If it's Easier in BBj, Then Just XCALL it Forward

Ralph, a software engineer for Fictitious Enterprises, was perplexed.

Until now, Ralph's in-house Enterprise Resource Planning (ERP) application could do everything the company needed, but then the day came when the sales force asked for a pie chart showing the clients that accounted for the majority of sales by telephone. Making a pie chart is possible but not terribly easy using drawing mnemonics in (V)PRO/5 (Visual PRO/5® and PRO/5®). He knew that there was a BBj control that was a faster and more attractive way to make a pie chart but the ERP application, written in (V)PRO/5 for CUI or GUI, had never been run in BBj®. Additionally, BBj's charts are full-featured and offer built-in functionality that allows users to interact with them and dynamically change their contents. By right-clicking and selecting options from the popup menu, users can zoom in and out, modify the axis ranges, and even save out a local copy of the chart as an image on their computer. Ralph knew he would eventually move the whole ERP application to BBj and that the transition would be seamless but on this particular day, he just needed a convenient way to create a compelling pie chart.

Was there an easy solution? Absolutely. Ralph remembered attending a Java Break with BASIS and learning that (V)PRO/5 version 10.00 offers the new XCALL verb!

Introducing the XCALL Verb

While the XCALL syntax is similar to the CALL verb, XCALL allows an earlier generation of BBx® program to call a BBj program or subroutine. The called program or subroutine in BBj is exactly as one would write it in (V)PRO/5: Create a label if desired, create an ENTER statement with the needed parameters, write whatever code is needed to perform the task, and then end the subroutine or program with an EXIT statement.

This new verb allows (V)PRO/5 to call BBj and run any code in the program or subroutine, and even retrieve results of the call through variables that were passed by reference into the call. XCALL would perfectly fulfill the sale force's request by allowing the ERP program to tell BBj to display a pie chart with a specified set of values all in one elegant line!

Ralph decided that instead of spending days trying to create a reasonably attractive pie chart himself, he would simply take advantage of the existing charting functionality provided in BBj via XCALL!

Follow along online with Ralph as he implements and deploys a pie chart (**Figure 1**) on his end users' desktops from within his company's ERP application, using XCALL. > >

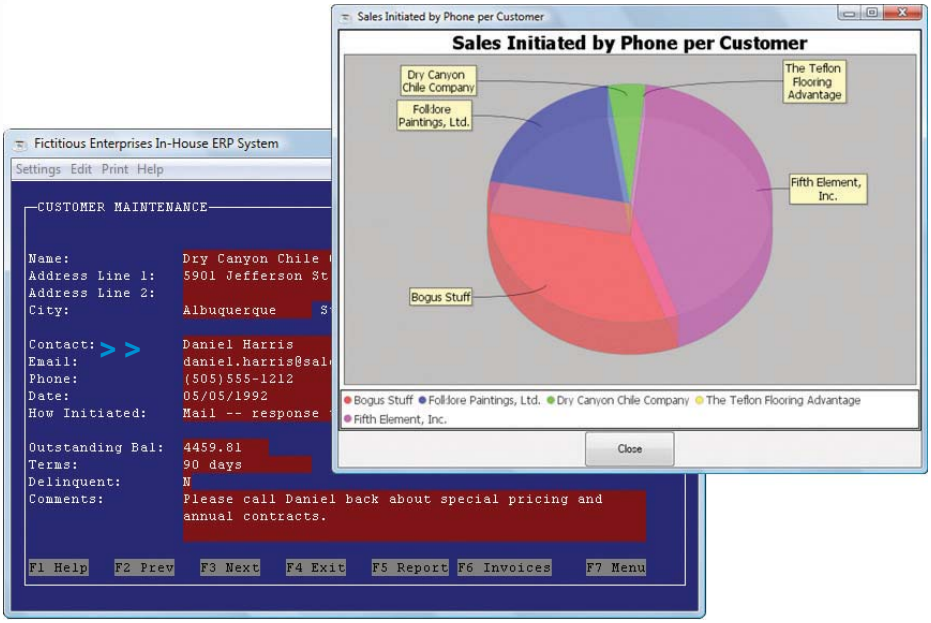

Figure 1. Sample pie chart from within the ERP application

By Shaun Haney
Quality Assurance Engineer

Planning

Having decided to use XCALL to show the sales people their pie charts from the ERP application, Ralph looked at how to configure (V)PRO/5 and BBj for XCALL. To accomplish this task, (V)PRO/5 needs to access BBj, and BBj in turn needs access to each salesperson's display.

BBj has some configuration options that are quite different from (V)PRO/5. (V)PRO/5 is mostly a single-tiered product which can use a data server, but the data server is optional. BBj is a client-server product with a central server to manage the interpreter, event-handling, file-handling, etc. when used in thin client mode, which is the default and most common configuration. All servers are part of BBj Services and while you can distribute BBj across multiple tiers it will run in a single-tier on the same machine without further configuration from the user.

The XCALL Server is a BBj program that handles (V)PRO/5's requests to run BBj subroutines and can run wherever it is needed. You can run XCALL Server centrally on the same machine as BBj Services if clients don't need a GUI or you can run an instance of the XCALL Server on each client's machine allowing BBj to display GUI as well as handle other kinds of requests.

Because Ralph wanted salespeople to be able to view the pie chart on their machine, he chose to run the XCALL Server individually on each machine. The PORTCOMMAND option, explained later in this article, allows (V)PRO/5 to launch the XCALL Server on demand, relieving the salesperson of any need to know that the XCALL Server is even there. Ralph installed the BBj ThinClient on a mapped drive, easily accessible from each salesperson's machine, saving him the trouble of installing and maintaining multiple installations.

Verifying (V)PRO/5 is Ready For XCALL

Beginning in version 10.01, (V)PRO/5 is preconfigured to run the XCALL Server in BBj 10.03 automatically when executing an XCALL statement. Ralph looked at his configuration to verify that (V)PRO/5 was configured correctly for his needs. First, he examined the short xcall.ini file to make sure the PORTCOMMAND option was in place. The PORTCOMMAND option automatically launches an instance of the XCALL Server on the same machine as (V)PRO/5. Then Ralph looked at the default line for PORTCOMMAND. On a Windows installation, the line reads: **> >**

```
portcommand  C:/Program Files/basis/bin/bbj.exe"  -cC:/Program Files/basis/cfg/config.min" -tT2 -q xcallserver.bbj - 0
```

Here's what this line contains:	
`C:/Program Files/basis/bin/bbj.exe`	Invokes the BBj interpreter
`-cC:/Program Files/basis/cfg/config.min`	Specifies the configuration file XCALL Server should use
`-tT2`	Specifies that the "invisible" SysGui terminal should be used. Users won't see the console, but will see GUI components of the application. In Ralph's case, the sales people will see the window containing the pie chart.
`-q`	Do not display the BBj splash screen when launching the XCALL Server.
`xcallserver.bbj`	By default, xcallserver.bbj is in BBj's PREFIX. Otherwise, it can be found at: **`<bbj install path>/utils/XCALL/xcallserver.bbj.`**
`- 0`	Specifies port 0 to the XCALL Server. When this port is specified, the XCALL Server will use whatever port is available.

Ralph made a couple of adjustments to the default PORTCOMMAND. Since he was accessing BBj on a mapped drive, he adjusted the paths to BBj and the config.min file. Then he added **-RHphydeaux** after the **-q** option, since BBj Services is running remotely on Phydeaux. Next, Ralph briefly perused (V)PRO/5's config.bbx file to verify it contains the line **XCALL=xcall.ini**, which indicates that (V)PRO/5 should load the xcall.ini file and that this file is in (V)PRO/5's current directory.

Writing the BBj Subroutine

Now that Ralph had (V)PRO/5 and BBj configured to use XCALL, it was time to write the BBj subroutine to display the pie chart. An XCALLed subroutine consists of an optional label, an ENTER statement specifying any arguments that the calling program will pass in, and an EXIT statement at the end of the routine, just like a (V)PRO/5 subroutine.

Ralph needed to decide what kind of data to pass in; XCALL supports strings, numbers, integers, and 1-, 2-, and 3-dimensional arrays. The BBjPieChart control takes a title as well as a name and a value for each slice in the pie so Ralph decided to pass the title "Sales Initiated by Phone per Customer" in as a string, the name of each customer in as a one-dimensional array of strings, and the number of phone-initiated sales for each customer as an array of numbers.

Ralph wrote a BBj subroutine that displays a pie chart based on the arguments passed in to the subroutine and then he saved this program on Phydeaux. He also added the pathname of the subroutine file to Phydeaux's prefix.

Calling the BBj Subroutine from (V)PRO/5

Once Ralph wrote the BBj subroutine, he decided to write a very short (V)PRO/5 program to test it before finally integrating the XCALL statement into the ERP application. He populated **title$** with the title of the chart and filled the **customer$** and **sales** arrays with string and numeric data, respectively. He then XCALLed the BBj subroutine with the following statement: **> >**

```
XCALL pathToBBjProgram$+   show_pie_chart" err=handl_err,title$,customer$[ALL],sales[ALL]
```

Here's what this line contains:

pathToBBjProgram$	The variable Ralph used to store the name that the BBj program will call. The BBj program must be accessible to BBj Services, either by use of a full path on the same machine as BBj Services, accessible via a data server with use of data server syntax, or contained in BBj's PREFIX. (V)PRO/5's PREFIX does not affect BBj's being able to find the file.
:show_pie_chart"	The label Ralph used for the BBj subroutine. XCALL can call subroutines without a label. However, using labels allows for including several BBj subroutines in the same file.
err=handl_err	The error branch for the XCALL statement, which like CALL, must be included before the subroutine's argument list. Including this expression after or within the argument list would cause it to be interpreted as a boolean comparison.
title$,customer$[ALL],sales[ALL]	The variables that comprise the argument list for the subroutine.

Once XCALL is invoked, (V)PRO/5 connects to the XCALL server, which then launches the BBj subroutine in its own BBj interpreter. It is this interpreter that will then show the window with the pie chart. Download and review the short (V)PRO/5 program, **xcall.bbx.src** and the subroutine Ralph wrote, **xcall.bbj.src**, from the link that appears at the end of this article.

Other Features of XCALL

This solution to Ralph's dilemma demonstrates a fairly straight-forward way to use the XCALL verb, but consider these additional XCALL verb command options:

- **Specify a timeout**
  The **tim=** option to XCALL allows the user to specify a timeout for XCALLing a program in number of seconds, including the amount of time it would take the BBj program to run and then respond back. This option is useful for non-interactive calls, where the network may be slow or unreliable; the default is 0 or "wait forever."

- **Employ the new error-handling options introduced with the XCALL verb**
  Error handling for XCALL introduces some new features. If an error occurs in an XCALLed program, XCALL will return an **!ERROR=85**. This is a general error informing the user that an unspecified error occurred in XCALL. The text for the error holds the real clue to what happened and can be retrieved in (V)PRO/5 by calling the **ERRMES** function with a value of -1. While in the past, **ERRMES(ERR)** gave a generalized message that corresponded to the error, **ERRMES(-1)** now gives specific information related to an error and is useful for any error, not just 85. Ralph decided to take full advantage of this new feature by printing out **ERRMES(-1)** as part of the diagnostics in his error branch.

- **Run the XCALL server locally or remotely and even on a different port**
  The host and port can be specified for XCALL in a couple of ways. The first way is to specify a host and port in an xcall.ini configuration file. The second way to specify a remote host and port is with the **mode=** option. A couple of the modes supported for XCALL are **"RemoteHost=<host>"**and **"RemotePort=<port>"**. By default, the values are **"RemoteHost=localhost"** and **"RemotePort=4444"**. Since Ralph is running **xcallserver.bbj** locally on each machine and not changing the port the XCALL Server runs on, he can accept the defaults.

- **Specify the PORTCOMMAND option at runtime**
  If needed, the PORTCOMMAND can be specified in the mode string with **"PORTCOMMAND=<command line>"**. This option is useful in situations where the location of the BBj ThinClient can not be determined before run time.

After Ralph tried out his XCALL statement in a sample program and verified that it worked, he proceeded to integrate the new pie chart feature into the sales force's software. Fictitious Enterprises now takes advantage of the new pie chart feature as well as many other BBj features through XCALL.

Conclusion

Fictitious Enterprises' fictional but feasible example shows one way to > >

use XCALL to leverage a BBj feature in (V)PRO/5. It also alludes to other ways to configure XCALL for non-GUI and non-interactive features. XCALL is a good and fast way to utilize features in BBj that are either difficult to implement or nonexistant in (V)PRO/5. Here are just a few other examples of other BBj features developers can access in (V)PRO/5 via XCALL:

- Fax documents
- Print PDFs
- Create VKEYED and XKEYED files
- Create BBjForms
- Create reports with iReport(r)
- Consume web services
- Parse XML documents
- Easily send email
- Public Key Cryptography
- Access databases via JDBC instead of ODBC
- Publish Google Docs

XCALL is a useful mechanism for transitioning code to BBj piecemeal, while still allowing a (V)PRO/5 infrastructure to continue running productively. The CALL-like syntax and the ability to transfer data via variables or access a common data file or DBMS, makes this transition a lot less code-intensive than an equivalent transition using SCALL and data files. Indeed, XCALL is handy within environments where BBj and (V)PRO/5 must coexist and interact. ■

- Download and extract links.basis.com/10xcallcode to run/review the sample subroutine xcall.bbj.src
- Re-take a Java Break with BASIS - "Mix 'n Match PRO/5 and BBj - REDUX" – for an example of XCALL. Click here and select "play." links.basis.com/javabreak
- Refer to the online documentation for more information on XCALL www.basis.com/onlinedocs/documentation/commands3/xcall_verb.htm