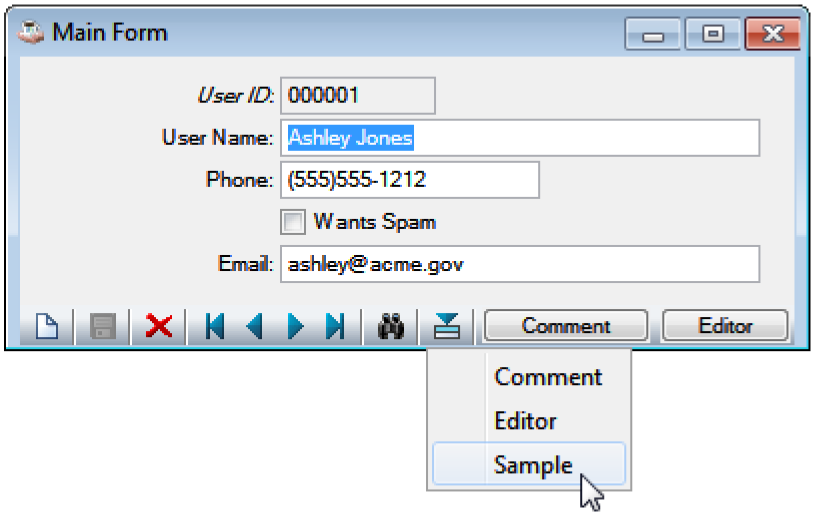
Barista Tutorial 3: Define Callpoints

This tutorial builds on Barista Tutorial 1 - Build a Simple Form.

What We Are Going To Build

We'll extend the form we created in the first tutorial by adding our own custom code and options buttons.



Main Form

User ID: 000001

User Name: Ashley Jones

Phone: (555)555-1212

Wants Spam: ☐

Email: ashley@acme.gov

Callpoints

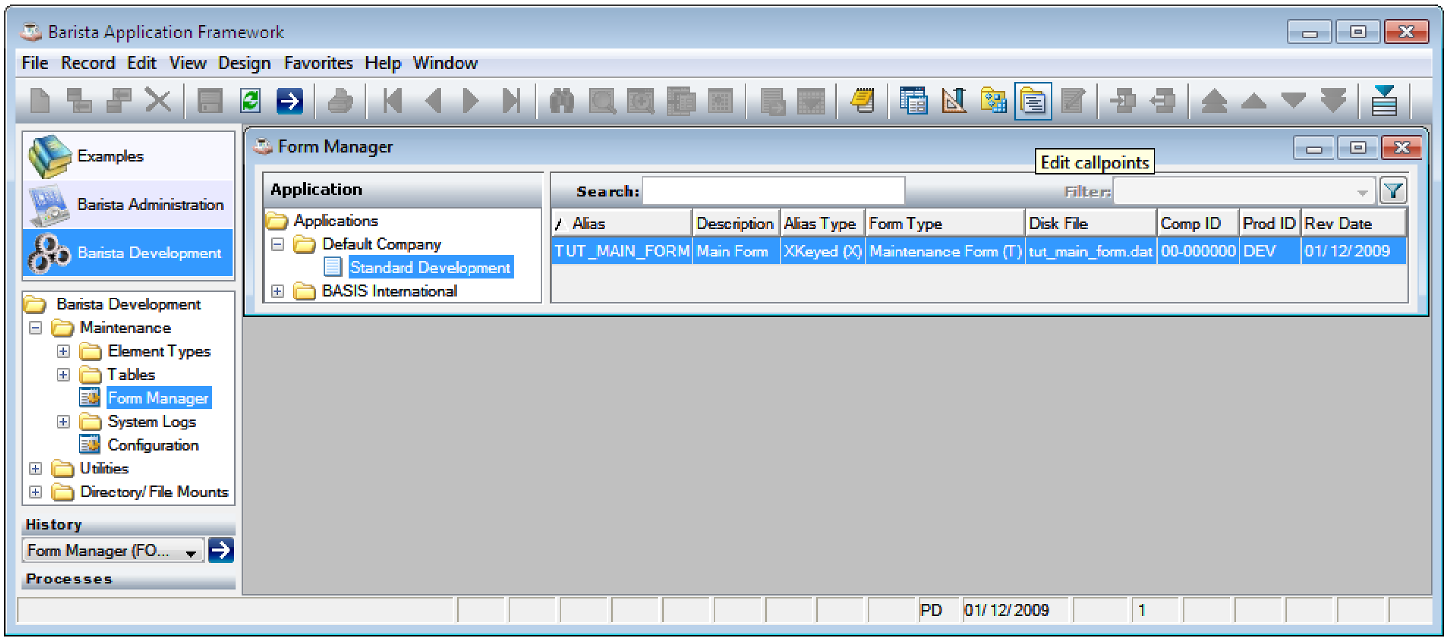
Barista is designed to implement standard business application behavior without the need to write custom code. There are times, however, when a developer needs to add specialized business logic, or start a background batch process, or perform some other operation that Barista does not handle automatically. Barista **Callpoints** address this requirement.

Options can be added to the button, as individual buttons on the bottom toolbar, or both. Developer-defined callpoint code determines what a given option will do.

See Getting Started for more detailed background information about Callpoints and Options.

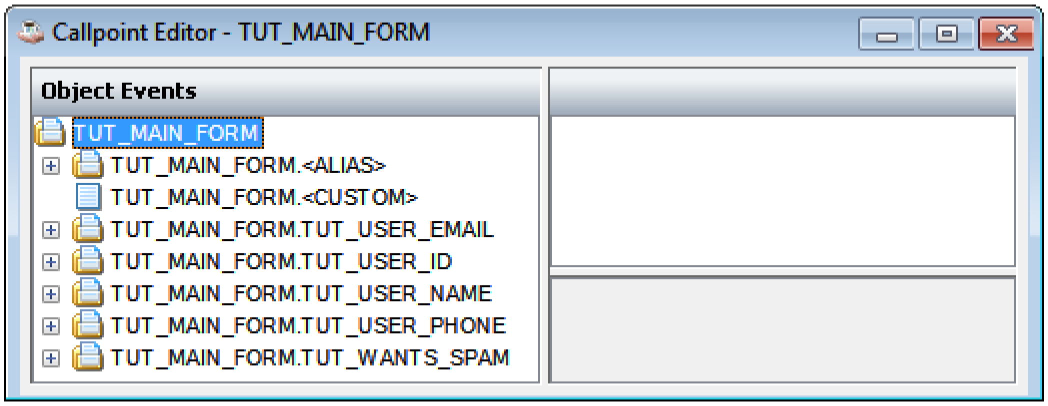
Bring up the Callpoint Editor

Open the Form Manager, select TUT_MAIN_FORM, and click or press F2 to load the Callpoint Editor.



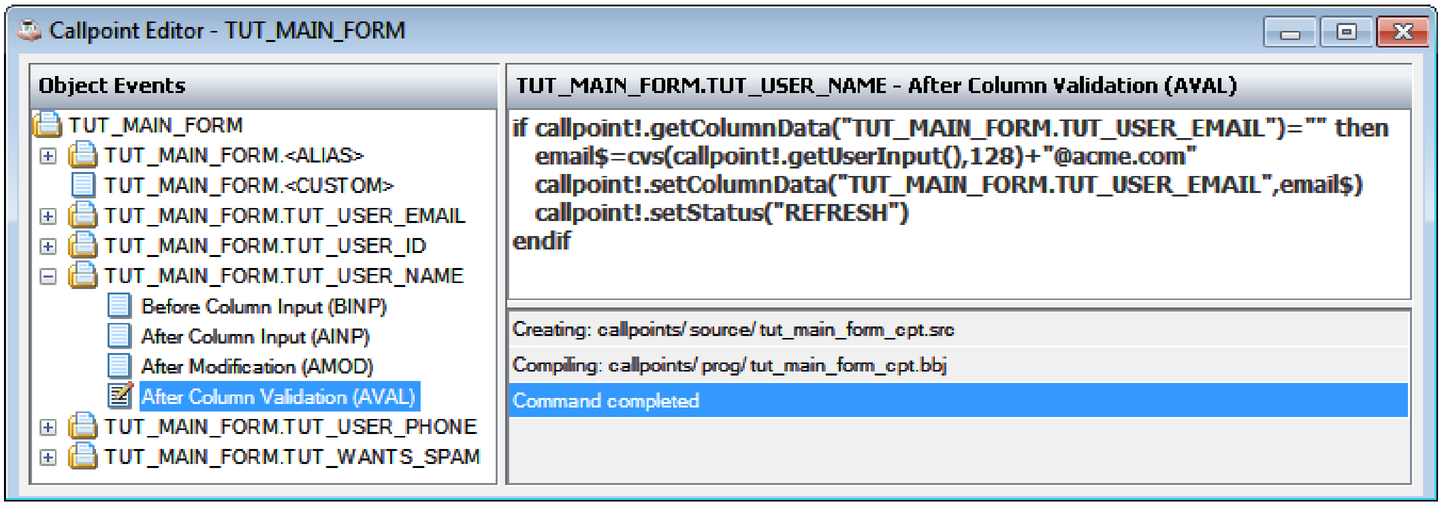
Alias	Description	Alias Type	Form Type	Disk File	Comp ID	Prod ID	Rev Date
TUT_MAIN_FORM	Main Form	XKeyed (X)	Maintenance Form (T)	tut_main_form.dat	00-000000	DEV	01/12/2009

Callpoint Editor


Callpoint Editor - TUT_MAIN_FORM
Object Events
TUT_MAIN_FORM
TUT_MAIN_FORM.<ALIAS>
TUT_MAIN_FORM.<CUSTOM>
TUT_MAIN_FORM.TUT_USER_EMAIL
TUT_MAIN_FORM.TUT_USER_ID
TUT_MAIN_FORM.TUT_USER_NAME
TUT_MAIN_FORM.TUT_USER_PHONE
TUT_MAIN_FORM.TUT_WANTS_SPAM

The callpoint editor consists of three panels. Along the left is a tree which contains the form, its columns and their associated callpoints. At a glance one can tell if there is code in a given callpoint based on the icon (= empty, = non-empty). The upper-right displays the code for the selected callpoint. The lower-left panel displays compilation and status. When a callpoint is selected, right clicking the upper-right panel (or clicking) displays a list of code templates which can serve as a quick reference and save you some typing.

We will use the After Column Validation callpoint to generate a default email address derived from the name.


Callpoint Editor - TUT_MAIN_FORM
Object Events
TUT_MAIN_FORM
TUT_MAIN_FORM.<ALIAS>
TUT_MAIN_FORM.<CUSTOM>
TUT_MAIN_FORM.TUT_USER_EMAIL
TUT_MAIN_FORM.TUT_USER_ID
TUT_MAIN_FORM.TUT_USER_NAME
Before Column Input (BINP)
After Column Input (AINP)
After Modification (AMOD)
After Column Validation (AVAL)
TUT_MAIN_FORM.TUT_USER_PHONE
TUT_MAIN_FORM.TUT_WANTS_SPAM
TUT_MAIN_FORM.TUT_USER_NAME - After Column Validation (AVAL)

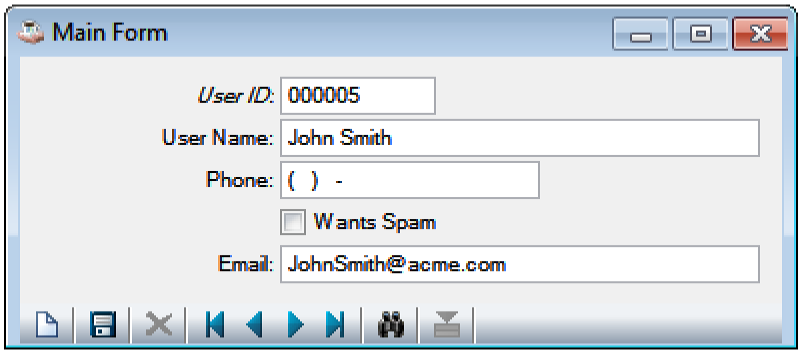
```
if callpoint!.getColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL")="" then
   email$=cvs(callpoint!.getUserInput(),128)+"@acme.com"
   callpoint!.setColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL",email$)
   callpoint!.setStatus("REFRESH")
endif
```

```
Creating: callpoints/source/tut_main_form_cpt.src
Compiling: callpoints/prog/tut_main_form_cpt.bbj
Command completed
```

- Bring up the Callpoint Editor for TUT_MAIN_FORM.
- Select After Column Validation (AVAL) under TUT_MAIN_FORM.TUT_USER_NAME.
- Add code (note we don't change an existing email value):

```
if callpoint!.getColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL")="" then
   email$=cvs(callpoint!.getUserInput(),128)+"@acme.com"
   callpoint!.setColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL",email$)
   callpoint!.setStatus("REFRESH")
endif
```

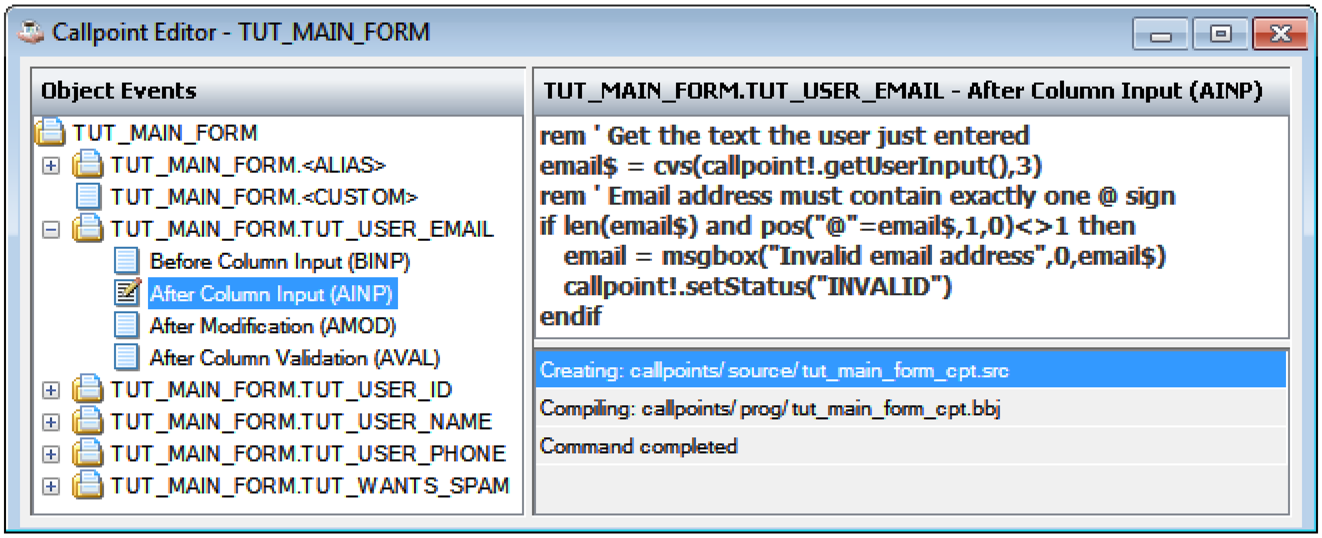
- Click or press F5 to run the form.
- Add a new record and note the email is automatically populated as you tab out of the User Name field:



User ID:	000005
User Name:	John Smith
Phone:	( ) -
	☐ Wants Spam
Email:	JohnSmith@acme.com

Add AINP Callpoint Code for TUT_USER_EMAIL

Barista provides several validation mechanisms, such as enforcing a minimum length, or performing a foreign key lookup. There are times you need to perform more complicated validation. The **After Column Input** callpoint is called after the user has attempted to leave the callpoint, but before validation. Here we will check for an obviously invalid email address.


Callpoint Editor - TUT_MAIN_FORM
Object Events
TUT_MAIN_FORM
TUT_MAIN_FORM.<ALIAS>
TUT_MAIN_FORM.<CUSTOM>
TUT_MAIN_FORM.TUT_USER_EMAIL
Before Column Input (BINP)
After Column Input (AINP)
After Modification (AMOD)
After Column Validation (AVAL)
TUT_MAIN_FORM.TUT_USER_ID
TUT_MAIN_FORM.TUT_USER_NAME
TUT_MAIN_FORM.TUT_USER_PHONE
TUT_MAIN_FORM.TUT_WANTS_SPAM
TUT_MAIN_FORM.TUT_USER_EMAIL - After Column Input (AINP)

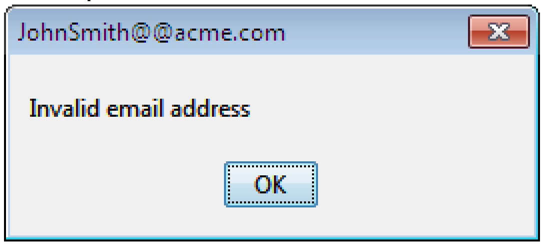
```
rem ' Get the text the user just entered
email$ = cvs(callpoint!.getUserInput(),3)
rem ' Email address must contain exactly one @ sign
if len(email$) and pos("@"=email$,1,0)<>1 then
   email = msgbox("Invalid email address",0,email$)
   callpoint!.setStatus("INVALID")
endif
```

```
Creating: callpoints/source/tut_main_form_cpt.src
Compiling: callpoints/prog/tut_main_form_cpt.bbj
Command completed
```

- Select After Column Input (AINP) under TUT_MAIN_FORM.TUT_USER_EMAIL.
- Add code:

```
rem ' Get the text the user just entered
email$ = cvs(callpoint!.getUserInput(),3)
rem ' Email address must contain exactly one @ sign
if len(email$) and pos("@"=email$,1,0)<>1 then
   email = msgbox("Invalid email address",0,email$)
   callpoint!.setStatus("INVALID")
endif
```

- Click or press F5 to run the form.
- Attempt to enter an email address with more than one @ sign:



Options Entry Forms

There are times you would like the user to enter data for immediate processing, as opposed to writing it to a database table. In the next example we will simply write a time-stamped email address and comment to a text file. This same basic approach can be used to send email, write reports, etc.

We'll start by defining a Table Alias that will be used to generate an Options Entry Form.

Open Tables Maintenance and define the following alias:

Location	Field	Value
Header	Table Alias	TUT_DATA_ENTRY
	Description	Comment Form
	Alias Type	Options Entry
Detail	Element Type	Data Element
	EXM_TEXT_30	EMAIL
	EXM_TEXT_30	COMMENT

If you didn't install the Barista Examples package, you'll need to first define EXM_TEXT_30 as a 30-character text field in Element Types Maintenance.

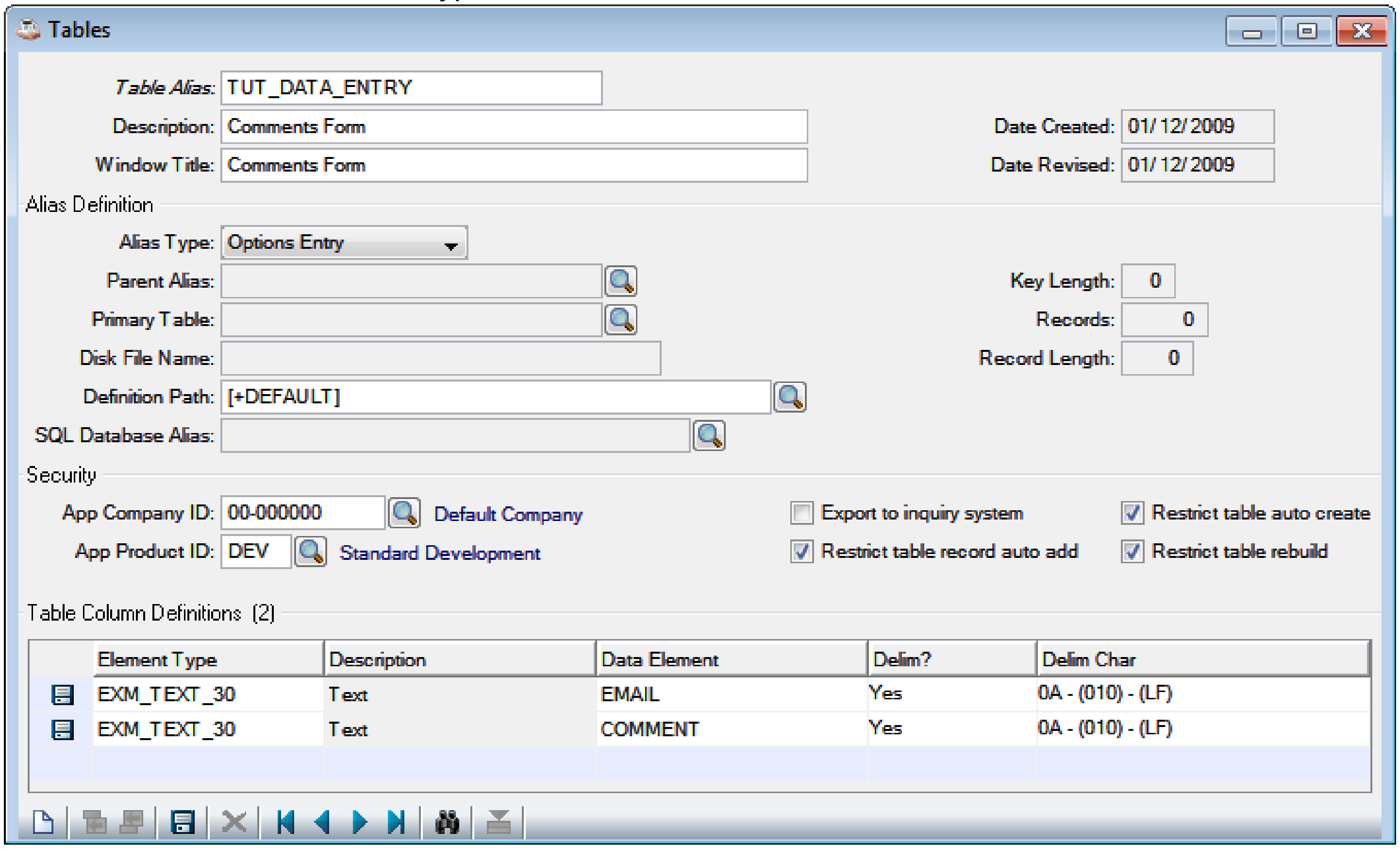

Tables

Table Alias: TUT_DATA_ENTRY
Description: Comments Form
Window Title: Comments Form
Date Created: 01/12/2009
Date Revised: 01/12/2009

Alias Definition

Alias Type: Options Entry
Parent Alias:
Primary Table:
Disk File Name:
Definition Path: [+DEFAULT]
SQL Database Alias:
Key Length: 0
Records: 0
Record Length: 0

Security

App Company ID: 00-000000 Default Company
App Product ID: DEV Standard Development

- [ ] Export to inquiry system
- [x] Restrict table record auto add
- [x] Restrict table auto create
- [x] Restrict table rebuild

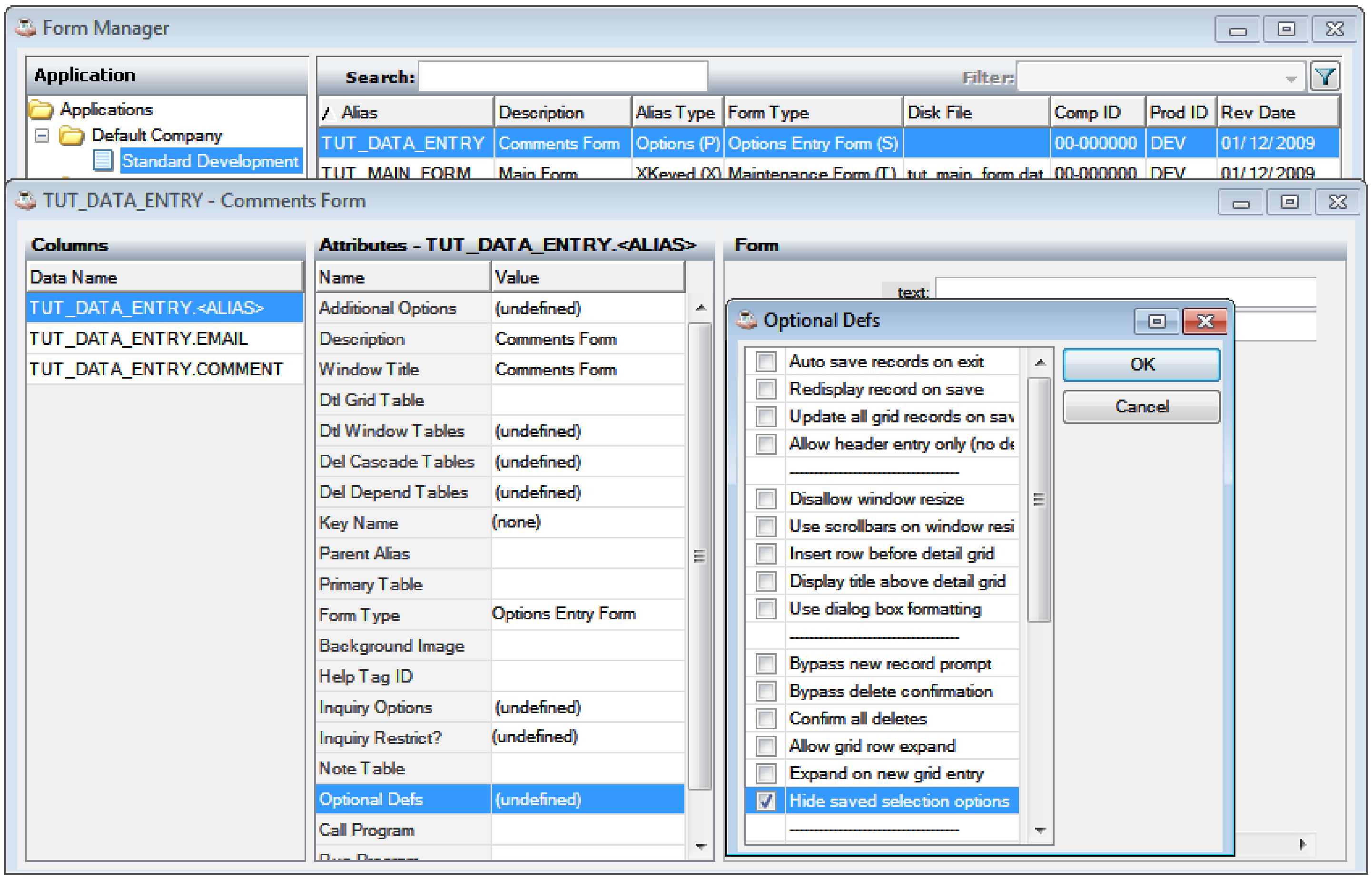
Table Column Definitions (2)

Element Type	Description	Data Element	Delim?	Delim Char
EXM_TEXT_30	Text	EMAIL	Yes	0A - (010) - (LF)
EXM_TEXT_30	Text	COMMENT	Yes	0A - (010) - (LF)

Click or press [Ctrl]+S to save this record, then close Tables Maintenance.

Switch back to the Form Manager, click or press [Alt]+F5 to update the list, and double-click on your newly defined alias to load it in the Form Designer.

- Check **Hide saved selection options** in Optional Defs for the Alias



Form Manager

Application

- Applications
  - Default Company
    - Standard Development

Search: Filter:

Alias	Description	Alias Type	Form Type	Disk File	Comp ID	Prod ID	Rev Date
TUT_DATA_ENTRY	Comments Form	Options (P)	Options Entry Form (S)		00-000000	DEV	01/12/2009
TUT_MAIN_FORM	Main Form	XKeyed (X)	Maintenance Form (T)	tut_main_form.dat	00-000000	DEV	01/12/2009

TUT_DATA_ENTRY - Comments Form

Columns

Data Name
TUT_DATA_ENTRY.<ALIAS>
TUT_DATA_ENTRY.EMAIL
TUT_DATA_ENTRY.COMMENT

Attributes - TUT_DATA_ENTRY.<ALIAS>

Name	Value
Additional Options	(undefined)
Description	Comments Form
Window Title	Comments Form
Dtl Grid Table	
Dtl Window Tables	(undefined)
Del Cascade Tables	(undefined)
Del Depend Tables	(undefined)
Key Name	(none)
Parent Alias	
Primary Table	
Form Type	Options Entry Form
Background Image	
Help Tag ID	
Inquiry Options	(undefined)
Inquiry Restrict?	(undefined)
Note Table	
Optional Defs	(undefined)
Call Program	

Form

text:

Optional Defs

☐	Auto save records on exit
☐	Redisplay record on save
☐	Update all grid records on sav
☐	Allow header entry only (no de

☐	Disallow window resize
☐	Use scrollbars on window resi
☐	Insert row before detail grid
☐	Display title above detail grid
☐	Use dialog box formatting

☐	Bypass new record prompt
☐	Bypass delete confirmation
☐	Confirm all deletes
☐	Allow grid row expand
☐	Expand on new grid entry
☑	Hide saved selection options

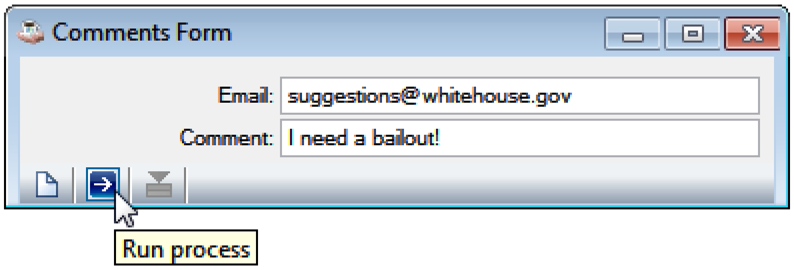
OK

Cancel

- Set the Control Label for the EMAIL field to "Email"
- Set the Control Label for the COMMENT field to "Message"
- Set the **Run Program** for the alias to ../apps/default/comment.src and save the following program to apps/default/comment.src under the BASIS home directory. On Windows, the full name would typically be C:\Program Files\basis\apps\default\comment.src:

```
email$ = option!.getOptionData("EMAIL")
comment$ = option!.getOptionData("COMMENT")
timestamp$ = new java.util.Date().toString()
message$ = timestamp$+$09$+email$+$09$+comment$
filename$ = "../apps/default/comment.txt"
string filename$,err=*next
comment = unt
open (comment,MODE="O_APPEND")filename$
print (comment)message$
close (comment)
release
```

Click or press F5 to run the form, type any values into the entry fields, then click or press F5 again to pass control to the overlay program.



Comments Form

Email: suggestions@whitehouse.gov

Comment: I need a bailout!

The log file looks like this:

```
        C:\Program Files\basis\apps\default>type comment.txt
        Mon Jan 12 21:00:21 PST 2009    suggestions@whitehouse.gov      I need a bailout!
```

Adding an Option Button to display our DATA_ENTRY form
Now we'll add a button to TUT_MAIN_FORM to display our new form.

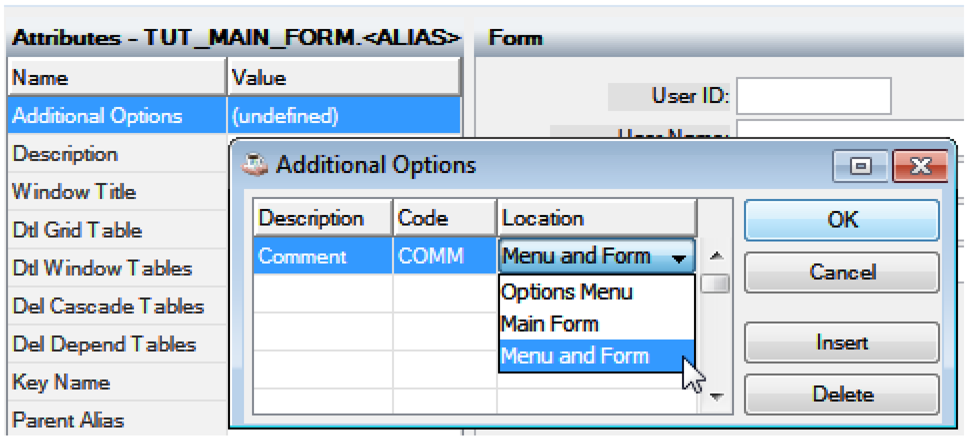
Load **TUT_MAIN_FORM** in the Forms Designer.

Double click the **Additional Options** attribute for the alias and enter:

Description	**Code**	**Location**
Comment	COMM*	Menu And Form**

* Any code will do, up to four characters.

** We can choose to add the option as a button on the form, as an item in the menu, or both.



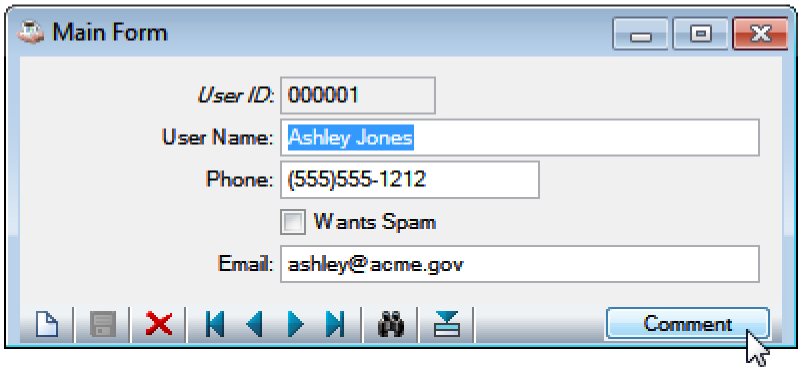
Attributes - TUT_MAIN_FORM.<ALIAS>

Name	Value
Additional Options	(undefined)
Description	
Window Title	
Dd Grid Table	
Dd Window Tables	
Del Cascade Tables	
Del Depend Tables	
Key Name	
Parent Alias	

Additional Options

Description	Code	Location
Comment	COMM	Menu and Form

Click or press F5 to run the form. It has a new button, but the button doesn't do anything yet.



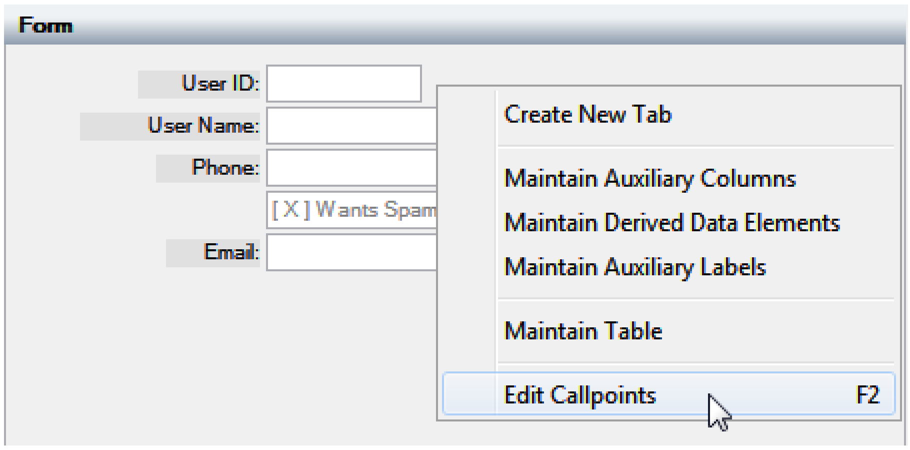
Main Form

User ID: 000001
User Name: Ashley Jones
Phone: (555)555-1212
Wants Spam
Email: ashley@acme.gov

Next, we'll add code to invoke the TUT_DATA_ENTRY form when the user clicks the Comment button.

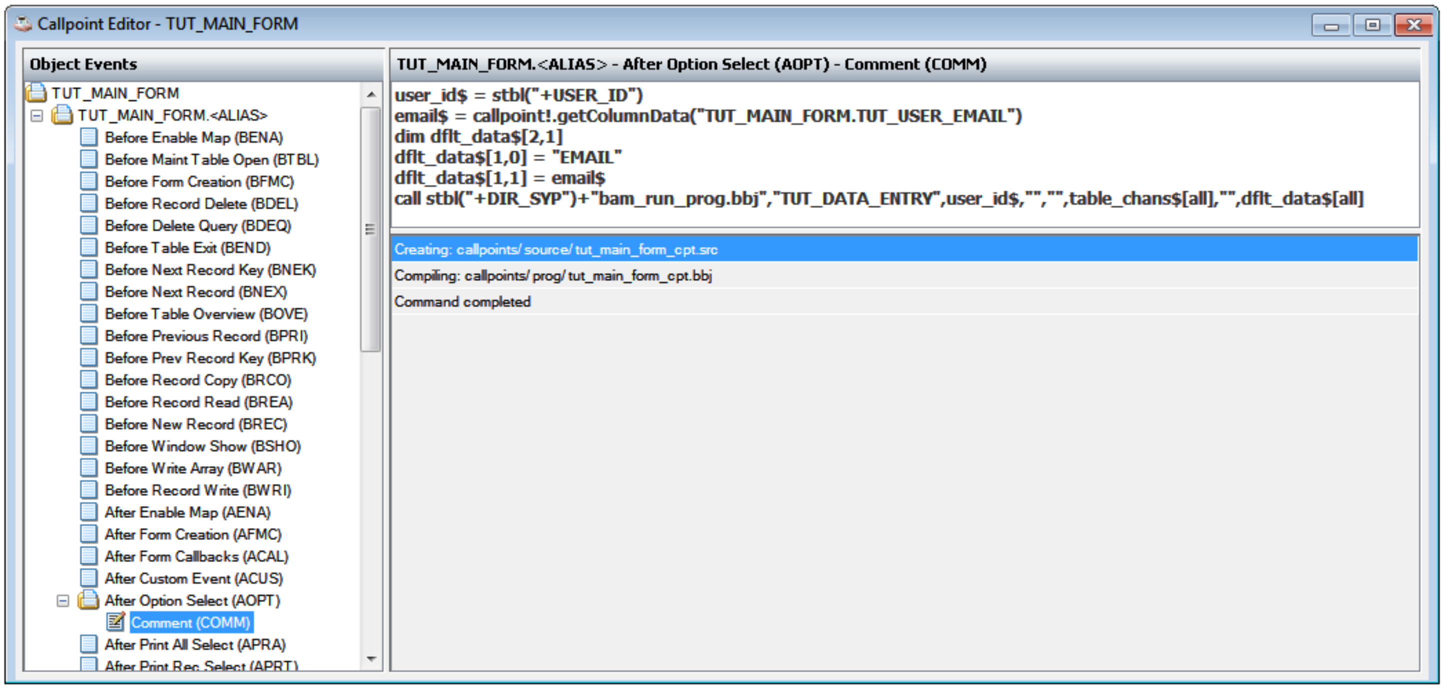
Add Callpoint code for the Option Button

Still in the Form Designer, right-click on the form and bring up the Callpoint Editor.



Under TUT_MAIN_FORMS.<ALIAS>, select After Option Select (AOPT), then select Comment (COMM) and paste in the following callpoint code:

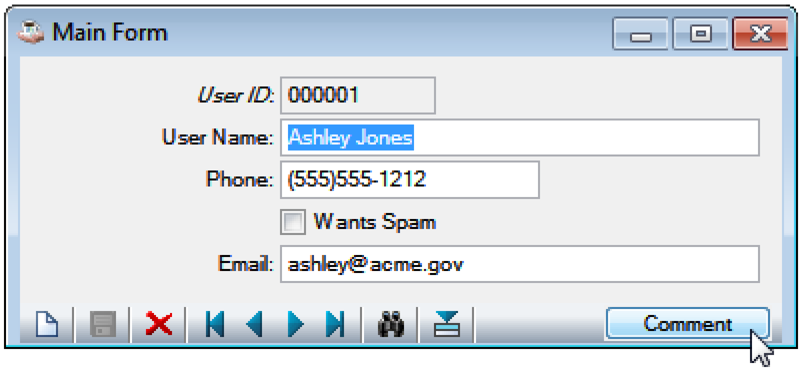
```
        user_id$ = stbl("+USER_ID")
        email$ = callpoint!.getColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL")
        dim dflt_data$[2,1]
        dflt_data$[1,0] = "EMAIL"
        dflt_data$[1,1] = email$
        call stbl("+DIR_SYP")
        +"bam_run_prog.bbj","TUT_DATA_ENTRY",user_id$,"","",table_chans$[all],"",dflt_data$
        [all]
```


Callpoint Editor - TUT_MAIN_FORM
Object Events
TUT_MAIN_FORM
TUT_MAIN_FORM.<ALIAS>
Before Enable Map (BENA)
Before Maint Table Open (BTBL)
Before Form Creation (BFMC)
Before Record Delete (BDEL)
Before Delete Query (BDEQ)
Before Table Exit (BEND)
Before Next Record Key (BNEK)
Before Next Record (BNEX)
Before Table Overview (BOVE)
Before Previous Record (BPRI)
Before Prev Record Key (BPRK)
Before Record Copy (BRCO)
Before Record Read (BREA)
Before New Record (BREC)
Before Window Show (BSHO)
Before Write Array (BWAR)
Before Record Write (BWRI)
After Enable Map (AENA)
After Form Creation (AFMC)
After Form Callbacks (ACAL)
After Custom Event (ACUS)
After Option Select (AOPT)
Comment (COMM)
After Print All Select (APRA)
After Print Rec Select (APRT)
TUT_MAIN_FORM.<ALIAS> - After Option Select (AOPT) - Comment (COMM)

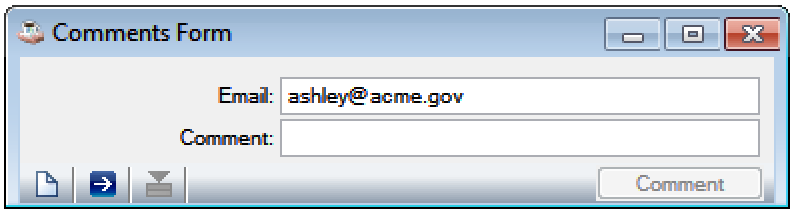
```
user_id$ = stbl("+USER_ID")
email$ = callpoint!.getColumnData("TUT_MAIN_FORM.TUT_USER_EMAIL")
dim dflt_data$[2,1]
dflt_data$[1,0] = "EMAIL"
dflt_data$[1,1] = email$
call stbl("+DIR_SYP")+"bam_run_prog.bbj","TUT_DATA_ENTRY",user_id$,"","",table_chans$[all],"",dflt_data$[all]
```

```
Creating: callpoints/source/tut_main_form_cpt.src
Compiling: callpoints/prog/tut_main_form_cpt.bbj
Command completed
```

Click or type [Ctrl]+B to rebuild the callpoints for the form, then click or press F5 to run the form. Now when you select a record and click the button, the overlay program comes up:


Main Form

User ID: 000001
User Name: Ashley Jones
Phone: (555)555-1212
Wants Spam
Email: ashley@acme.gov


Comments Form

Email: ashley@acme.gov
Comment:

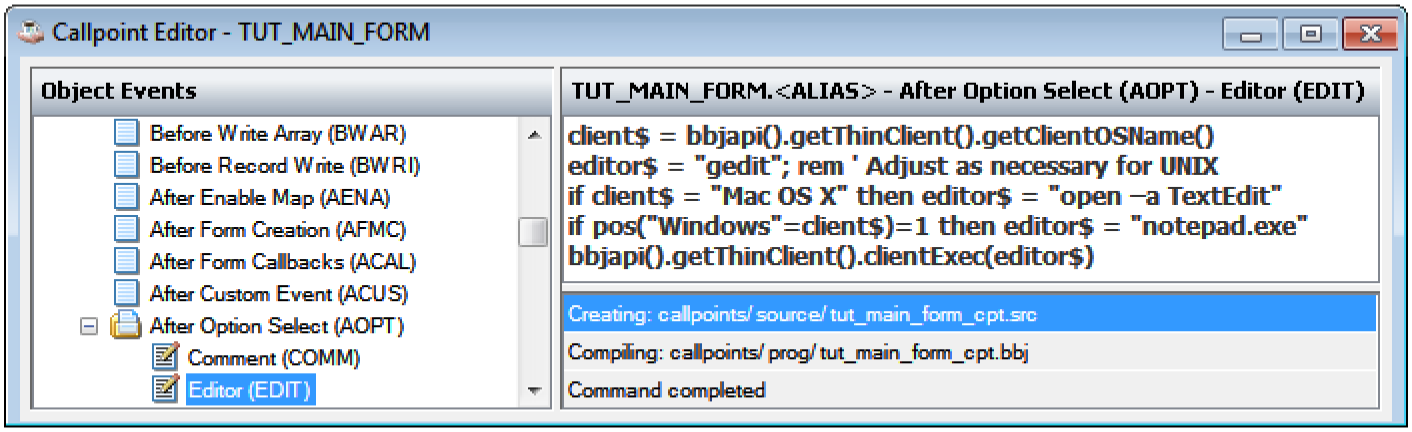
By default, options buttons are only enabled when a record is active. This can be changed via the **Additional options always enabled** setting in **Optional Defs**.

Add an editor option via the **Add Options** attribute in the Form Designer for TUT_MAIN_FORM.

Description	Code	Location
Editor	EDIT	Menu and Form

Add callpoint code for **Editor (EDIT):**

```
client$ = bbjapi().getThinClient().getClientOSName()
editor$ = "gedit"; rem ' Adjust as necessary for UNIX
if client$ = "Mac OS X" then editor$ = "open -a TextEdit"
if pos("Windows"=client$)=1 then editor$ = "notepad.exe"
bbjapi().getThinClient().clientExec(editor$)
```


Callpoint Editor - TUT_MAIN_FORM

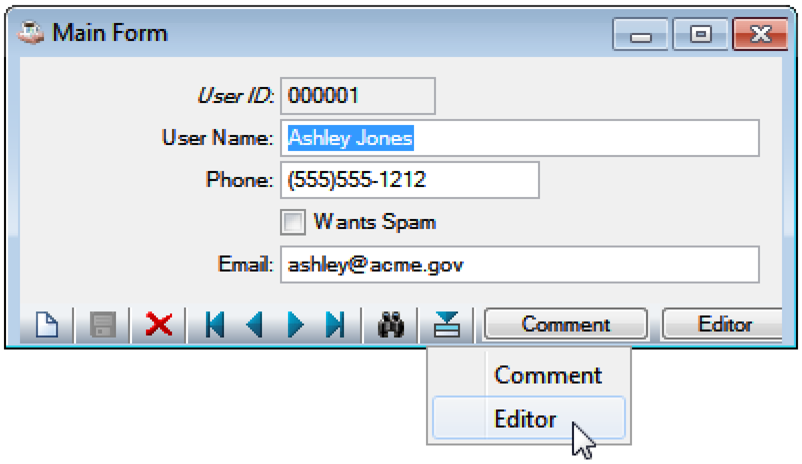
Object Events
Before Write Array (BWAR)
Before Record Write (BWRI)
After Enable Map (AENA)
After Form Creation (AFMC)
After Form Callbacks (ACAL)
After Custom Event (ACUS)
After Option Select (AOPT)
Comment (COMM)
Editor (EDIT)

TUT_MAIN_FORM.<ALIAS> - After Option Select (AOPT) - Editor (EDIT)

```
client$ = bbjapi().getThinClient().getClientOSName()
editor$ = "gedit"; rem ' Adjust as necessary for UNIX
if client$ = "Mac OS X" then editor$ = "open -a TextEdit"
if pos("Windows"=client$)=1 then editor$ = "notepad.exe"
bbjapi().getThinClient().clientExec(editor$)
```

```
Creating: callpoints/source/tut_main_form_cpt.src
Compiling: callpoints/prog/tut_main_form_cpt.bbj
Command completed
```

Click ➔ or press F5 to run the form:



User ID: 000001
User Name: Ashley Jones
Phone: (555)555-1212
Wants Spam
Email: ashley@acme.gov

Save the following program as apps/default/sample.src:

```
        sysgui = unt
        open (sysgui)"X0"
        sysgui! = bbjapi().getSysGui()
        window! = sysgui!.addWindow(100,100,200,200,"Sample")
        window!.setCallback(window!.ON_CLOSE,"eoj")
        button! = window!.addButton(101,50,50,100,30,"Click",$$)
        button!.setCallback(button!.ON_BUTTON_PUSH,"click")
        process_events
        click:
        i=msgbox("Click")
        return
        eoj:
        release
```

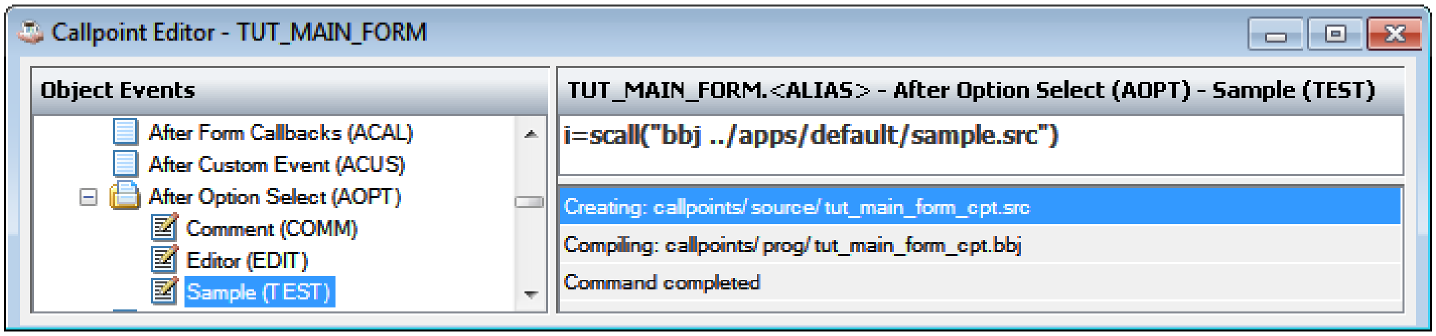
Calling a custom program via an Option Button

Add another option via the **Add Options** attribute in the Form Designer for TUT_MAIN_FORM

Description	Code	Location
Sample	TEST	Menu and Form

Add callpoint code for Sample (TEST):

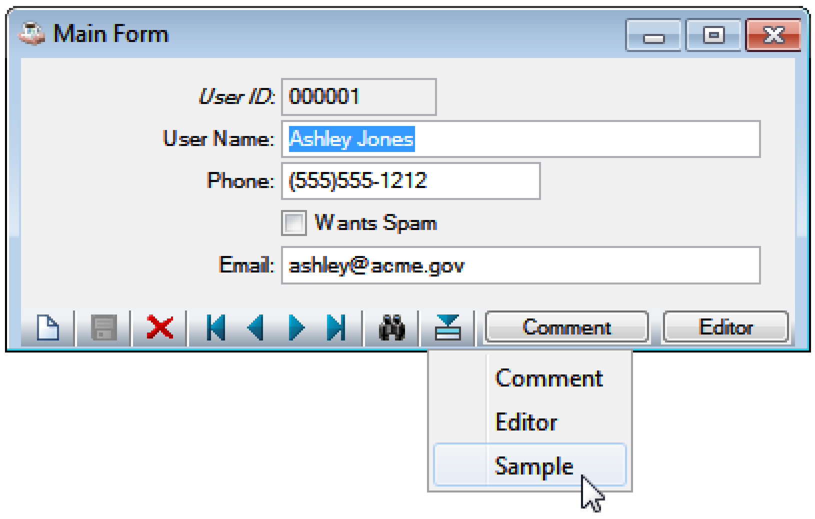
```
        i=scall("bbj ../apps/default/sample.src")
```


Callpoint Editor - TUT_MAIN_FORM
Object Events
After Form Callbacks (ACAL)
After Custom Event (ACUS)
After Option Select (AOPT)
Comment (COMM)
Editor (EDIT)
Sample (TEST)
TUT_MAIN_FORM.<ALIAS> - After Option Select (AOPT) - Sample (TEST)

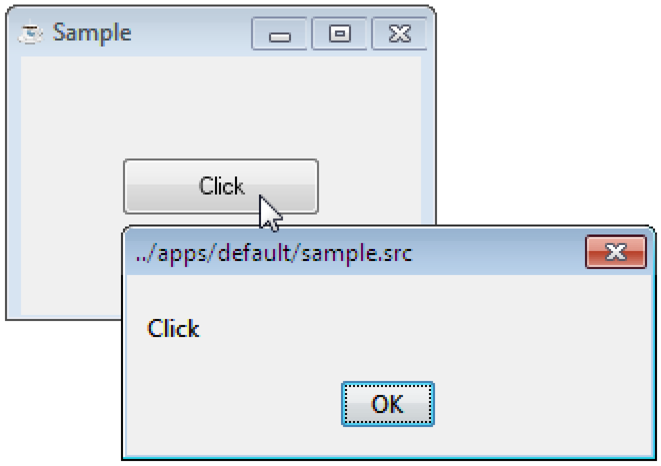
```
i=scall("bbj ../apps/default/sample.src")
```

```
Creating: callpoints/source/tut_main_form_cpt.src
Compiling: callpoints/prog/tut_main_form_cpt.bbj
Command completed
```

Click or press F5 to run the form:



User ID:	000001
User Name:	Ashley Jones
Phone:	(555)555-1212
☐	Wants Spam
Email:	ashley@acme.gov



../apps/default/sample.src

Click

Barista automatically takes care of the basics of building standard data-entry forms. Callpoints take over from there to enable the developer to customize Barista applications as necessary.

Want more? Try the Barista Tutorial 2 - Customize a Form, or the Administration Tutorial, which shows how to deal with projects, add menu items, users, internationalization.