[9.7] Creating 'dynamic' dialog boxes

You may run into a programming situation in which you want to prompt for user input with the *Request* function, but you don't know the variable in advance. You can use *expr()* with a string argument to create the dialog box, like this:

```
expr("request " & char(34) & promptn & char(34) & "," & vname)
```

where *promptn* is the prompt string, and *vname* is the variable name as a string. For example, if

promptn = "what?"
vname = "myvar"

then the expression results in a dialog box like this:

```
what?:
```

and the string that the user enters will be stored in *myvar*. The value of this approach is that the actual prompt string and variable name can be stored in *promptn* and *vname* by preceding code, then the single *expr()* executes the dialog box. In other words, you can change the contents of a dialog box as needed, depending on run-time conditions.

This idea can be extended to all functions that can be used in a dialog box: *Text*, *DropDown*, and *Title*, in addition to *Request*. The basic principle is to build the complete *Dialog ... EndDlog* block up as a string. Each program line is separated by ":".

This dialog box demonstrates the use of all four dialog box functions.

To explain each option, as well as make it easier to build the dialog box, I have defined these functions:

dbttl():	Dialog box title (Title)
dbreq():	Dialog box request (Request)
dbdrd():	Dialog box drop-down menu (DropDown)
dbtxt():	Dialog box text string (Text)
dbend():	Terminate the dialog box string

These are the steps you use in your program to create the dialog box:

1. Initialize a string variable to start building the dialog box string
2. Call the four functions above, as needed, to make the dialog box you want
3. Terminate the dialog box string
4. Display the dialog box with *expr()*

The code shown below creates my dialog box example.

```
dbdemo()
Prgm

¨dbdemo() - dynamic dialog box demo
¨31 aug 99/dab
¨dburkett@infinet.com

¨Define local variables
local promptn,vname,boxtitle,sometext,ddtitle,dditems,ddvar,dbox

¨Initialize dialog box items
"what number?"»promptn
"myvar"»vname
"BOX TITLE"»boxtitle
"This is some text"»sometext

"drop-down here:"»ddtitle
{"item1","item2"}»dditems
"dropvar"»ddvar

¨Initialize the dialog box string
"dialog"»dbox

¨Build the dialog box string
dbttl(dbox,boxtitle)»dbox
dbreq(dbox,promptn,vname)»dbox
dbdrd(dbox,ddtitle,dditems,ddvar)»dbox
dbtxt(dbox,sometext)»dbox

¨Terminate the dialog box string
dbend(dbox)»dbox

¨Display the dialog box
expr(dbox)

EndPrgm
```

In this example, I use the local variable *dbox* to hold the dialog box string. Note that the *dbox* is initialized to "dialog"; you must always initialize your dialog box string just like this.

After *dbox* is initialized, I call each of the four functions to create the title, a request, a drop-down menu and some text. Note that the first argument of each function is the dialog box string *dbox*. Each function simply appends the appropriate string to the current *dbox*. The table below shows the arguments for each function.

Description	**Call convention**	**Arguments**
Create box title	dbttl(db,titletext)	db: dialog box string titletext: string to use for title
Add Request	dbreq(db,promptn,vname)	db: dialog box string promptn: prompt string vname: variable name as a string
Add drop-down menu	dbdrd(db,prmpt,ddlist,ddvar)	db: dialog box string prmpt: prompt string ddlist: list of menu item strings ddvar: variable name as a string
Add text	dbtxt(db,txt)	db: dialog box string txt: text string

Here is the code for the functions:

```
dbttl(db,titletxt)
func
db&":title "&char(34)&titletxt&char(34)
Endfunc

dbreq(db,promptn,vname)
func
db&":request "&char(34)&promptn&char(34)&","&vname
Endfunc

dbdrd(db,prmpt,ddlist,ddvar)
func
db&":dropdown "&char(34)&prmpt&char(34)&","&string(ddlist)&","&ddvar
Endfunc

dbtxt(db,txt)
func
db&":text "&char(34)&txt&char(34)
Endfunc

dbend(db)
func
db&":enddlog"
Endfunc
```

Note that you don't really need to use these functions, instead, you can just build the dialog box as a big string and use it as the argument to *expr().* This is a better approach if you only have a single dialog box.