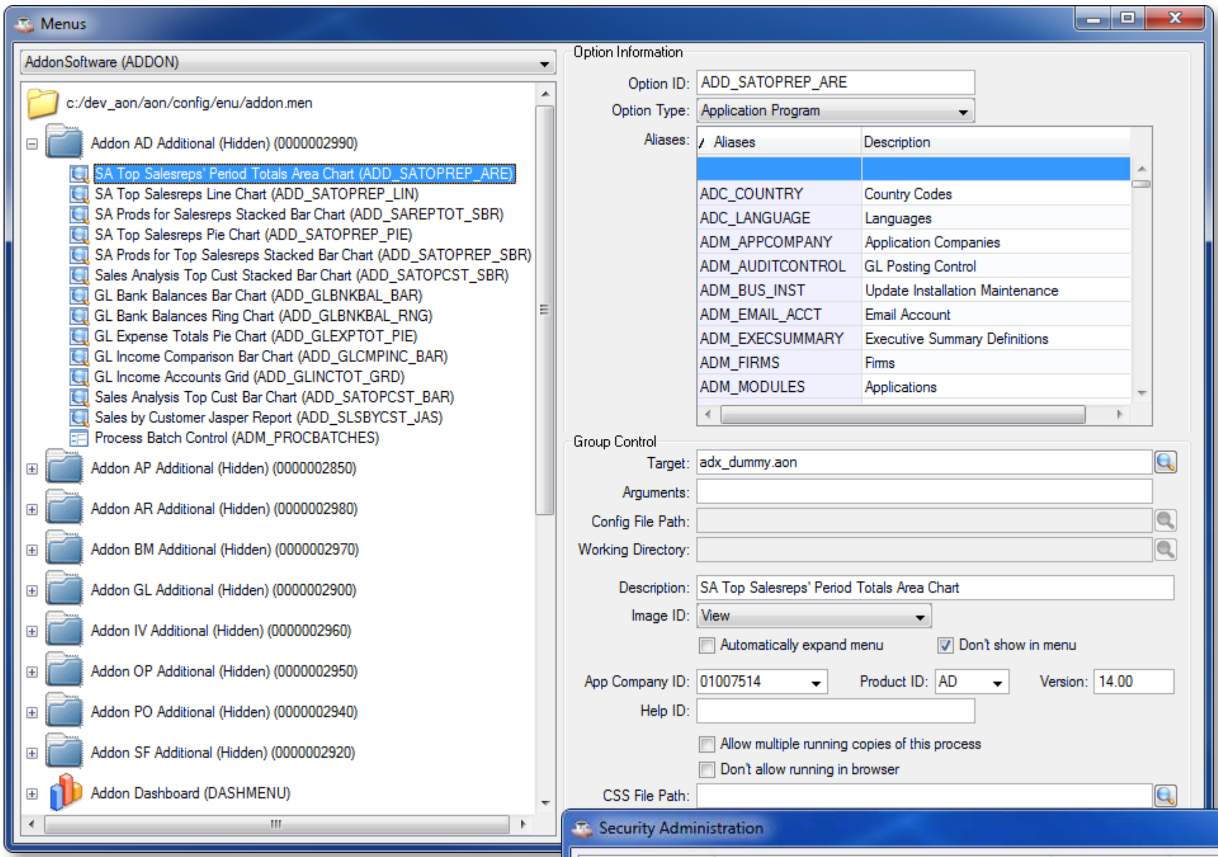
Look and Feel
In terms of overall design, we had to think about the general look and feel of the dashboard. While it was tempting to use different types of widgets, colors, and fonts, and to experiment with rendering the data as a flat vs. 3D graphic, we opted to let the utility's defaults set the theme. This decision made our job easier and also provided a more consistent look.

Launch

Since we needed the ability to launch the dashboard either inside the Barista MDI or via the browser user interface (BUI), we added code to update the progress meter in Barista's menu panel, and perform other miscellaneous initialization tasks to facilitate a BUI launch.

Security

Barista’s security is tied to menu items; therefore to address security considerations, we created hidden menu items for each widget (**Figure 4**), not just for the dashboard as a whole. This gave us the infrastructure to set Barista security options on a per-widget basis to tailor each user's version of the dashboard to their security permissions (**Figure 5**). Without the granularity provided by assigning menu items to each widget, security could only be applied at the level of the dashboard as a whole – a user could either access all of the dashboard or none of it. Clearly, that would not have been an optimal solution.



Menus

AddonSoftware (ADDON)

- c:/dev_aon/aon/config/enu/addon.men
  - Addon AD Additional (Hidden) (0000002990)
    - SA Top Salesreps' Period Totals Area Chart (ADD_SATOPREP_ARE)
    - SA Top Salesreps Line Chart (ADD_SATOPREP_LIN)
    - SA Prods for Salesreps Stacked Bar Chart (ADD_SAREPTOT_SBR)
    - SA Top Salesreps Pie Chart (ADD_SATOPREP_PIE)
    - SA Prods for Top Salesreps Stacked Bar Chart (ADD_SATOPREP_SBR)
    - Sales Analysis Top Cust Stacked Bar Chart (ADD_SATOPCST_SBR)
    - GL Bank Balances Bar Chart (ADD_GLBNKBAL_BAR)
    - GL Bank Balances Ring Chart (ADD_GLBNKBAL_RNG)
    - GL Expense Totals Pie Chart (ADD_GLEXPTOT_PIE)
    - GL Income Comparison Bar Chart (ADD_GLCMPINC_BAR)
    - GL Income Accounts Grid (ADD_GLINCTOT_GRD)
    - Sales Analysis Top Cust Bar Chart (ADD_SATOPCST_BAR)
    - Sales by Customer Jasper Report (ADD_SLSBYCST_JAS)
    - Process Batch Control (ADM_PROCBATCHES)
  - Addon AP Additional (Hidden) (0000002850)
  - Addon AR Additional (Hidden) (0000002980)
  - Addon BM Additional (Hidden) (0000002970)
  - Addon GL Additional (Hidden) (0000002900)
  - Addon IV Additional (Hidden) (0000002960)
  - Addon OP Additional (Hidden) (0000002950)
  - Addon PO Additional (Hidden) (0000002940)
  - Addon SF Additional (Hidden) (0000002920)
  - Addon Dashboard (DASHMENU)

Option Information

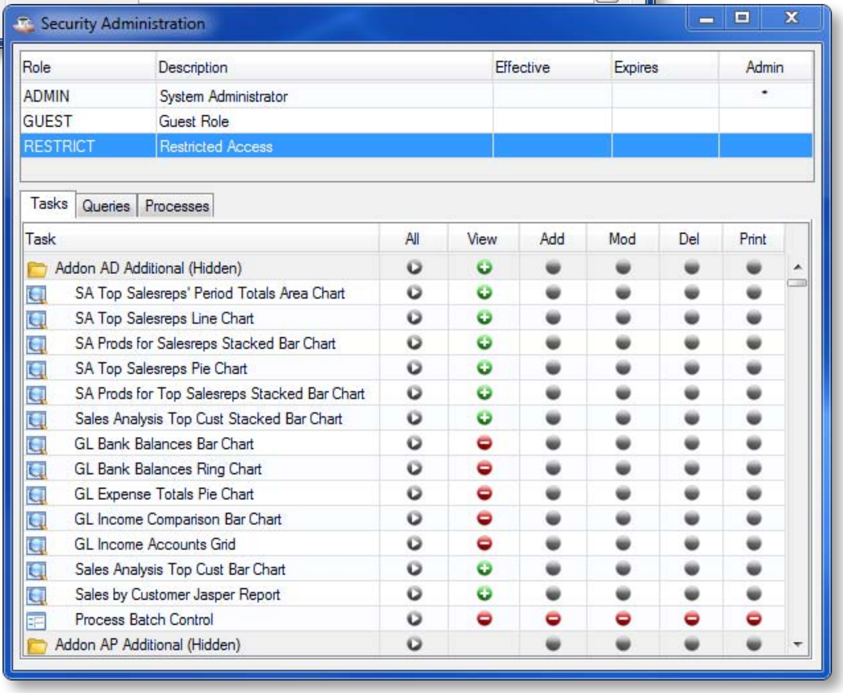
Option ID: ADD_SATOPREP_ARE
Option Type: Application Program
Aliases:

Aliases	Description
ADC_COUNTRY	Country Codes
ADC_LANGUAGE	Languages
ADM_APPCOMPANY	Application Companies
ADM_AUDITCONTROL	GL Posting Control
ADM_BUS_INST	Update Installation Maintenance
ADM_EMAIL_ACCT	Email Account
ADM_EXECSUMMARY	Executive Summary Definitions
ADM_FIRMS	Firms
ADM_MODULES	Applications

Group Control

Target: adx_dummy.aon
Arguments:
Config File Path:
Working Directory:
Description: SA Top Salesreps' Period Totals Area Chart
Image ID: View
[ ] Automatically expand menu [✓] Don't show in menu
App Company ID: 01007514 Product ID: AD Version: 14.00
Help ID:
[ ] Allow multiple running copies of this process
[ ] Don't allow running in browser
CSS File Path:

Figure 4. Hidden menu items for each widget provide a link to Barista security



Security Administration

Role	Description	Effective	Expires	Admin
ADMIN	System Administrator			•
GUEST	Guest Role			
RESTRICT	Restricted Access			

Tasks | Queries | Processes

Task	All	View	Add	Mod	Del	Print
Addon AD Additional (Hidden)	◐	⊕	●	●	●	●
SA Top Salesreps' Period Totals Area Chart	◐	⊕	●	●	●	●
SA Top Salesreps Line Chart	◐	⊕	●	●	●	●
SA Prods for Salesreps Stacked Bar Chart	◐	⊕	●	●	●	●
SA Top Salesreps Pie Chart	◐	⊕	●	●	●	●
SA Prods for Top Salesreps Stacked Bar Chart	◐	⊕	●	●	●	●
Sales Analysis Top Cust Stacked Bar Chart	◐	⊕	●	●	●	●
GL Bank Balances Bar Chart	◐	⊖	●	●	●	●
GL Bank Balances Ring Chart	◐	⊖	●	●	●	●
GL Expense Totals Pie Chart	◐	⊖	●	●	●	●
GL Income Comparison Bar Chart	◐	⊖	●	●	●	●
GL Income Accounts Grid	◐	⊖	●	●	●	●
Sales Analysis Top Cust Bar Chart	◐	⊕	●	●	●	●
Sales by Customer Jasper Report	◐	⊕	●	●	●	●
Process Batch Control	◐	⊖	⊖	⊖	⊖	⊖
Addon AP Additional (Hidden)	◐		●	●	●	●

Figure 5. Barista Security Administration controls access to widgets based on user role

Logic

Last, but not least, we added code to construct widgets only if the associated AddonSoftware application module was installed. Adding this logic brought the AddonSoftware Dashboard into alignment with AddonSoftware standards for module integration.

Combining all of these design features made the AddonSoftware Dashboard seem right at home within the Barista framework rather than having the appearance of an external bolt-on. Having addressed these issues, we turned our focus to the design of our individual widgets.

Setting Design Standards

At the widget level, we found that even though we had to revisit certain design questions with each widget, the process became easier after the first time through, and was even faster by establishing some standards.

The first question in designing our widgets was, *"Which of the many widget types would best depict the data we wanted to display*?" The answer was largely an educational step, and the team went to the Internet to learn about charting. For example, consider the two representations of sales rep data shown in **Figure 6** and **Figure 7.** Shown as a line chart, the individual sales amounts for each rep display clearly. The same data shown as a stacked area chart places the reps' sales amounts atop each other, so you also have a visual of total sales.

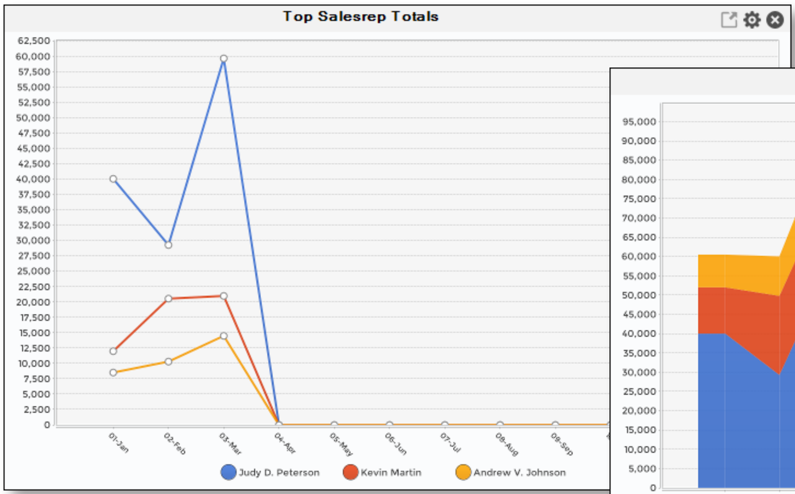


Figure 6. Line chart presents individualized data for Top Salespersons from Sales Analysis

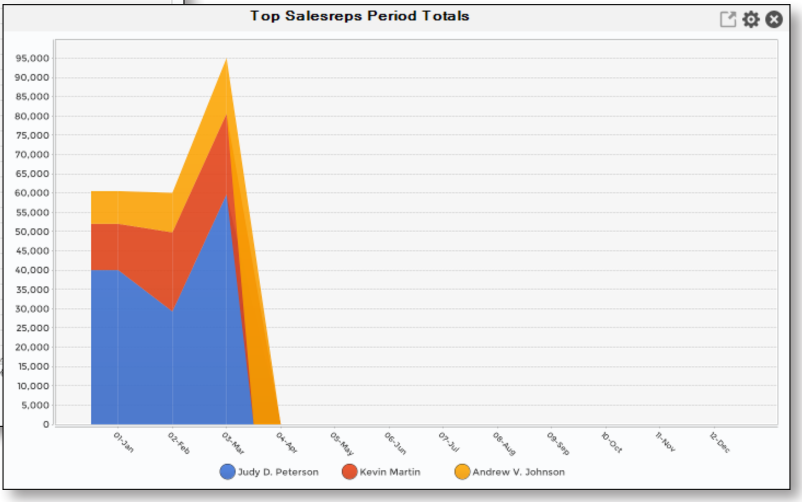


Figure 7. Top Salesrep total sales by period in a Stacked Area Chart calls out the totals for each period

It is also important to understand that different widgets require different recordset structures, so we had to decide on the widget type before we set about writing the stored procedure (SPROC). **Figure 8** shows examples of the recordsets needed for a pie chart vs. a stacked area chart.

SALESREP	TOTAL
Judy D. Peterson	128912
Kevin Martin	53460.5
Andrew V. Johnson	33237.5

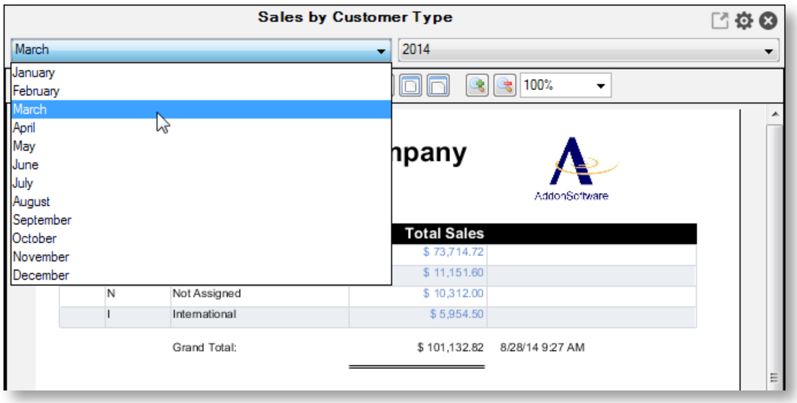
SALESREP	PERIOD	TOTAL
Judy D. Peterson	01-Jan	40008
Judy D. Peterson	02-Feb	29264
Judy D. Peterson	03-Mar	59639.5
Judy D. Peterson	04-Apr	0

Figure 8. Recordset for sales rep pie chart (left) and stacked area chart (right)

Since SPROCs are written with BBj code, we had the freedom to write them using either SQL or native file access. We wrote a few selected SPROCs with both kinds of access, then REM'd out one or the other and ran traces and/or called the SPROCs from within the Enterprise Manager to compare the performance. In general, if the desired data came from only one or two tables and was already in a normalized form and adequately indexed, SQL worked great. Otherwise, we found we could get our return recordset more quickly using native file access. That principle made it faster to design and code the remaining SPROCs.

Adding Filters

Another widget-specific consideration was whether to add filters so the user would have options for tailoring the result. For example, the Sales category contains a widget showing the AddonSoftware Accounts Receivable Drilldown Sales Report. When run stand-alone from the menu, the user establishes the month and year for the report. In the dashboard version, the widget initially appears based on defaults set out in the code, and the user can then select the period of their choice from the month/year filters (**Figure 9**).



			$ 73,714.72
			$ 11,151.60
N	Not Assigned		$ 10,312.00
I	International		$ 5,954.50
	Grand Total:		$ 101,132.82 8/28/14 9:27 AM

Figure 9. Filters allow users to change the widget data dynamically

Delivering our Final Product

At this point in the development, we had sailed over most of the hurdles, resulting in minimal ramp-up time for VARs wanting to demo or modify the AddonSoftware Dashboard. By following a process similar to our undertaking, a VAR's step into the new world of graphical data display should be straightforward. The general process is the same as any AddonSoftware development.

1. Define your goal.
2. Familiarize yourself with the user interface.
3. Review existing code logic pertaining to your goal.
4. Follow the examples provided when making your custom logic.
5. Consult the BASIS Dashboard Utility documentation, as needed.

Read on for some of the more technical details of our implementation.

Dissecting the Components

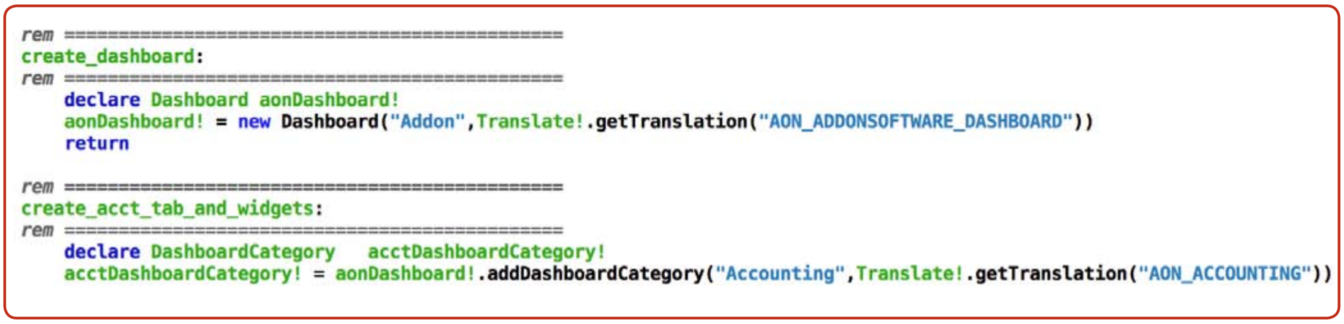
As an overview, the BASIS Dashboard Utility provides the underlying widget and dashboard objects and contains a number of components. Refer to the links at the end of this article for more information about the Dashboard Utility that is published in this issue and the online documentation.

Barista handles framework functionality such as security, document warehousing, STBLs, sysinfo, localization, and masking. AddonSoftware leverages these primarily via a program named **`adx_aondashboard.aon`**, a launcher program that contains all of the logic to interface the AddonSoftware data with the BBj and Barista components. To help with customization efforts, this program includes examples of the majority of widget types, each of which utilize calls to SPROCs to collect data. Its easy-to-follow structure lends itself to customization.

The flow of **`adx_aondashboard.aon`** is essentially this:

1. Initialize various processes and variables, including a check for BUI.
2. Create the dashboard.
3. For each category, create the category tab control.
4. For each widget, if security allows it and the associated module is installed, create the widget.

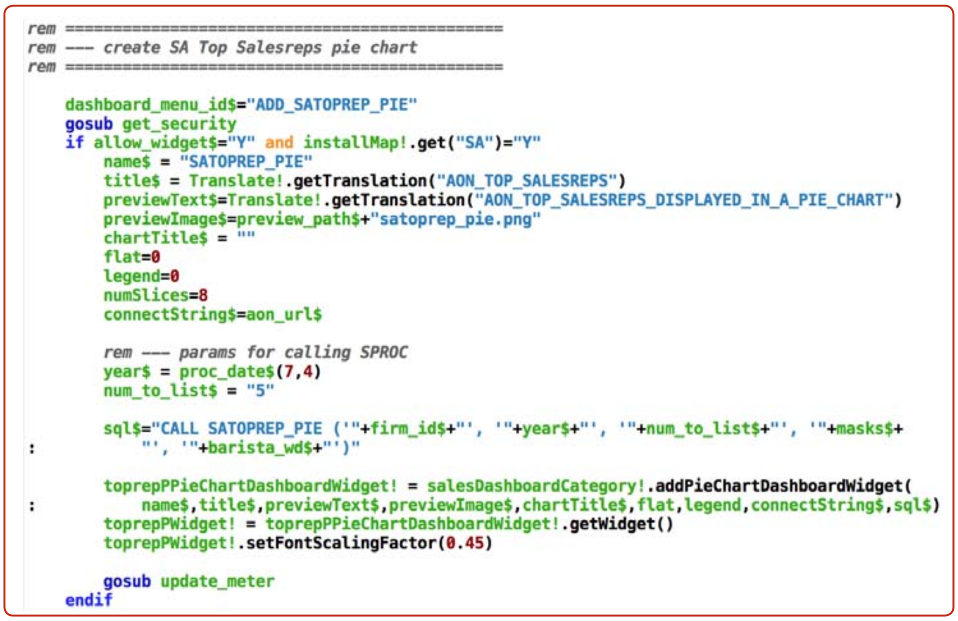
Creating the dashboard and its categories is very straightforward – just two lines of code each. To illustrate, **Figure 10** shows the code snippets that create the dashboard itself and the accounting category.



```
rem ==============================================
create_dashboard:
rem ==============================================
    declare Dashboard aonDashboard!
    aonDashboard! = new Dashboard("Addon",Translate!.getTranslation("AON_ADDONSOFTWARE_DASHBOARD"))
    return

rem ==============================================
create_acct_tab_and_widgets:
rem ==============================================
    declare DashboardCategory   acctDashboardCategory!
    acctDashboardCategory! = aonDashboard!.addDashboardCategory("Accounting",Translate!.getTranslation("AON_ACCOUNTING"))
```

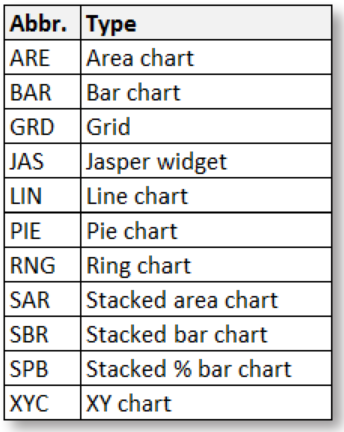
Figure 10. The code that creates the Dashboard and the Accounting category

Initially we borrowed the code for creating individual widgets from the BASIS Demo Dashboard, altered it for AddonSoftware, and then wrapped it in the logic to create the widget based on the user's security and whether or not the requisite module is installed (**Figure 11**). Note also that the user-facing text – the dashboard title, category names, widget titles, filter contents, etc. – are localized to display the appropriate text for the user's locale/language.



```
rem ==============================================
rem --- create SA Top Salesreps pie chart
rem ==============================================

    dashboard_menu_id$="ADD_SATOPREP_PIE"
    gosub get_security
    if allow_widget$="Y" and installMap!.get("SA")="Y"
        name$ = "SATOPREP_PIE"
        title$ = Translate!.getTranslation("AON_TOP_SALESREPS")
        previewText$=Translate!.getTranslation("AON_TOP_SALESREPS_DISPLAYED_IN_A_PIE_CHART")
        previewImage$=preview_path$+"satoprep_pie.png"
        chartTitle$ = ""
        flat=0
        legend=0
        numSlices=8
        connectString$=aon_url$

        rem --- params for calling SPROC
        year$ = proc_date$(7,4)
        num_to_list$ = "5"

        sql$="CALL SATOPREP_PIE ('"+firm_id$+"', '"+year$+"', '"+num_to_list$+"', '"+masks$+
:           "', '"+barista_wd$+"')"

        toprepPPieChartDashboardWidget! = salesDashboardCategory!.addPieChartDashboardWidget(
:           name$,title$,previewText$,previewImage$,chartTitle$,flat,legend,connectString$,sql$)
        toprepPWidget! = toprepPPieChartDashboardWidget!.getWidget()
        toprepPWidget!.setFontScalingFactor(0.45)

        gosub update_meter
    endif
```

Figure 11. Widget construction code wrapped with tests to check security and application installation

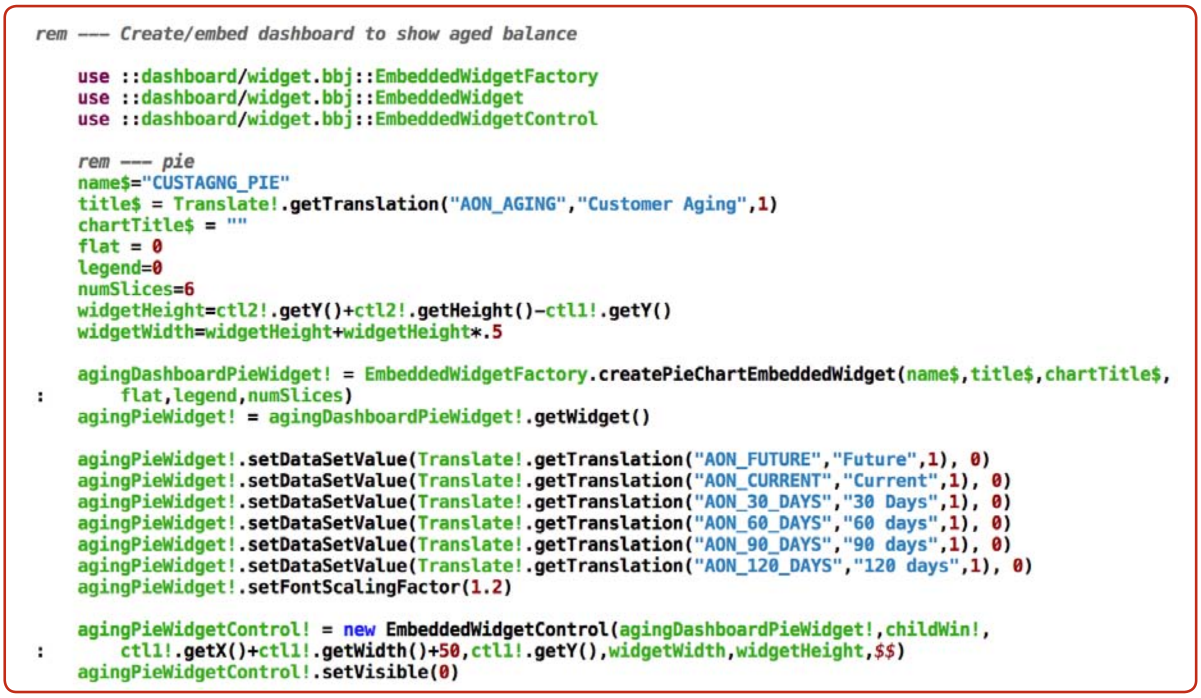


Abbr.	Type
ARE	Area chart
BAR	Bar chart
GRD	Grid
JAS	Jasper widget
LIN	Line chart
PIE	Pie chart
RNG	Ring chart
SAR	Stacked area chart
SBR	Stacked bar chart
SPB	Stacked % bar chart
XYC	XY chart

Figure 12. Three-character widget types used in Addon Dashboard naming conventions

To simplify widget creation, we developed a naming convention for widgets. We then broadened the convention to add consistency across the various components (SPROCs, Barista menu ID's, widget names, thumbnail images, and filter selection events). Per AddonSoftware standards, AddonSoftware Dashboard names include a two-character module ID and descriptive acronym. For clarity, the last four characters are an underscore followed by a three-character widget-type abbreviation as shown in **Figure 12**.

Embedding a widget in a form like we did on the 'AddonSoftware Customer' form (**Figure 2**) uses code that is very similar to adding a widget to the dashboard, except we used the EmbeddedWidget classes as shown in this excerpt from the 'After Show (ASHO)' callpoint (**Figure 13**).



```
rem --- Create/embed dashboard to show aged balance

    use ::dashboard/widget.bbj::EmbeddedWidgetFactory
    use ::dashboard/widget.bbj::EmbeddedWidget
    use ::dashboard/widget.bbj::EmbeddedWidgetControl

    rem --- pie
    name$="CUSTAGNG_PIE"
    title$ = Translate!.getTranslation("AON_AGING","Customer Aging",1)
    chartTitle$ = ""
    flat = 0
    legend=0
    numSlices=6
    widgetHeight=ctl2!.getY()+ctl2!.getHeight()-ctl1!.getY()
    widgetWidth=widgetHeight+widgetHeight*.5

    agingDashboardPieWidget! = EmbeddedWidgetFactory.createPieChartEmbeddedWidget(name$,title$,chartTitle$,
:       flat,legend,numSlices)
    agingPieWidget! = agingDashboardPieWidget!.getWidget()

    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_FUTURE","Future",1), 0)
    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_CURRENT","Current",1), 0)
    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_30_DAYS","30 Days",1), 0)
    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_60_DAYS","60 days",1), 0)
    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_90_DAYS","90 days",1), 0)
    agingPieWidget!.setDataSetValue(Translate!.getTranslation("AON_120_DAYS","120 days",1), 0)
    agingPieWidget!.setFontScalingFactor(1.2)

    agingPieWidgetControl! = new EmbeddedWidgetControl(agingDashboardPieWidget!,childWin!,
:       ctl1!.getX()+ctl1!.getWidth()+50,ctl1!.getY(),widgetWidth,widgetHeight,$$)
    agingPieWidgetControl!.setVisible(0)
```

Figure 13. Callpoint code in Addon's Customer form constructs an embedded pie chart widget

Comparing **Figure 11** with **Figure 13**, you'll notice numerous points of similarity, including the use of the getTranslation method to localize the user-facing text in accordance with AddonSoftware's multilingual capabilities.

Tips on Customizing

VARs wishing to customize the AddonSoftware Dashboard already have a roadmap in place to follow. We've done the heavy lifting so developers don't have to! Here are some things to keep in mind if you want to build your own dashboard or modify the standard.

- Customizations should pay attention to the handling of security as demonstrated in **`adx_aondashboard.aon`**.
- Making use of the same/similar naming conventions that AddonSoftware implemented in the standard product will make it easy to identify/locate the various components (SPROCs, Barista menu ID's, thumbnail images, etc.).
- Begin by reviewing the data underlying the widget you want to create so that you can decide on the right widget for the job and whether you'll want to use SQL or native file access to build your recordset.
- Don't forget that each SPROC needs to be defined in Enterprise Manager. AddonSoftware does this with the **`adx_buildsproc.aon`** utility that runs with the first launch of AddonSoftware (via Barista's Auto-Launch mechanism). VARs can create a similar auto-launch process to make sure SPROCs are re-defined after new installations or upgrades.

Summary

The BASIS Dashboard Utility throws the data visualization doors wide open and AddonSoftware's dashboard implementation gives VARs not only a ready-made solution for demonstrating graphical capability in the application, but also a great tool to use as a springboard for customization. ■

- Refer to the following resources for deepening your understanding of AddonSoftware's Digital Dashboard
  - *Dash Boredom With the Dashboard Utility*
  - *Easier Decision Making With the Dashboard Utility*
  - *Dashboard Utility Overview* in the online documentation
- Download and run the code samples