# Barista Plumbing *Exposed!*

## Integrate your handcrafted BBj code with the Barista MDI

By Christine Hawkins

## Introduction

One of the powerful features in the Barista® Menu system is the capability to run programs other than Barista forms, making it possible to run your custom BBj® code from within the Barista MDI. In this tutorial, we'll see how to "level up" your custom code by integrating it with the Barista MDI, providing not only a Barista look and feel, but also extended functionality through the use of the Barista menus and toolbuttons.

exm_functions.bbj
We'll begin by looking at a sample program that demonstrates how to incorporate the Barista Menu and Toolbar to an "external" form, that is, a form designed outside of the Barista Application Framework.

Rather than using the form for traditional data I/O, `exm_functions.bbj` is a tutorial in itself; we'll intercept keyboard and mouse activity on all of the menu items and toolbuttons, displaying information about each function in a grid control. Selecting or de-selecting a checkbox in a given grid row will toggle the enabled status of the specified toolbutton and corresponding menu item.

By running the `exm_functions.bbj` program and reviewing the code, you will quickly learn how to incorporate Barista functionality into your own forms.

cust_form.bbj
Next, we'll review a simple customer form written in BBj, but outside of the Barista Application Framework. The form allows basic add, change, and delete operations on a customer table. While you can launch the form easily via the Barista menu, the functionality is limited, and the look and feel is inconsistent with other Barista forms.

For more information on GUI programming in BBj and other examples of the customer form, go to http://www.basis.com/solutions/Guide_to_GUI_Programming.pdf.

cust_form_bar.bbj

For the finale, the `cust_form_bar.bbj` example combines the original `cust_form.bbj` with sections of code from `exm_functions.bbj`, allowing us to preserve the investment in our custom code while "perking it up" with integrated Barista menu and toolbuttons!

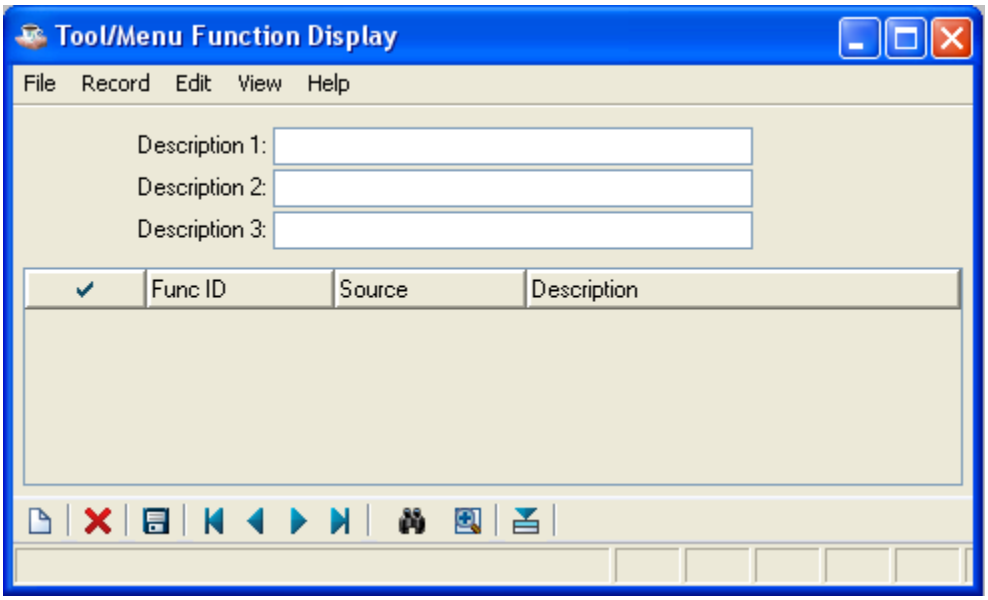
The code for all of the sample programs and associated .arc files is included in *Appendix A*.

We stored all of the program and .arc files in `C:/Samples`; you can add and run all of the programs directly from the Barista menu. *Appendix B* contains screenshots depicting the required menu setup for our three forms.

## exm_functions.bbj

The .arc for `exm_functions.bbj` describes a very simple form, containing just three input controls and labels. If viewed from the BASIS IDE, we see:


But when we run `exm_functions.bbj` from the Barista menu, this more robust form emerges:




The first several lines of `exm_functions.bbj` handle this general set-up:

- instantiate the Translator! object so our form can run in different languages
- initialize functions/variables
- open the SysGUI device
- setup a group namespace that lets us use the Barista progress meter and monitor when we have accessed a menu or toolbutton

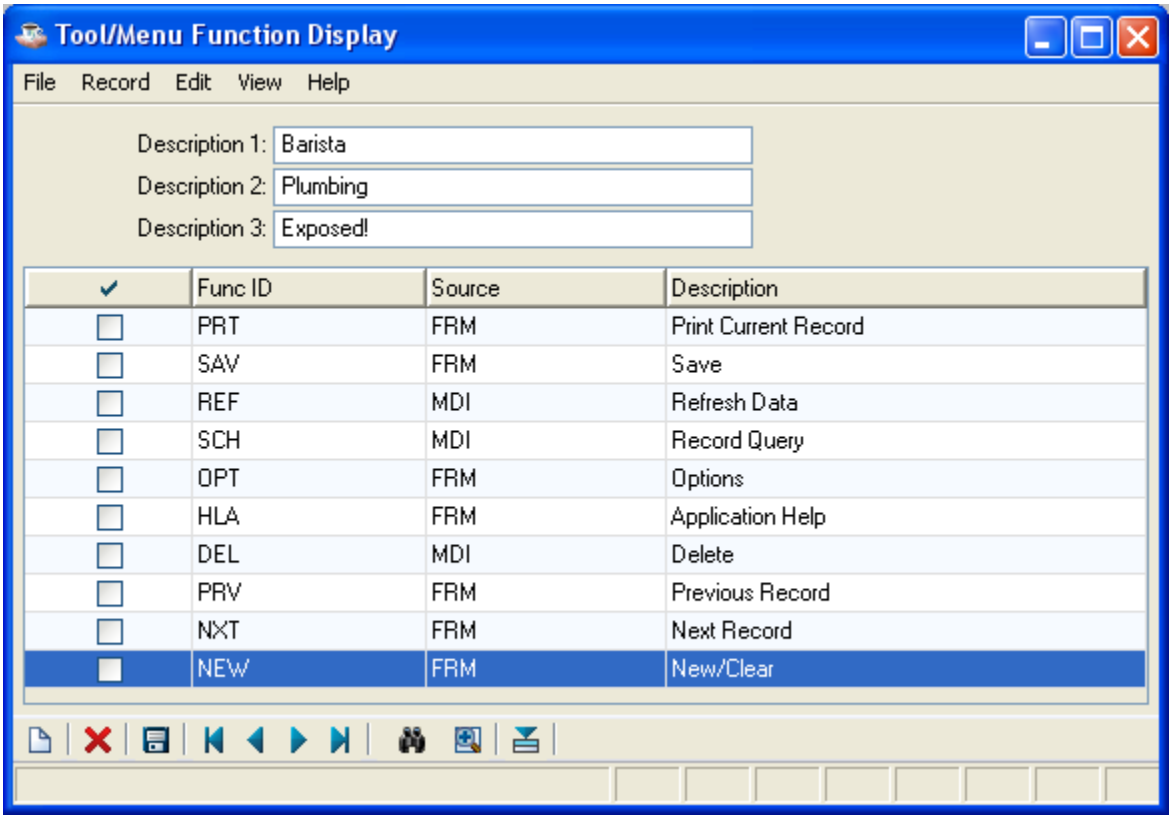
The next section is where our form comes to life. We use the Barista publics `bac_mdi_ctls.bbj` and `bam_enable.bbj` to build a list of all menu and toolbutton functions available in Barista, then add the menu and toolbar to the form and activate all desired functions. Because it is a tutorial form, `exm_functions.bbj` installs and activates a complete set of functions, so you can see the values returned to Barista as you select each one.

Now that we know what menu items and toolbuttons we need, we open the .arc and add the menu and toolbar using bam_controls.bbj, and then add a grid control with bam_grid_init.bbj (see the disp_form: and def_grid: subroutines). The grid control could have been built into the .arc, but using bam_grid_init.bbj gives us a handy way to define grid behavior, as well as allowing multi-language capability when defining the column headings.

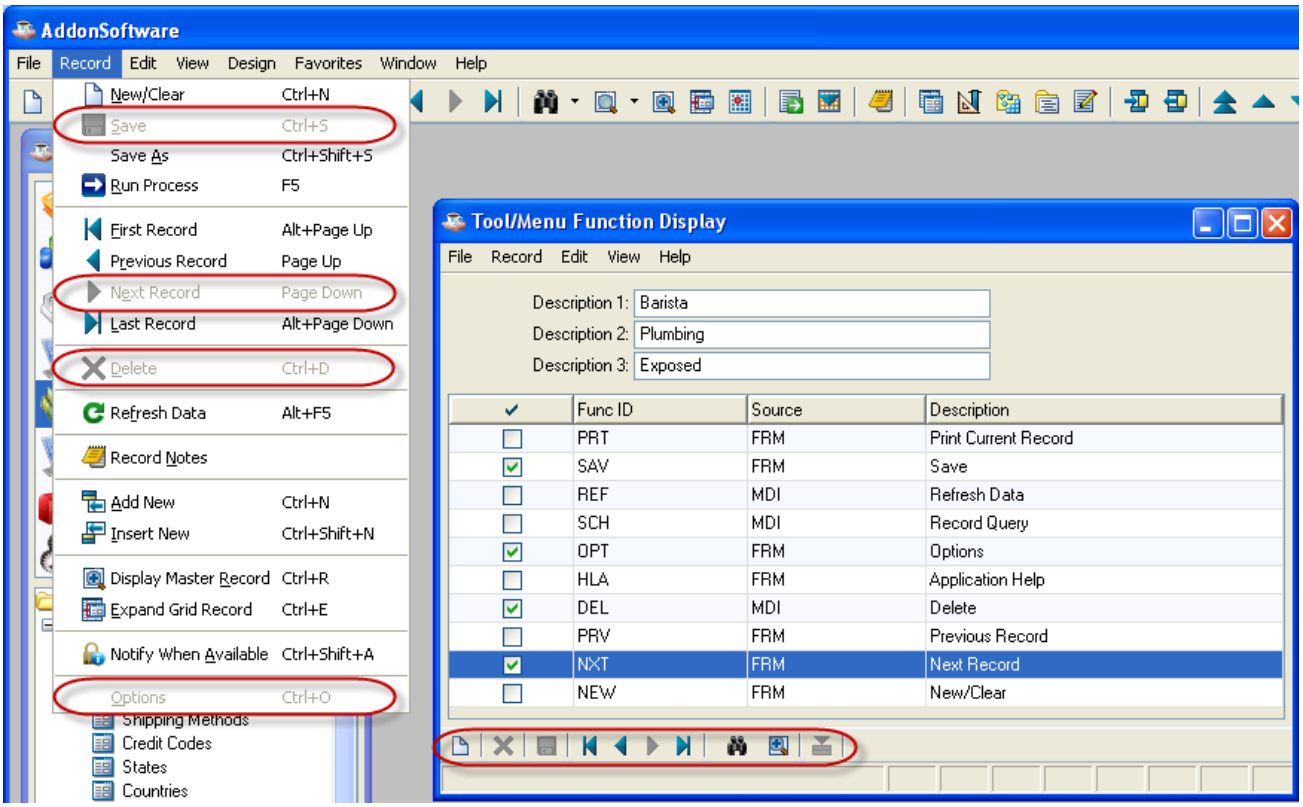
The description fields in `exm_functions.bbj` are inert. No events fire when you enter data or move around in these controls. The emphasis in this form is to illustrate the

various function names that fire, their source (did you click on the MDI toolbar or on the form's toolbar?), how to enable and disable them, and how to intercept desired functions so you can take appropriate action in your code (write a record, delete a record, move to a new record, etc.). Each toolbutton or menu item selected (via mouse or keyboard) adds a row of information to the grid, along with a checkbox. Selecting/unselecting the checkbox toggles the enabled status of the specified control.

For example, when we run the form and use either the keyboard or mouse to click on menu items and toolbuttons on the MDI and form, `exm_functions.bbj` fills the grid with all of the functions we have used, a description, and whether we invoked the function from the MDI menu/toolbar or from the menu/toolbar on the form itself as shown here:




If we select some of the checkboxes in the first column of the grid, Barista disables the menu item and toolbutton for the corresponding function as follows:




How does all of this happen? Fortunately, Barista does most of the heavy lifting, making it easy for you to incorporate this sort of functionality into your own programs.

When we added the menu and toolbar to the form using `bam_controls.bbj`, that program registered callbacks for the menu select and toolbutton push events, so any time we select a menu item or a toolbutton on the form, we'll be sent to the route_func: routine. In addition, in the event_ctl: section of `exm_functions.bbj`, we set a callback for the group namespace variable that monitors our task (`task_name$+".func"`), and executes the get_active_func: routine if a menu or toolbutton item is selected in the MDI. In these routines, we use the SysGUI!.getLastEvent() method to determine the function we've selected. Then, in the route_active_func: routine, we compare this value to the functions that we want to take action on.

The route_active_func: routine in exm_functions.bbj is a framework you can use in your custom programs to execute code based on menu and/or toolbar selections. We'll learn more about this when we look at the sample customer form. In exm_functions.bbj, every case in the `switch` is just a placeholder, and after the `swend`, we "fall through" to the

routine that adds the selected function information to the grid.

The code that enables and disables controls fires because of the callbacks registered for the grid check on/off events. When we check or uncheck one of the boxes, we're taken to the toggle_status: subroutine, which determines the grid row and associated function code. We use another Barista public, `bam_enable.bbj` to enable or disable the selected menu item and toolbutton.

Now that you see how Barista incorporates the menu and toolbars on a form, and how it intercepts and handles the associated events, you can begin to see `exm_functions.bbj` as a toolkit from which you can borrow code to integrate Barista functionality into your custom forms.

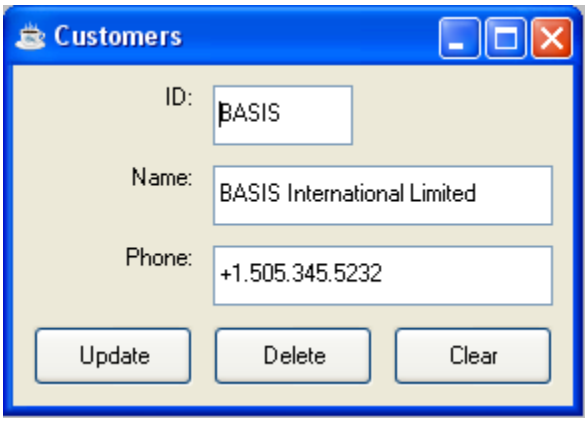
## cust_form.bbj

Now let's take a look at a form that, while simple, is representative of the sorts of forms you may have in your own application. The `cust_form.bbj` program can be run from a BBj SysConsole, but we've hooked it up to the Barista menu for convenience.

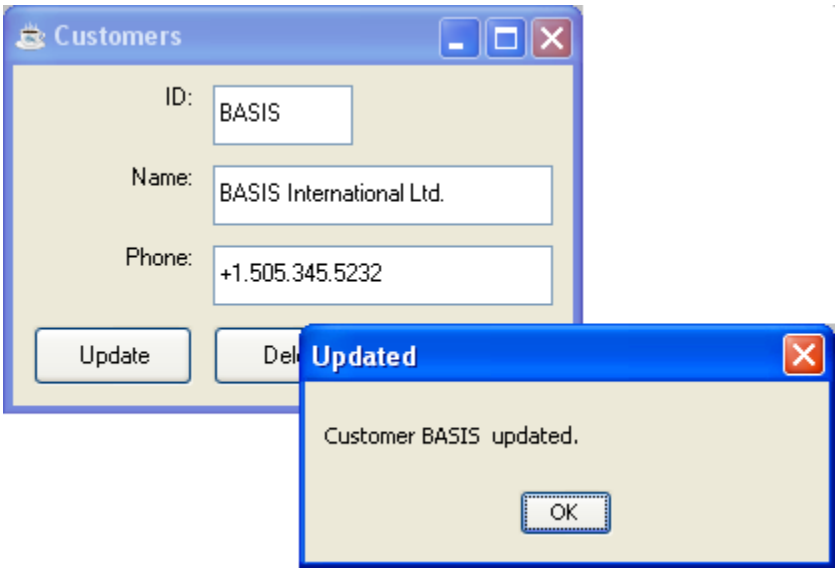
This program opens a customer table. If the table doesn't exist, it creates it and uses `dread` to add some records. The form (`cust_form.arc`) contains input fields for the customer ID, a name/description, and a phone number. In addition, it has three buttons that allow us to delete or update a record, or clear the form.

The program `cust_form.bbj` registers callbacks when we close the form, edit or lose focus in the customer ID field, or push any of the three buttons. There is no modern record navigation. You must know a customer ID in order to call up any given record.

When first launched, `cust_form.bbj` reads and displays the "BASIS" customer record such as:



We can clear the form and enter a new record. Note that if we clear out the ID field, `cust_form.bbj` will disable both the [Update] and [Delete] buttons. Any time we update or delete a record, we get this message box confirming the action:




As is, `cust_form.bbj` is functional and we can run it easily from the Barista menu. However, it lacks the look of other Barista forms, as well as the ability to interact with the menu or toolbuttons.

## cust_form_bar.bbj

The `cust_form_bar.bbj` program uses cust_form.bbj as a starting point, and harnesses the functionality demonstrated in `exm_functions.bbj` to produce a custom form that bridges the gap between your application and Barista.

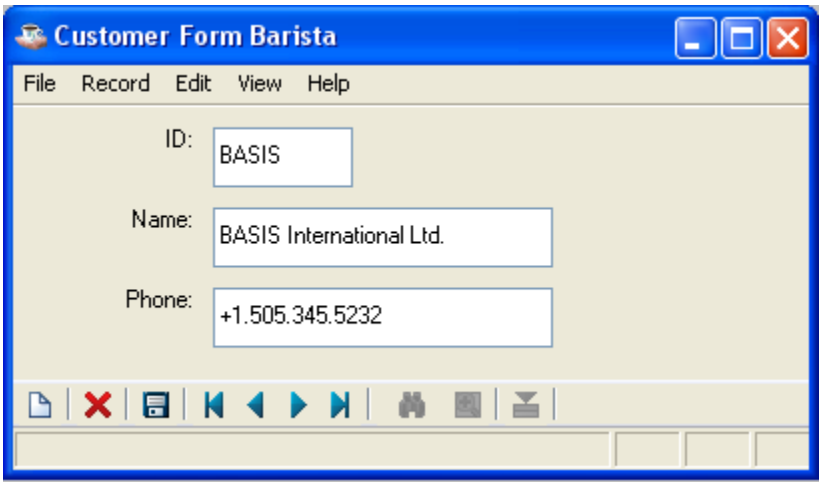
The [Delete], [Update], and [Clear] buttons have been removed in favor of Barista's New, Save, and Delete toolbuttons. In addition, we've enabled standard record navigation – an improvement over the old version of the form.

We modified the new .arc file in two ways. First, we deleted the buttons we no longer need. Next, we changed the window control ID from 101 to 1000. Barista uses (and expects) certain controls to have IDs in pre-defined ranges. Developers rarely need

to concern themselves with control IDs when designing forms within the Barista Application Framework. However, if integrating an "outside" .arc and program with Barista, the various control IDs in use need to be examined and possibly altered to avoid conflicts. See *Appendix C* for more information about Barista control numbering.

To create the `cust_form_bar.bbj` program, we began with the original, and merged in sections from exm_functions.bbj as needed. As you examine the source code in *Appendix A*, you can see that the majority of code from `exm_functions.bbj` has been propagated to `cust_form_bar.bbj`. We aren't using a grid in our customer form, so we didn't need the tutorial code for building and displaying the grid.

Our revised form now looks and behaves like other Barista forms!




The `exm_functions.bbj` program displayed and enabled a superset of menu items and toolbuttons. Our customer form has a more limited set of buttons, as described in the remarks found in the source code (see *Appendix A*).

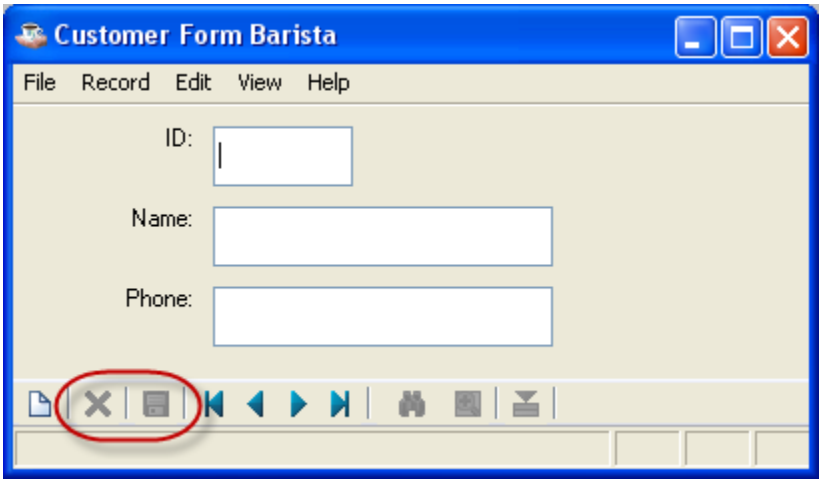
The original program contained callbacks for the [Delete], [Update] and [Clear] buttons that routed to corresponding subroutines. Those subroutines have been retained in the new program, but are now executed as a result of toolbutton or their corresponding menu events.

The code to open/create the customer file remains, along with the code to fetch and display records. Our new navigation buttons use the same fetch and display code.

In the `exm_functions.bbj` program, the code to route program execution based on the selected menu item or toolbutton was basically an empty switch block that just loaded information about the event into the custom grid. In our new customer form, we

flesh out the various `case/break` paragraphs to perform the appropriate action (new, save, delete, last record, next record, etc.).

As with the old form, when we clear and begin new data entry, we are able to disable the delete and save toolbuttons and menu items. In the old form, the code operated directly on the delete and update buttons. In our new form, we use the Barista `bam_enable.bbj` program in conjuction with the appropriate control IDs:




Our new program retains the hard-coded message boxes issued when records are saved and deleted. These could be replaced by adding messages to the Barista Administration => System Messages file, and using the Barista bac_message.bbj public (found in the disp_message: routine in the sample code). However, our objective in this lesson is to show how to use and preserve your custom code while adding Barista form and function, so we've retained the original messages.

If our sample table were part of a BASIS data dictionary, we could extend the functionality of our new form even further. We could route the find and search functions to the Barista `bam_inquiry.bbj` public, and we'd also be able to use the print/print all menu items to automatically produce a DocOut pdf preview of our file contents! We'll cover this and other more advanced functions in upcoming lessons.

# Appendix A

⇧back to top

## exm_functions.bbj

```
rem Function Display
rem Proprietary Software. BASIS International Ltd.
rem Program ID: exm_functions.bbj  <Jan 05, 2010>

rem This program is an example of developer built custom form used in conjuction with the Barista
MDI
rem tool and menu functions. Your custom program should be added to the Barista menu system as an
rem "Application" type of record so all the necessary STBLS are passed to the program.

    if stbl("+USE_SETERR")="YES" seterr error_proc

rem --- Get Translator Object

    use ::bbtranslator.bbj::BBTranslator
    declare BBTranslator Translator!
    Translator!=BBTranslator.getInstance("examples",stbl("+USER_LOCALE"),null(),dsk("")+dir("")
+stbl("+EXAMPLES_PROP"))

rem --- Functions

    def fnstr_pos(tmp0$,tmp1$,tmp0)=int((pos(tmp0$=tmp1$,tmp0)+tmp0-1)/tmp0)

rem --- Get Menu Information Passed by Barista

    dim sysinfo$:stbl("+SYSINFO_TPL")
    sysinfo$=stbl("+SYSINFO")

rem --- Misc Definitions

    process_id$="EXM_FUNCTIONS"
    task_val$=process_id$+str(dec(info(3,0)):"0000")

rem --- Open Tables

    num_files=1
    dim
open_tables$[1:num_files],open_opts$[1:num_files],open_chans$[1:num_files],open_tpls$[1:num_files
]
  ope
    n_tables$[1]=stbl("+GUI_DEVICE"),open_opts$[1]="O"
    gosub open_tables
    gui_dev=num(open_chans$[1])

rem --- Get Objects and Set Up Group Namespace

    sysGUI!=BBjAPI().getSysGui()

    grpSpace!=BBjAPI().getGroupNamespace()
    grpSpace!.setValue("+build_task","ON");rem --- Starts the Barista MDI Progress Meter

    hshMenus!=new java.util.HashMap()
```

```
rem --- Set Function Enable/Disable String

    call stbl("+DIR_SYP")+"bac_mdi_ctls.bbj"
    call stbl("+DIR_SYP")+"bam_enable.bbj",gui_dev,"CREATE","IDLE-INIT-
MDI",rd_sys_mdi_ctls$,able_map$,"",0

rem sys_act_ctls$ contain the active functions for this task. To activate all available
functions, use:

    sys_act_ctls$=rd_sys_mdi_ctls$

rem To activate only selected functions, use the following. The function values are contained
rem in variables called sys_<func>_ctls$ as seen below.

rem    sys_act_ctls$=
rem :       rd_sys_new_ctls$+
rem :       rd_sys_sav_ctls$+
rem :       rd_sys_del_ctls$+
rem :       rd_sys_sch_ctls$

    call stbl("+DIR_SYP")+"bam_enable.bbj",gui_dev,"ACTIVE-
UPDATE","MDI",sys_act_ctls$,able_map$,"",0

rem --- Get Barista Attribute Arrays

    call stbl("+DIR_SYP")+"bam_attr_init.bbj",attr_def_tbl$[all],attr_def_col$[all],"ALL"

rem --- Main Process

    gosub disp_form
    gosub def_grid
    gosub get_func_desc
    gosub get_form_coords
    gosub resize_form

    able_ctls$=sys_act_ctls$
    gosub  enable_ctls

    frmCustom!.setVisible(1)
    frmCustom!.focus()
    gosub set_active_task
    frmCustom!.setCursor(0)

    grpSpace!.setValue("+build_task","OFF");rem --- Stops the Barista MDI Progress Meter

event_ctl:rem --- Event Control

    frmCustom!.setCallback(frmCustom!.ON_CLOSE,"exit_prog")
    frmCustom!.setCallback(frmCustom!.ON_RESIZE,"resize_form")
    frmCustom!.setCallback(frmCustom!.ON_ACTIVATE,"set_active_task")
    frmCustom!.setCallback(frmCustom!.ON_RESTORE,"set_active_task")

    grdFuncs!.setCallback(grdFuncs!.ON_GRID_CHECK_ON,"toggle_status")
    grdFuncs!.setCallback(grdFuncs!.ON_GRID_CHECK_OFF,"toggle_status")

    grpSpace!.setCallbackForVariable("+"+task_val$+".func","get_active_func")

    process_events
```

```
toggle_status:rem --- Subroutine to Enable or Disable Selected Function

    sel_row=grdFuncs!.getSelectedRow()
    cell_state=grdFuncs!.getCellState(sel_row,0)
    sel_func$=grdFuncs!.getCellText(sel_row,1)
    able_ctls$=seval("rd_sys_"+cvs(sel_func$,8)+"_ctls$")

    if cell_state=1
        gosub disable_ctls
    else
        gosub enable_ctls
    endif

    return

set_active_task:rem --- Set Barista MDI Focus to This Form

    grpSpace!.setValue("+active_task",task_val$)
    gosub update_ctls
    frmCustom!.focus()

    return

route_func:rem --- Get Function From Form Based Toolbar/menubar

    evtQueue!=sysGUI!.getLastEvent()
    tempCtl!=evtQueue!.getControl()
    active_func$=tempCtl!.getUserData()
    func_source$="FRM"

    goto route_active_func

get_active_func:rem --- Get Function Value From MDI/Namespace

    active_func$=""
    active_func$=sysGUI!.getLastEvent().getNewValue(err=*return)
    if active_func$="" return
    grpSpace!.setValue("+"+task_val$+".func","")
    func_source$="MDI"

route_active_func:rem Route Function to Subroutine

    func_str$="EXT;NEW;SAV;DEL;SCH;";rem --- Add the required three character function to
activate it.
    switch fnstr_pos(active_func$,func_str$,4)
        case fnstr_pos("EXT",func_str$,4)
            gosub exit_prog
        break
        case fnstr_pos("NEW",func_str$,4)
        break
        case fnstr_pos("SAV",func_str$,4)
        break
        case fnstr_pos("DEL",func_str$,4)
        break
        case fnstr_pos("SCH",func_str$,4)
        break
        case default
        break
    swend
```

```
    gosub add_to_display

    return

add_to_display: rem --- Add Retrieved Function Code to Custom Grid

    if pos(active_func$=active_str$,3)=0 then
        new_row=grdFuncs!.getNumRows()
        grdFuncs!.setNumRows(new row+1)
        grdFuncs!.setSelectedRow(new_row)
        grdFuncs!.setRowVisible(new_row)
        grdFuncs!.setCellText(new_row,1,active_func$)
        grdFuncs!.setCellText(new_row,2,func_source$)
        grdFuncs!.setCellText(new_row,3,hshMenus!.get(active_func$))
        active_str$=active_str$+active_func$
    endif

    return

get_func_desc:rem --- Get Function Descriptions for This Sample Display Program

    ini_chan=unt
    open(ini_chan,err=*next)stbl("+FILE_SET")
    ini_read$=""

    while 1
        read(ini_chan,end=*break)ini_data$
        if pos("[MenuDefinitions]"=ini_data$)<>0 then ini_read$="YES"
        if ini_read$="YES" and pos("MENU_"=ini_data$)=1 and pos(ini_data$(6,1)="_>")=0
            iniDesc!=ini_data$(23)
            ini_desc$=iniDesc!.replace("&","")
            hshMenus!.put(ini_data$(6,3),ini_desc$)
        endif
    wend

    return

open_tables:rem -----<Open Tables

    call stbl("+DIR_SYP")+"bac_open_tables.bbj",
:0,0,open_tables$[all],open_opts$[all],open_chans$[all],open_tpls$[all],table_chans$[all],0,open_
status$

    if open_status$<>""
        msg_id$="ENTRY_OPEN_ERROR"
        dim msg_tokens$[1]
            msg_tokens$[1]=open_status$
        gosub disp_message
        goto exit_prog
    endif

    return

disp_message:rem --- Display Message Dialog

    call dir_syp$+"bac_message.bbj",msg_id$,msg_tokens$[all],msg_opt$,table_chans$[all]
    return

disp_form:rem --- Display Custom Form
```

```
    dim ctl_misc$[20];rem --- DIMs Template Needed by bam_controls

    dim attr_tbl$[len(attr_def_tbl$[0,0])/5];rem --- DIMs Template Needed by bam_controls
    attr_tbl$[0]=attr_def_tbl$[0,0]
    attr_tbl$[fnstr_pos("OPTS",attr_def_tbl$[0,0],5)]="T";rem Adds Toolbar, menubar and statusbar
to form
    attr_tbl$[fnstr_pos("WINT",attr_def_tbl$[0,0],5)]=sysinfo.task_desc$;
:rem Incoming menu item desc. for form titlebar

    dim attr_col$[len(attr_def_col$[0,0])/5,10];rem --- DIMs Template Needed by bam_controls
    dim rec_data$[len(attr_def_col$[0,0])/5,10];rem --- DIMs Template Needed by bam_controls

    form_id$="c:/samples/exm_functions.arc"

    if pos(stbl("+FORM_NAV_BAR",err=*next)="TB")<>0 nav_bar$=";NAV"

    form_type$="FMSTD";rem --- Type of form
rem             "FMSTD - Linear Form"
rem             "FMGRD - Grid Form"

    bar_type$="MAINT_FORM"; rem --- Type of navigation bar on form
rem            "MAINT_FORM - Maintenance form"
rem            "MAINT_COMBO - Header/detail maintenance form"
rem            "MAINT GRID - Maintenance grid"
rem            "OPT_FORM - Option entry form"
rem            "OPT_GRID - Option entry grid"

    call stbl("+DIR_SYP")+"bam_controls.bbj",
:gui_dev,sysGUI!,form_id$,form_coords[all],frmCustom!,
:"DISPLAY-"+form_type$+"-"+bar_type$,open_chan$[all],ctl_misc$[all],attr_tbl$[all],x$[all],x!

    form_disp$="YES"

    return

def_grid:rem --- Define Custom Grid
rem This grid is manually added and retrieves it's column headings from the examples property
file
rem specified at the top of the program. If control num(stbl("+GRID_CTL")) is used as the grid
control
rem number, the Barista window settings public will also "memorize" the grid column widths.

    grdFuncs!=frmCustom!.addGrid(num(stbl("+GRID_CTL")),5,80,300,100)

    def_cols=4
    dim gattr_col$[def_cols,len(attr_def_col$[0,0])/5]

    gattr_col$[1,fnstr_pos("DVAR",attr_def_col$[0,0],5)]="STATUS"
    gattr_col$[1,fnstr_pos("CTLW",attr_def_col$[0,0],5)]="50"
    gattr_col$[1,fnstr_pos("CTYP",attr_def_col$[0,0],5)]="C"

    gattr_col$[2,fnstr_pos("DVAR",attr_def_col$[0,0],5)]="FUNC_ID"
    gattr_col$[2,fnstr_pos("CTLW",attr_def_col$[0,0],5)]="100"
    gattr_col$[2,fnstr_pos("LABS",attr_def_col$[0,0],5)]=Translator!.getTranslation
:("EXM_FUNCTIONS.FUNC_ID","Func ID")

    gattr_col$[3,fnstr_pos("DVAR",attr_def_col$[0,0],5)]="FUNC_SRC"
    gattr_col$[3,fnstr_pos("CTLW",attr_def_col$[0,0],5)]="100"
    gattr_col$[3,fnstr_pos("LABS",attr_def_col$[0,0],5)]=Translator!.getTranslation
```

```
:("EXM_FUNCTIONS.FUNC_SRC","Source")

    gattr_col$[4,fnstr_pos("DVAR",attr_def_col$[0,0],5)]="FUNC_DESC"
    gattr_col$[4,fnstr_pos("CTLW",attr_def_col$[0,0],5)]="300"
    gattr_col$[4,fnstr_pos("LABS",attr_def_col$[0,0],5)]=Translator!.getTranslation
:("EXM_FUNCTIONS.FUNC_DESC","Description")

    for curr_attr=1 to def_cols
        gattr_col$[0,1]=gattr_col$[0,1]+pad(process_id$+"."+gattr_col$[curr_attr,fnstr_pos
:("DVAR",attr def col$[0,0],5)],40)
    next curr_attr
    gattr_disp_col$=gattr_col$[0,1]

    call stbl("+DIR_SYP")+"bam_grid_init.bbj",
:gui_dev,grdFuncs!,"COLH-AUTO-LINES-LIGHT-
CHECKS",0,attr_def_col$[all],gattr_disp_col$,gattr_col$[all]

    colEnable!=grdFuncs!.getBackColor()
    colDisable!=sysGUI!.makeColor(255,192,192)
    grdFuncs!.setColumnEditable(0,1)
    grdFuncs!.setEditClickCount(1)

    return

resize_form:rem --- Resize Form Contents

    grdFuncs!.setSize(frmCustom!.getWidth()-(grdFuncs!.getX()*2),frmCustom!.getHeight()-
(grdFuncs!.getY()
:+grdFuncs!.getX()))
    grdFuncs!.setFitToGrid(grdFuncs!.AUTO_RESIZE_LAST_COLUMN)

    return

get_form_coords:rem --- Get Form Location and Size

    call stbl("+DIR_SYP")
+"bac_winsize.bbj",pad(process_id$,20),"W","",frmCustom!,"READ",form_coords[all]
    return

save_form_coords:rem --- Save Form Location and Size

    call stbl("+DIR_SYP")
+"bac_winsize.bbj",pad(process_id$,20),"W","",frmCustom!,"SAVE",form_coords[all]
    return

disable_ctls:rem --- Disable Controls

    if able_ctls$<>"" call stbl("+DIR_SYP")+"bam_enable.bbj",
:gui_dev,"DISABLE","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return

enable_ctls:rem --- Enable Controls

    if able_ctls$<>"" call stbl("+DIR_SYP")+"bam_enable.bbj",
:gui_dev,"ENABLE","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return
```

```
update_ctls:rem --- Update Control Status on Activated Form

    call stbl("+DIR SYP")+"bam enable.bbj",
:gui_dev,"UPDATE-MDI","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return

error_proc:rem --- Error Processing Routine

    err_text$=""
    if tcb(2)=0 and tcb(5) then err_text$=pgm(tcb(5),tcb(13),err=*next)

    call stbl("+DIR_SYP")+"bac_error.bbj",pgm(-2),str(tcb(5)),str(err),err_text$,err_act$

    if pos("EXIT"=err_act$)<>0 goto exit_prog
    if pos("ESCAPE"=err_act$)<>0 seterr 0;setesc 0
    if pos("RETRY"=err_act$)<>0 retry

exit_prog:rem --- Exit Program

    if form_disp$="YES" gosub save_form_coords

    release
```

## exm_functions.arc

```
//#charset: windows-1252

// Barista Application Framework - ASCII Resource File
// EXM_FUNCTIONS - Display Barista Tool/Menu Functions
// Proprietary Information. BASIS International Ltd. All rights reserved.

VERSION "4.0"

WINDOW 1000 "" 10 40 0400 0102
BEGIN
    NAME "win_exm_functions"
    MANAGESYSCOLOR
    KEYBOARDNAVIGATION
    DIALOGBEHAVIOR
    EVENTMASK 1136656524
    INVISIBLE
    ENTERASTAB

    STATICTEXT 02001, "Description 1:", 56, 13, 71, 16
    BEGIN
        NOT OPAQUE
        JUSTIFICATION 32768
        NAME "txt_description_1"
    END

    INPUTE 03001, "", 130, 10, 240, 19
    BEGIN
        NAME "ine_description_1"
        CLIENTEDGE
        PASSENTER
        HIGHLIGHT
        PADCHARACTER 0
        MAXLENGTH 30
```

```
    END

    STATICTEXT 02002, "Description 2:", 56, 34, 71, 16
    BEGIN
        NOT OPAQUE
        JUSTIFICATION 32768
        NAME "txt description 2"
    END

    INPUTE 03002, "", 130, 31, 240, 19
    BEGIN
        NAME "ine_description_2"
        CLIENTEDGE
        PASSENTER
        HIGHLIGHT
        PADCHARACTER 0
        MAXLENGTH 30
    END

    STATICTEXT 02003, "Description 3:", 56, 55, 71, 16
    BEGIN
        NOT OPAQUE
        JUSTIFICATION 32768
        NAME "txt_description_3"
    END

    INPUTE 03003, "", 130, 52, 240, 19
    BEGIN
        NAME "ine_description_3"
        CLIENTEDGE
        PASSENTER
        HIGHLIGHT
        PADCHARACTER 0
        MAXLENGTH 30
    END
END
```

## cust_form.bbj

```
rem ' Customer master file maintenance (BBj GUI user interface)

dim customer$:"id:c(6),name:c(32),phone:c(24)"
pathname$="c:/samples/"
filename$ = "customer.dat"
customer = unt
open (customer,err=makefile)pathname$+filename$
goto init

makefile:
mkeyed pathname$+filename$,[0:1:6],0,64
open (customer)pathname$+filename$
while 1
   dread customer.id$,customer.name$,customer.phone$,err=eof
   write record(customer)customer$
   continue
eof:
   break
wend
data "BASIS","BASIS International Ltd.","+1.505.345.5232"
data "CHILE","Chile Company","+1.555.555.1212"

init:
sysgui = unt
open (sysgui)"X0"; rem ' ALIAS X0 SYSGUI

sysgui! = bbjapi().getSysGui()
cust = resopen(pathname$+"cust_form.arc")
window! = sysgui!.createTopLevelWindow(cust,101)
id! = window!.getControl(102)
name! = window!.getControl(104)
phone! = window!.getControl(106)
update! = window!.getControl(201)
delete! = window!.getControl(202)
clear! = window!.getControl(203)
resclose (cust)
id!.setText("BASIS")
id!.focus()
gosub fetch

id!.setCallback(id!.ON_EDIT_MODIFY,"toggle")
id!.setCallback(id!.ON_LOST_FOCUS,"fetch")
update!.setCallback(update!.ON_BUTTON_PUSH,"update")
delete!.setCallback(delete!.ON_BUTTON_PUSH,"remove")
clear!.setCallback(clear!.ON_BUTTON_PUSH,"clear")
window!.setCallback(window!.ON_CLOSE,"eoj")

process_events

eoj:
release

toggle:
    id$ = cvs(id!.getText(),7)
    update!.setEnabled(len(id$))
    delete!.setEnabled(len(id$))
return
```

```
fetch:
    id$ = pad(cvs(id!.getText(),7),6)
    if customer.id$ = id$ then return
    dim customer$:fattr(customer$)
    let customer.id$ = id$
    read record(customer,key=customer.id$,dom=notfound)customer$
notfound:
    gosub display
return

update:
    customer.id$ = ctrl(sysgui,102,1)
    customer.name$ = ctrl(sysgui,104,1)
    customer.phone$ = ctrl(sysgui,106,1)
    write record (customer)customer$
    i = msgbox("Customer "+customer.id$+" updated.",0,"Updated")
    gosub clear
return

remove:
    remove (customer,key=customer.id$,dom=nodelete)
    i = msgbox("Customer "+customer.id$+" deleted.",0,"Deleted")
nodelete:
    gosub clear
return

clear:
    dim customer$:fattr(customer$)
    gosub display
    gosub toggle
    id!.focus()
return

display:
    id!.setText(cvs(customer.id$,3))
    name!.setText(cvs(customer.name$,3))
    phone!.setText(cvs(customer.phone$,3))
return
```

## cust_form.arc

```
//#charset: windows-1252

VERSION "4.0"

WINDOW 101 "Customers" 100 100 280 170
  BEGIN
        KEYBOARDNAVIGATION
        EVENTMASK 12582912
        NAME "Customer"
        STATICTEXT 101, "ID:", 10, 10, 80, 30
        BEGIN
        JUSTIFICATION 32768
        NAME "ID Label"
        END
        EDIT 102, "", 100, 10, 70, 30
        BEGIN
        NAME "ID"
        CLIENTEDGE
        END
        STATICTEXT 103, "Name:", 10, 50, 80, 30
        BEGIN
        JUSTIFICATION 32768
        NAME "Name Label"
        END
        EDIT 104, "", 100, 50, 170, 30
        BEGIN
        NAME "Name"
        CLIENTEDGE
        END
        STATICTEXT 105, "Phone:", 10, 90, 80, 30
        BEGIN
        JUSTIFICATION 32768
        NAME "Phone Label"
        END
        EDIT 106, "", 100, 90, 170, 30
        BEGIN
        NAME "Phone"
        CLIENTEDGE
        END
        BUTTON 201, "Update", 10, 130, 80, 30
        BEGIN
        NAME "Update"
        END
        BUTTON 202, "Delete", 100, 130, 80, 30
        BEGIN
        NAME "Delete"
        END
        BUTTON 203, "Clear", 190, 130, 80, 30
```

```
        BEGIN
        NAME "Clear"
        END
  END
```

## cust_form_bar.bbj

```
rem ' Customer master file maintenance (BBj GUI user interface)
rem ' Modified to integrate Barista Menu/Toolbar


rem --- This code block added for Barista version ----------------------------------------------
---------

    if stbl("+USE_SETERR")="YES" seterr error_proc

rem --- Functions

    def fnstr_pos(tmp0$,tmp1$,tmp0)=int((pos(tmp0$=tmp1$,tmp0)+tmp0-1)/tmp0)

rem --- Misc Definitions

    process_id$="CUST_EXTERNAL"
    task_val$=process_id$+str(dec(info(3,0)):"0000")

rem --- Get Menu Information Passed by Barista

    dim sysinfo$:stbl("+SYSINFO_TPL")
    sysinfo$=stbl("+SYSINFO")

rem --- Open Tables

    num_files=1
    dim
open_tables$[1:num_files],open_opts$[1:num_files],open_chans$[1:num_files],open_tpls$[1:num_files
]

    open_tables$[1]=stbl("+GUI_DEVICE"),open_opts$[1]="O"
    gosub open_tables
    gui_dev=num(open_chans$[1])

rem --- Get Objects and Set Up Group Namespace

    sysGUI!=BBjAPI().getSysGui()

    grpSpace!=BBjAPI().getGroupNamespace()
    grpSpace!.setValue("+build_task","ON");rem --- Starts the Barista MDI Progress Meter

rem --- Set Function Enable/Disable String

    call stbl("+DIR_SYP")+"bac_mdi_ctls.bbj"
    call stbl("+DIR_SYP")+"bam_enable.bbj",gui_dev,"CREATE","IDLE-INIT-
MDI",rd_sys_mdi_ctls$,able_map$,"",0

rem --- sys_act_ctls$ contains the active functions for this task.
rem     To activate all available functions, use:

rem    sys_act_ctls$=rd_sys_mdi_ctls$

rem --- To activate only selected functions, use the following. The function values are contained
rem     in variables called sys_<func>_ctls$ as seen below.
```

```
    sys_act_ctls$=
:       rd_sys_new_ctls$+
:       rd_sys_sav_ctls$+
:       rd_sys_del_ctls$+
:       rd_sys_fst_ctls$+
:       rd_sys_prv_ctls$+
:       rd_sys_nxt_ctls$+
:       rd_sys_lst_ctls$

    call stbl("+DIR_SYP")+"bam_enable.bbj",gui_dev,"ACTIVE-
UPDATE","MDI",sys_act_ctls$,able_map$,"",0

rem --- Get Barista Attribute Arrays

    call stbl("+DIR_SYP")+"bam_attr_init.bbj",attr_def_tbl$[all],attr_def_col$[all],"ALL"

rem --- End added code block -------------------------------------------------------------------
---------

rem --- from original version
dim customer$:"id:c(6),name:c(32),phone:c(24)"
pathname$="c:/samples/"
filename$ = "customer.dat"
customer = unt
open (customer,err=makefile)pathname$+filename$
goto init

makefile:
mkeyed pathname$+filename$,[0:1:6],0,64
open (customer)pathname$+filename$
while 1
   dread customer.id$,customer.name$,customer.phone$,err=eof
   write record(customer)customer$
   continue
eof:
   break
wend
data "BASIS","BASIS International Ltd.","+1.505.345.5232"
data "CHILE","Chile Company","+1.555.555.1212"

rem --- modified from original version
rem     use Barista routines to open/display for using modified .arc (removed update/delete/clear
buttons
rem     create control objects and setup/display first record same as original version

init:

    gosub disp_form
    gosub get_form_coords

    able_ctls$=sys_act_ctls$
    gosub  enable_ctls

    frmCustom!.setVisible(1)
    frmCustom!.focus()
    gosub set_active_task
    frmCustom!.setCursor(0)

    grpSpace!.setValue("+build_task","OFF");rem --- Stops the Barista MDI Progress Meter
```

```
   id! = frmCustom!.getControl(102)
    name! = frmCustom!.getControl(104)
    phone! = frmCustom!.getControl(106)

    id!.setText("BASIS")
    id!.focus()
    gosub fetch

event_ctl:rem --- Event Control
rem --- no longer need events for the update/delete/clear buttons that were removed from the form
rem             callbacks for id! control remain as they were in original version

    frmCustom!.setCallback(frmCustom!.ON_CLOSE,"exit_prog")
    frmCustom!.setCallback(frmCustom!.ON_ACTIVATE,"set_active_task")
    frmCustom!.setCallback(frmCustom!.ON_RESTORE,"set_active_task")

  id!.setCallback(id!.ON_EDIT_MODIFY,"toggle")
  id!.setCallback(id!.ON_LOST_FOCUS,"fetch")

    grpSpace!.setCallbackForVariable("+"+task_val$+".func","get_active_func")

    process_events

toggle:
rem --- this routine used to enable/disable the update/delete buttons
rem --- now enables/disables Barista delete and save buttons/menu items

    id$ = cvs(id!.getText(),7)
    able_ctls$=seval("rd_sys_sav_ctls$+rd_sys_del_ctls$")
  if len(id$)
       gosub enable_ctls
  else
       gosub disable_ctls
  endif
return

fetch:
rem --- this routine is from the original version
rem --- modified to add fetch_nav entry point for use with Barista navigation buttons

    id$ = pad(cvs(id!.getText(),7),6)
    if customer.id$ = id$ then return
fetch_nav:
    dim customer$:fattr(customer$)
    let customer.id$ = id$
    read record(customer,key=customer.id$,dom=notfound)customer$
notfound:
    gosub display
return

update:
rem --- this routine used to run when the 'update' button was pressed
rem --- now triggered from the Barista 'save' toolbar button or menu item

    customer.id$ = ctrl(sysgui,102,1)
    customer.name$ = ctrl(sysgui,104,1)
    customer.phone$ = ctrl(sysgui,106,1)
    write record (customer)customer$
```

```
    i = msgbox("Customer "+customer.id$+" updated.",0,"Updated")
    gosub clear
return

remove:
rem --- this routine used to run when the 'delete' button was pressed
rem --- now triggered from the Barista 'delete' toolbar button or menu item

    remove (customer,key=customer.id$,dom=nodelete)
    i = msgbox("Customer "+customer.id$+" deleted.",0,"Deleted")
nodelete:
    gosub clear
return

clear:
rem --- this routine used to run when the 'clear' button was pressed
rem --- now triggered from the Barista 'new' toolbar button or menu item

    dim customer$:fattr(customer$)
    gosub display
    able_ctls$=seval("rd_sys_sav_ctls$+rd_sys_del_ctls$")
       gosub disable_ctls
    id!.focus()
return

display:
rem --- this routine is the from the original version
rem --- modified to enable the Barista 'delete' and 'save' buttons/menu items

    id!.setText(cvs(customer.id$,3))
    name!.setText(cvs(customer.name$,3))
    phone!.setText(cvs(customer.phone$,3))
       able_ctls$=seval("rd_sys_sav_ctls$+rd_sys_del_ctls$")
       gosub enable_ctls
return

rem --- added for Barista version--------------------------------------------------------------------
--

route_func:rem --- Get Function From Form Based Toolbar/menubar
rem --- callback to this routine was registered in earlier call to bam_controls.bbj
rem --- this routine runs when clicking a menu item or toolbutton on the form

    evtQueue!=sysGUI!.getLastEvent()
    tempCtl!=evtQueue!.getControl()
    active_func$=tempCtl!.getUserData()
    func_source$="FRM"

    goto route_active_func

get_active_func:rem --- Get Function Value From MDI/Namespace
rem --- this routine runs when clicking on a menu item or toolbutton on the MDI

    active_func$=""
    active_func$=sysGUI!.getLastEvent().getNewValue(err=*return)
    if active_func$="" return
    grpSpace!.setValue("+"+task_val$+".func","")
    func_source$="MDI"
```

```
route_active_func:rem Route Function to Subroutine
rem --- use this routine to respond to events triggered by pressing toolbuttons or selecting menu
items

    func_str$="EXT;NEW;SAV;DEL;FST;PRV;NXT;LST;";rem --- Add the required three character
function to
:activate it.
    switch fnstr_pos(active_func$,func_str$,4)
        case fnstr_pos("EXT",func_str$,4)
            gosub exit_prog
        break
        case fnstr_pos("NEW",func_str$,4)
                gosub clear
        break
        case fnstr_pos("SAV",func_str$,4)
                gosub update
        break
        case fnstr_pos("DEL",func_str$,4)
                gosub remove
        break
        case fnstr_pos("FST",func_str$,4)
                id$=keyf(customer,err=*break)
                gosub fetch_nav
        break
        case fnstr_pos("PRV",func_str$,4)
                read record (customer,key=customer.id$,dir=0,err=*break)
                id$=keyp(customer,err=*break)
                gosub fetch_nav
        break
        case fnstr_pos("NXT",func_str$,4)
                id$=key(customer,err=*break)
                gosub fetch_nav
        break
        case fnstr_pos("LST",func_str$,4)
                id$=keyl(customer,err=*next)
                gosub fetch_nav
        break
        case default
        break
    swend

    return

set_active_task:rem --- Set Barista MDI Focus to This Form

    grpSpace!.setValue("+active_task",task_val$)
    gosub update_ctls
    frmCustom!.focus()

    return

update_ctls:rem --- Barista Update Control Status on Activated Form

    call stbl("+DIR_SYP")+"bam_enable.bbj",
:gui_dev,"UPDATE-MDI","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return
```

```
disp_form:rem --- Barista routine to Display Custom Form

    dim ctl_misc$[20];rem --- DIMs Template Needed by bam_controls

    dim attr_tbl$[len(attr_def_tbl$[0,0])/5];rem --- DIMs Template Needed by bam_controls
        attr tbl$[0]=attr def tbl$[0,0]
        attr_tbl$[fnstr_pos("OPTS",attr_def_tbl$[0,0],5)]="T";rem Adds Toolbar, menubar and
statusbar to
:form
        attr_tbl$[fnstr_pos("WINT",attr_def_tbl$[0,0],5)]=sysinfo.task_desc$;rem Incoming menu
item desc.
:for form titlebar

    dim attr_col$[len(attr_def_col$[0,0])/5,10];rem --- DIMs Template Needed by bam_controls
    dim rec_data$[len(attr_def_col$[0,0])/5,10];rem --- DIMs Template Needed by bam_controls

    form_id$="c:/samples/cust_form_bar.arc"; rem --- arc modified from original version
                                             rem --- no longer contains delete/update/clear
buttons

    if pos(stbl("+FORM_NAV_BAR",err=*next)="TB")<>0 nav_bar$=";NAV"

    form_type$="FMSTD";rem --- Type of form - FMSTD - Linear Form
    bar_type$="MAINT_FORM"; rem --- Type of navigation bar on form - MAINT_FORM - Maintenance
form

    call stbl("+DIR_SYP")+"bam_controls.bbj",
:gui_dev,sysGUI!,form_id$,form_coords[all],frmCustom!,"DISPLAY-"+form_type$+"-"+bar_type$,
:open_chan$[all],ctl_misc$[all],attr_tbl$[all],x$[all],x!

    form_disp$="YES"

    return

get_form_coords:rem --- Barista Get Form Location and Size

    call stbl("+DIR_SYP")
+"bac_winsize.bbj",pad(process_id$,20),"W","",frmCustom!,"READ",form_coords[all]
    return

save_form_coords:rem --- Barista Save Form Location and Size

    call stbl("+DIR_SYP")
+"bac_winsize.bbj",pad(process_id$,20),"W","",frmCustom!,"SAVE",form_coords[all]
    return

disable_ctls:rem --- Barista Disable Controls

    if able_ctls$<>"" call stbl("+DIR_SYP")+"bam_enable.bbj",
:gui_dev,"DISABLE","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return

enable_ctls:rem --- Barista Enable Controls

    if able_ctls$<>"" call stbl("+DIR_SYP")+"bam_enable.bbj",
:gui_dev,"ENABLE","MDI"+nav_bar$,able_ctls$,able_map$,able_ctx$,able_ctx,
```

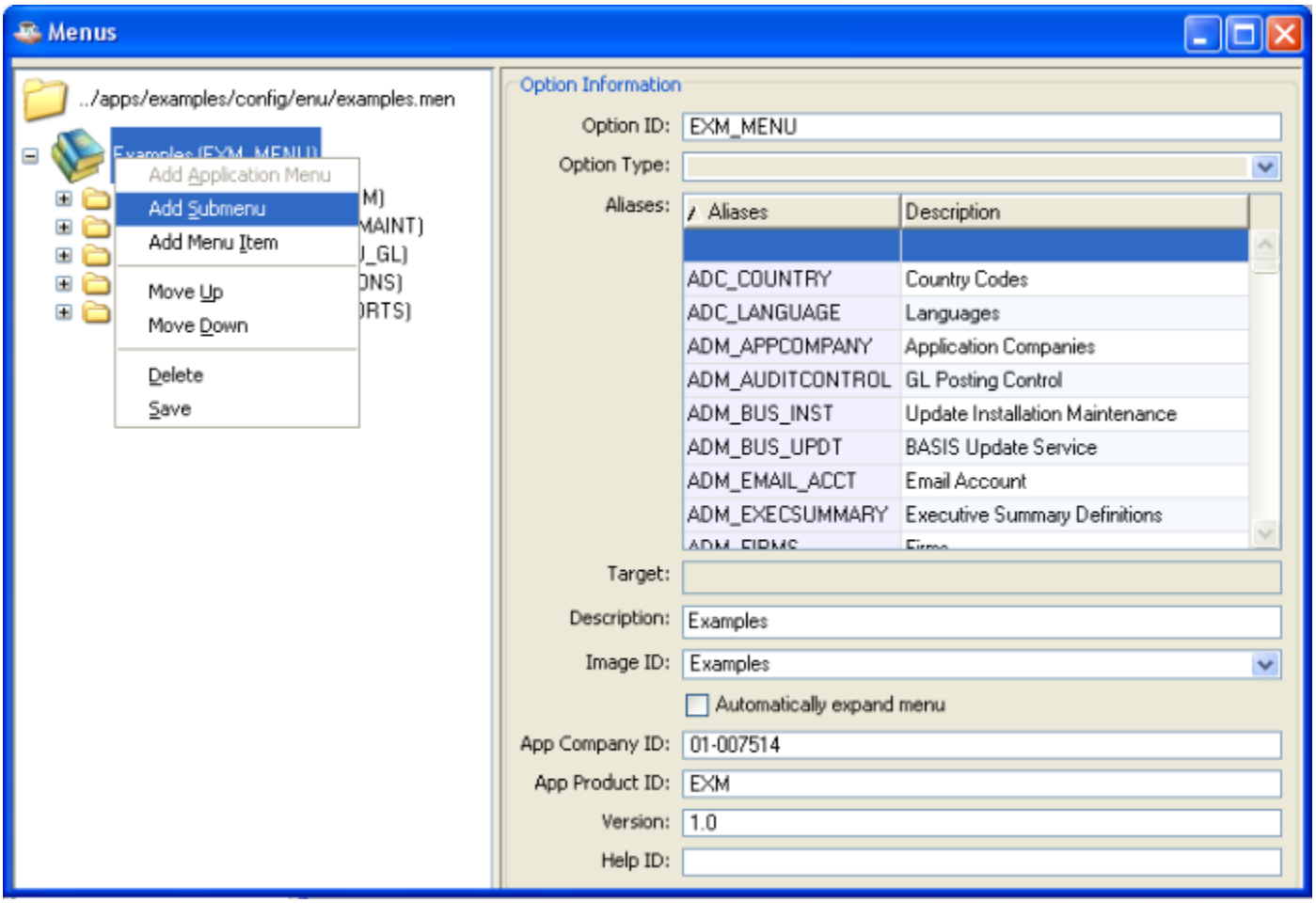
```
:sysGUI!,rec_data$[all],"",attr_def_tbl$[all],attr_def_col$[all],attr_tbl$[all],attr_col$[all],fr
mCustom!
    return

open_tables:rem --- standard Barista Open Tables routine

    call stbl("+DIR_SYP")+"bac_open_tables.bbj",
:0,0,open_tables$[all],open_opts$[all],open_chans$[all],open_tpls$[all],table_chans$[all],0,open_
status$

    if open_status$<>""
        msg_id$="ENTRY_OPEN_ERROR"
        dim msg tokens$[1]
            msg_tokens$[1]=open_status$
        gosub disp_message
        goto exit_prog
    endif

    return

disp_message:rem --- Barista Display Message Dialog

    call dir_syp$+"bac_message.bbj",msg_id$,msg_tokens$[all],msg_opt$,table_chans$[all]
    return

error_proc:rem --- Barista Error Processing Routine

    err_text$=""
    if tcb(2)=0 and tcb(5) then err_text$=pgm(tcb(5),tcb(13),err=*next)

    call stbl("+DIR_SYP")+"bac_error.bbj",pgm(-2),str(tcb(5)),str(err),err_text$,err_act$

    if pos("EXIT"=err_act$)<>0 goto exit_prog
    if pos("ESCAPE"=err_act$)<>0 seterr 0;setesc 0
    if pos("RETRY"=err_act$)<>0 retry

exit_prog:rem --- Barista Exit Program

    if form_disp$="YES" gosub save_form_coords

    release
```

***cust_form_bar.arc***

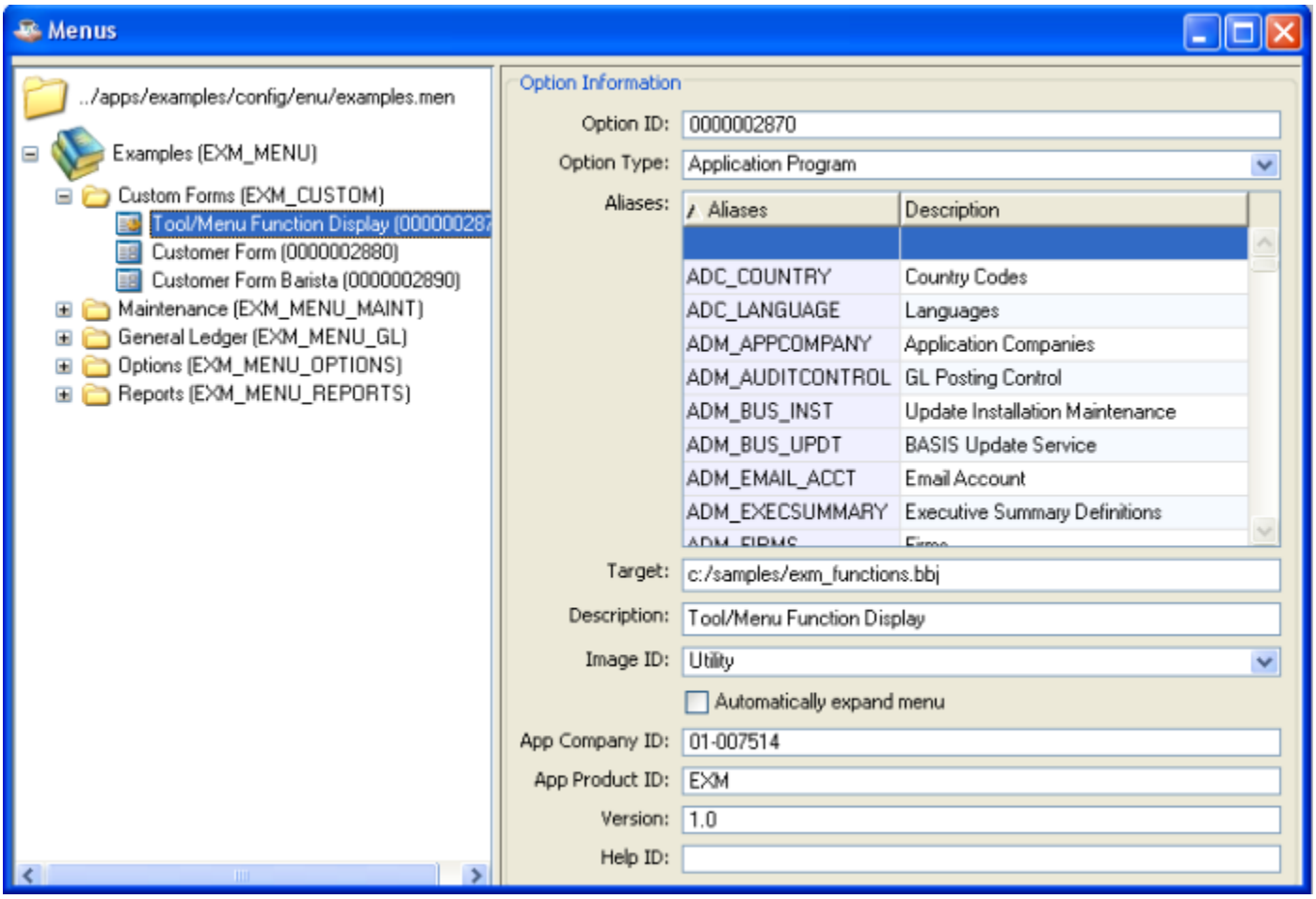
# Appendix B

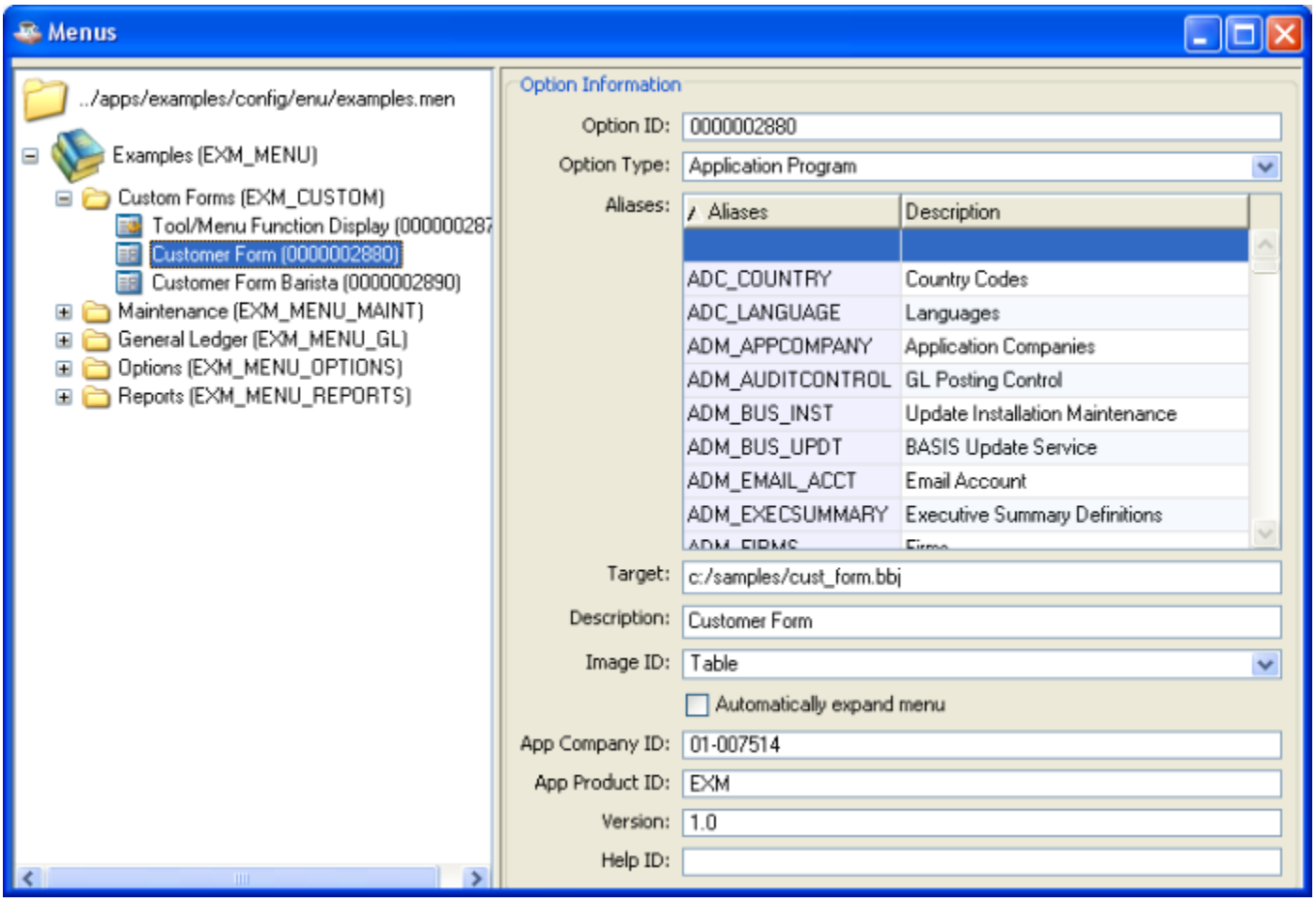
⇧back to top

1. Access the Examples menu from Barista Administration => Menus.
2. Select the `examples.men` file from the grid listing installed menus, and click [Maintain].
3. Right-click on the Examples main menu, and select [Add Submenu].




4. Create a new menu with Option ID `EXM_CUSTOM` and Description `Custom Forms`.
5. Add the `exm_functions.bbj`, `cust_form.bbj`, and `cust_form_bar.bbj` programs as new menu items:


Menus
../apps/examples/config/enu/examples.men
Examples (EXM_MENU)
Custom Forms (EXM_CUSTOM)
Tool/Menu Function Display (00000028
Customer Form (0000002880)
Customer Form Barista (0000002890)
Maintenance (EXM_MENU_MAINT)
General Ledger (EXM_MENU_GL)
Options (EXM_MENU_OPTIONS)
Reports (EXM_MENU_REPORTS)
Option Information
Option ID: 0000002870
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_BUS_UPDT
BASIS Update Service
ADM_EMAIL_ACCT
Email Account
ADM_EXECSUMMARY
Executive Summary Definitions
Target: c:/samples/exm_functions.bbj
Description: Tool/Menu Function Display
Image ID: Utility
Automatically expand menu
App Company ID: 01-007514
App Product ID: EXM
Version: 1.0
Help ID:


Menus
../apps/examples/config/enu/examples.men
Examples (EXM_MENU)
Custom Forms (EXM_CUSTOM)
Tool/Menu Function Display (000000287
Customer Form (0000002880)
Customer Form Barista (0000002890)
Maintenance (EXM_MENU_MAINT)
General Ledger (EXM_MENU_GL)
Options (EXM_MENU_OPTIONS)
Reports (EXM_MENU_REPORTS)
Option Information
Option ID: 0000002880
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_BUS_UPDT
BASIS Update Service
ADM_EMAIL_ACCT
Email Account
ADM_EXECSUMMARY
Executive Summary Definitions
Target: c:/samples/cust_form.bbj
Description: Customer Form
Image ID: Table
Automatically expand menu
App Company ID: 01-007514
App Product ID: EXM
Version: 1.0
Help ID:

# Appendix C

### Barista Control Numbering

⇧back to top

The following `barista.cfg` entries specify the control numbering used by Barista. Bolded entries specify starting numbers for ranges. All others are specific numbers for one control per form.

**+TEXT_CTL=2000**
**+ENTRY_CTL=3000**
**+DISPLAY_CTL=4000**
**+IMAGE_CTL=6000**
+STATUS_CTL=10000
+PROGRESS_CTL=10010
+WINDOW_RES=1000
+TAB_CTL=1050
**+BLOCK_CTL=19000**
**+BLOCK_TEXT_CTL=19500**
**+CHILD_WIN=1100**
+GRID_CTL=5000
+GRID_LIST_CTL=7000
**+BUTTON_CTL=20000**
+BACKGROUND_CTL=19990
+CUSTOM_CTL=25000
+NAVBAR_CTL=1500
+DETAIL_GRID_WIN=1109

Barista toolbar and menubar functions use the 31000 range. The actual control IDs (and accelerator key codes) for these functions can be found in the barista.ini file.

Text, entry, display, and button control numbers correspond to the entry order sequence of the column, and the window (or child window) on which it's located. The ranges can be explained by using the following format, where `w` equals the window ID and `nn` equals the entry sequence number:

```
+CHILD_WIN=110w
+TEXT_CTL=2wnn
+ENTRY_CTL=3wnn
+DISPLAY_CTL=4wnn
+IMAGE_CTL=6wnn
+BUTTON_CTL=20wnn
     Find = 20wnn
```

```
Calendar = 21wnn
Drilldown = 22wnn
Link = 23wnn
Multi-Language = 24wnn
```

For example, if the first column on a non-tabbed form had a static text label, entry control, validation display and and an inquiry button, the following controls would be used for that column:

Static text: 200**1**
Entry control: 300**1**
Display control: 400**1**
Inquiry "Find" button: 2000**1**

The second column on the form would use:If

Static text: 200**2**
Entry control: 300**2**
Display control: 400**2**
Inquiry "Find" button: 2000**2**

If the third column were on the first tab (child window), the following numbers would be used:

Child window: 110**1**
Static text: 2**1**0**1**
Entry control: 3**1**0**1**
Display control: 4**1**0**1**
Inquiry "Find" button: 20**1**0**1**