Rapid Development Using *Kilauea*

By Jeff Steffanina

Getting any graphic user interface (GUI) application up and running can be a real challenge. Not only does it require commitment, but a well-conceived plan. *Kilauea* contains features that enable developers to create GUI applications, while at the same time employing existing BBx® programming skills. This article demonstrates the features of *Kilauea* and highlights some of the keys to using this new BASIS development tool.

Developing a plan to take advantage of these tools will produce an application that performs at a consistently high level with standard features such as a data dictionary. In *Kilauea*, the BASIS data dictionary builder, called DDBuilder™, has been converted to GUI, making it easy to use. Creating a data dictionary also makes MKEYED files accessible to external products such as Oracle, Excel, Microsoft Access, and Informix. The external interface in these and other packages is managed using the 32-bit BASIS ODBC Driver™.

Conservative practice dictates that all data files are normalized. But, in many cases, normalizing can be an expensive and time-consuming process. With *Kilauea*, however, BASIS has created a feature to define a file with multiple record formats. In other words, the new BASIS data dictionary eliminates the need to normalize data files.

Non-normalized records are defined in DDBuilder as "views." With this new feature, *Kilauea* allows the developer to take advantage of the data dictionary without having to redesign data file structure or existing code. Still, the use of the data dictionary on any existing file does not preclude its use without the data dictionary. Once the data dictionary is developed, input windows can be created quickly and easily with the *Kilauea* ResBuilder™. This point-and-click utility also enables the creation of complex windows for use as templates in either typical or custom developments.

A new characteristic of ResBuilder is that the actual resource file is stored in ASCII format. From this format, ASCII files can be edited without having to restart the utility. In addition, once the data files have been populated, the new *Kilauea* Program Wizard may be used to create reports and file maintenance routines. The Program Wizard works by prompting the user through a series of questions and then creating a BBx program. The program created by the Program Wizard is heavily remarked and offers the developer an opportunity to customize.

Migrating Existing Code

New BASIS products center on GUI. *Kilauea*, though not technically a new product, is rather a GUI enhancement to the familiar BBx environment. All existing code will work in *Kilauea*, while providing the option to mix character-based and GUI applications on multiple platforms.

When designing a GUI project, the existing code should be well-structured and modular. By creating and maintaining structured code, the developer can include a substantial portion of that code in his or her GUI application. Subroutines that perform functions unrelated to screen and cursor control are also candidates for use in the GUI environment. For example, an editing subroutine that verifies the presence of an inventory part number will probably work with little or no modification in the new GUI program.

Creating An Application

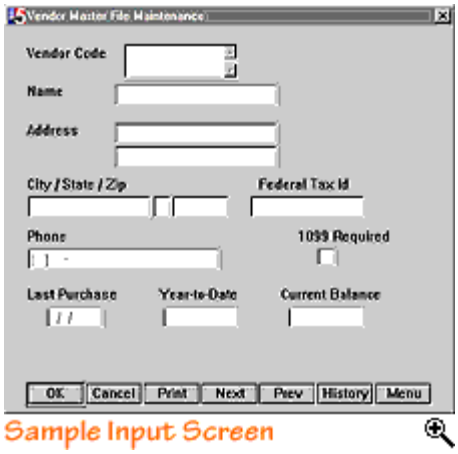
Creating an application using these new tools takes very little time. Following this step-by-step example, begin by using the GUI data dictionary to create the data source ACCOUNTING, which consists of several accounting tables. Then, create the VENDOR master table using DDBuilder. This process is simple, and the GUI logic is well presented. Pay close attention to the prompts for the required configuration file "config.tpm," discussed in greater detail below.

Table: VENDOR

VendorCode	C10	VendorName	C20
Address1	C10	Address2	C20
City	C15	State	C2
Zip	C5	FederalTaxId	C9
PhoneNumber	C9	Req1099	C1
LastPurchaseDate	C8	YrToDateAmt	N8
CurrentBal	N8		

Next, create the following input screen using ResBuilder. Notice the inclusion of the "in-field" editing for the phone number. This was done using the new *Kilauea* mnemonic INPUTE. This mnemonic is basically the GUI version of the INPUTE command introduced in BBxPROGRESSION/4®.

This demonstration also creates astandard set of buttons across the bottom of all windows. When thewindow is displayed, the necessary buttons will become"active." Inactive windows will



Sample Input Screen

display their labels in gray, active buttons will be shown in black. At this point, take advantage of "keyboard navigation," which automatically advances "focus" in the window from one object to the next. Each object on a window is assigned a unique identification value. When keyboard navigation is active, focus is shifted to the object with the next highest *id* value. This *id* value is also passed to the event queue, which is discussed later.

At this point, the "typical" BBx code to invoke the new windows created with ResBuilder can be added. Most of this simple logic is taken from the examples in the documentation and from the Visual PRO/5™ training seminar offered by BASIS. The following code displays the vendor screen created in ResBuilder.

```
  2000 REM Display a Resource Window
2010 LET APWIN=RESOPEN("AP.BRF")
2020 LET A$=RESGET(APWIN,1,1); REM from the APWIN library,
retrieve window 1
2030 LET SG=UNT; OPEN(SG) "X0"; REM open the SYSGUI device
2040 PRINT (SG) 'RESOURCE' (LEN(A$)), A$; REM display
```

Now, add the BBx coding to handle all the input from the GUI screen. When a GUI window is opened, activity tracking begins. Activity tracking ensures that any activity that occurs is funneled to one place, the event queue. The opportunity to determine exactly what happened occurs when an event is passed to the queue. For example, the mouse has either moved, or the left mouse button was double-clicked when it was pointing at the fourth button from the left at the bottom of the window. In short, all activity on the window is reported to the event queue. To provide additional control, a window may be created to track a specific event. This special event mask is defined by using a series of hex flags. Finally, the event queue is managed by using a typical READ RECORD command. When an event occurs, the READ RECORD receives a standard event string that provides specific information related to the event.

To demonstrate, the routine in the file http://www.basis.com/advantage/mag-v1n3/code-radkilauea.txt is an example prepared by the BASIS staff. It contains the necessary code to both read the event queue and determine exactly what event has occurred. This sample program also displays each event as it occurs. Notice how the code reads the event string template **`E$`**. This template, which is created using the new function TMPL, always returns the same status string. The command used to establish the template is **`DIM E$:TMPL(sg)`**.

Next, the code that makes the data dictionary available for the vendor table is added. The BASIS data dictionary is not as difficult to work with as it appears. The key is in the setup process. First, be certain that the configuration file "config.tpm" correctly identifies the path to the data dictionary components. Be sure to review the documentation and samples provided. The files are organized in a logical manner, but take the time to become familiar with the relationship between the data location and the directory scheme employed by the data dictionary. To get the "feel" for the utility, create several small files, and write the necessary code to access them. If the data dictionary is started without creating "config.tpm," the data dictionary states that "config.tpm" does not exist and creates one automatically. The code needed to gain file access via the data dictionary is listed in the data dictionary documentation.

Finally, create a report that uses the new "print preview" utility. BASIS' print preview operates like the print preview features in other packages by generating a GUI display of the report that can be paged forward and back. To invoke print preview, add a new mode (MODE="PREVIEW") to the OPEN of a SYSPRINT device.

This simple project was completed in about two hours. For further assistance with a GUI project, BASIS has provided an exceptional documentation set making extensive use of HTML-based data retrieval. The entire Visual PRO/5 documentation set is in HTML format and is readily accessible, providing ease of access and functionality.

In short, *Kilauea* offers the developer an array of impressive tools for rapid development. As with any new project, time and effort must be invested to make GUI happen. It is, however, an investment well rewarded.

For a full copy of the BBx code used in this exercise, email me at mailto:jsteff@advcbci.com.