UI BUILDER™ FOR ACCESS – STARTER EDITION VERSION 5

Application Guide

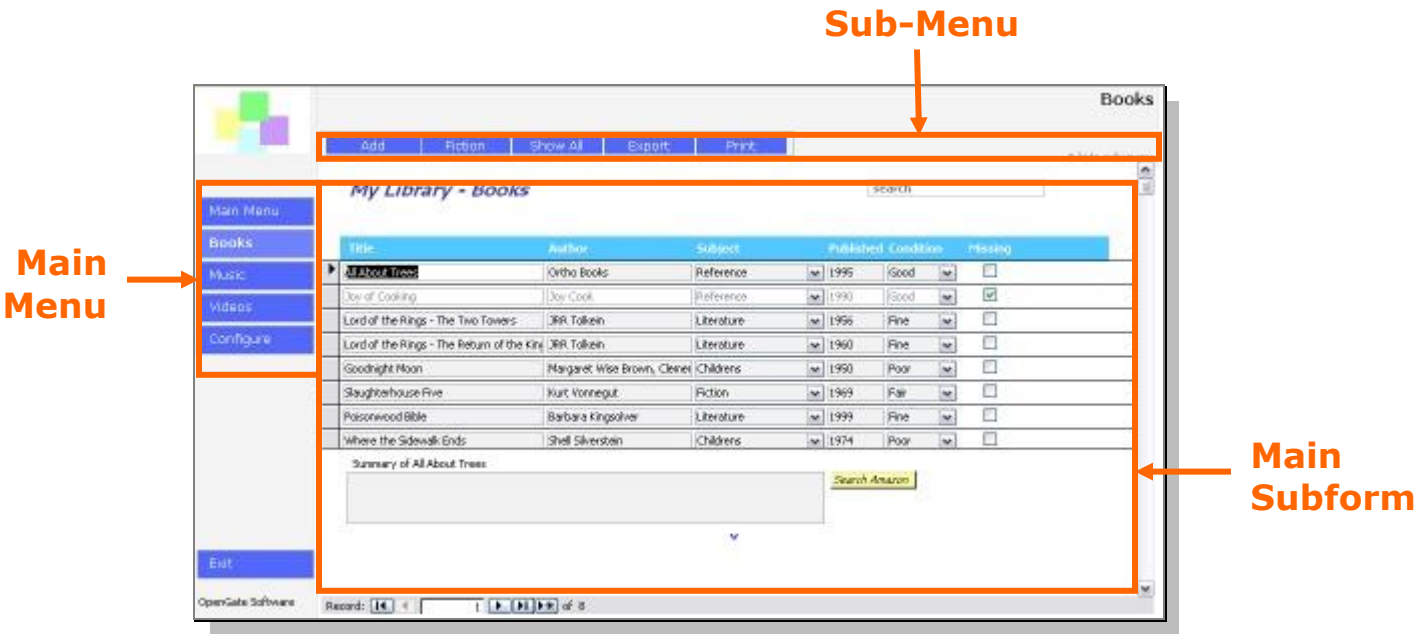
Version 1.14.2018

TABLE OF CONTENTS

1 General Concepts and Resources

This section describes the general concepts used in this document and when working with the UI Builder™ application.

1.1 USER INTERFACE



As shown above, the Main Menu, at left, is a set of up to five menu buttons displayed to a user at all times. The Sub-Menu, shown at the top, is a set of up to five sub-menu buttons that can be shown or hidden by the user, and are configurable for each selected Main Menu button. For example, you may decide to have no sub-menu buttons for the "Music" Main Menu option, two sub-menu buttons for the "Videos" Main Menu option, and five sub-menu buttons for the "Books" Main Menu option. Buttons can be configured to perform distinct actions that you specify. Finally, the Main Subform, shown at the center of the screen above, is where you display your application forms to the user.

1.2 RESOURCES

In addition to this document, you will be able to access help information from the Administrator and Sub-Menu Administrator forms by pressing F3 when you place the mouse cursor in a specific field. Additionally, there is an online demonstration available at: http://www.opengatesw.net

2 Migrating Your Application

2.1 MIGRATION OPTIONS

If you are integrating your own application into the UI Builder framework, you have two methods to chose from:

1. Import your database application forms, queries, tables, macros, and code modules into the UI Builder database file.
2. Import the UI Builder database file forms, queries, tables, macros, and code modules.

Method (1) above is best suited for situations where your database application does not have references to ActiveX or other dynamic link libraries (DLLs) beyond the Microsoft® Access™ default references. If you are unsure about whether this is true for your application or not, it is recommended you use method (1). Method (2) is generally easier if you have many references to ActiveX or other dynamic link libraries (DLLs) beyond the Microsoft Access default references, or if you are unfamiliar with how to import forms, queries, tables, macros, and code modules into a database.

2.2 MIGRATION STEPS

2.2.1 IMPORTING YOUR FORMS, QUERIES, TABLES, MACROS, AND CODE MODULES

To import your forms, queries, tables, macros, and code modules into the UI Builder file, follow these steps:

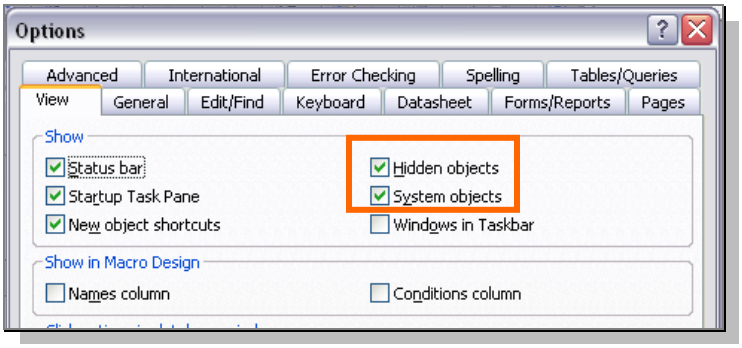
1. Create a copy of the UI Builder database file so that you can create other database applications using UI Builder in the future. You can then rename one of the copies to the name of your database application to begin importing your forms, queries, tables, macros, and code modules into the UI Builder framework.
2. Select "Migrate an Application" from the opening menu screen.
3. In the dialog that follows, please read the notification and select "Next."
4. In the Import Objects dialog, select all forms, queries, tables, macros, and code modules you want to migrate.
5. If you have an relationships or import/export specifications, be sure to select "Options>>" at the bottom right of the Import Objects dialog and select the corresponding checkboxes.
6. Select **OK**.
7. You are now ready to proceed to the next step in Section 2.3.

2.2.2 Importing UI Builder Into your Database

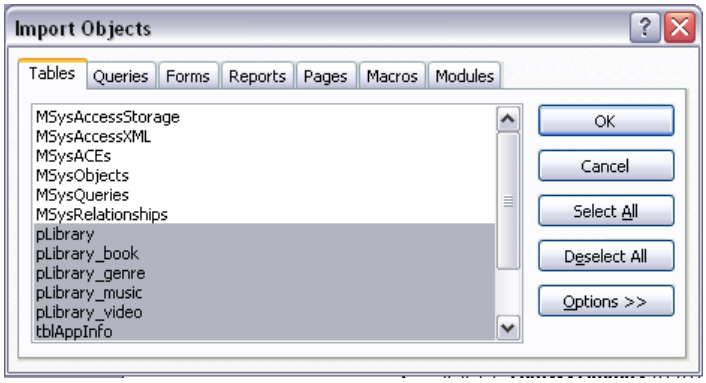
2.2.2.1 Access 2000 and Access 2003 Users

To import UI Builder forms, queries, tables, macros, and code modules into your existing database file, follow these steps:

1. We recommend you first backup your current application. You can backup your database when it is open by selecting **File>>Backup**, or by copying and pasting your database file in Windows Explorer.
2. Select **Tools>>Options** from the File menu.
3. In the Options dialog, select the View tab.
4. At the top right, make certain the "Hidden objects" and "System objects" check boxes are checked as shown below.



5. Select **OK**.
6. Select **File>>Get External Data>>Import…** from the File menu.
7. Select the UI Builder file you have downloaded or received in the File Open dialog.
8. In the Import Objects dialog, as shown below, select all tables except in the Tables tab except those that begin with "Msys"

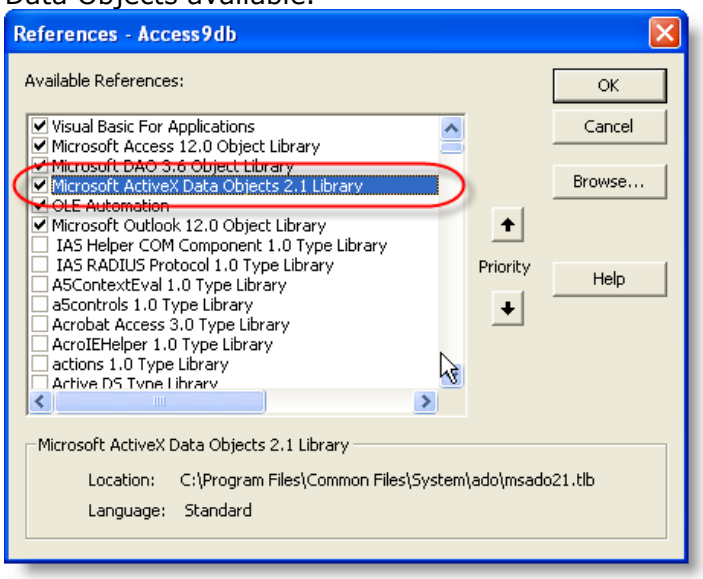


9. Select each remaining tab, and for each, click on the "Select All" button.
10. Click **OK**.

> **Important!**
> Any objects with the same name will be imported as "Name1." That is, Access will append a number to the end of the duplicate-named object.

11. Press **Ctrl+G** on your keyboard to open the Visual Basic Editor.
12. Select **Tools>>References** from the menu.

13. In the dialog, make sure you have an item with the name "Microsoft ActiveX Data Objects 2.x" checked. If you do not, scroll through the list and check the box next to the highest 2.x version of Microsoft ActiveX Data Objects available.

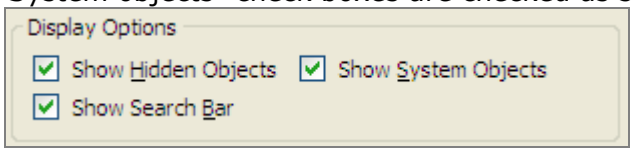


14. Click **OK**.
15. Click the Save icon in the toolbar.
16. Close the Visual Basic Editor Window.
17. You are now ready to proceed to the next step in Section 2.3.

2.2.2.2 Access 2007 Users

To import UI Builder forms, queries, tables, macros, and code modules into your existing database file, follow these steps:

1. We recommend you first backup your current application. You can backup your database when it is open by selecting the Office icon  , then **Manage>>Back Up Database** or by copying and pasting your database file in Windows Explorer.
2. Select the Office icon  , then **Access Options>>Current Database>>Navigation Options**...
3. In the Display Options box, make certain the "Show Hidden objects" and "Show System objects" check boxes are checked as shown below.

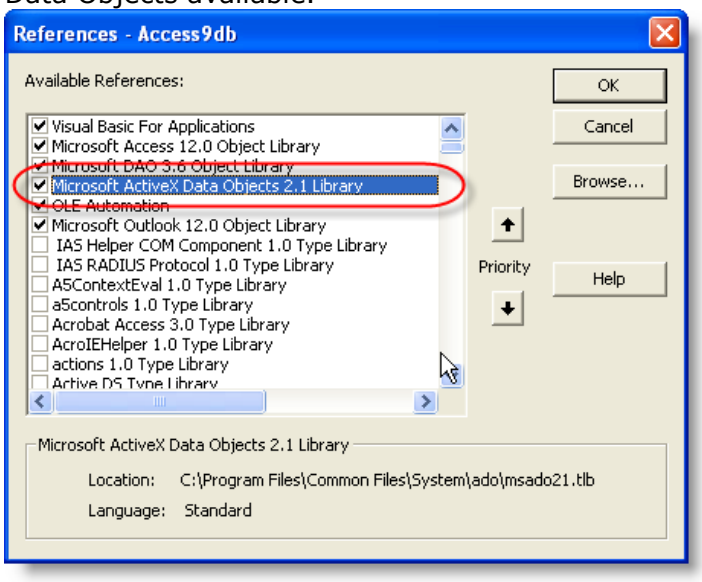


4. Select **External Data>>Access** from the Ribbon menu.
5. Select the UI Builder file you have downloaded or received in the File Open dialog.
6. In the Import Objects dialog select all tables except in the Tables tab except those that begin with "Msys." You can also exclude the sample tables: **pCustomer, pOrder, pCustomer_audit, pOrder_audit, pContact**

7. Select each remaining tab, and for each, click on the "Select All" button.

> **Important!**
> Any objects with the same name will be imported as "Name1." That is, Access will append a number to the end of the duplicate-named object.

8. Click **OK**.
9. Press **Ctrl+G** on your keyboard to open the Visual Basic Editor.
10. Select **Tools>>References** from the menu.
11. In the dialog, make sure you have an item with the name "Microsoft ActiveX Data Objects 2.x" checked. If you do not, scroll through the list and check the box next to the highest 2.x version of Microsoft ActiveX Data Objects available.



12. Click **OK**.
13. Click the Save icon in the toolbar.
14. Close the Visual Basic Editor Window.

2.3 VALIDATE OBJECTS

Once you have migrated your forms, queries, tables, macros, and code modules, we recommend you validate that everything is functioning as expected before you begin to integrate your application with UI Builder. To do so, simply perform the usual tasks you would normally do using your database application. This will ensure any integration issues are easily identified. If you application does not appear to be functioning as expected after migrating objects, ensure you imported all the objects you need, including any table relationships, Visual Basic for Applications (VBA) references, and import/export specifications. If you have VBA in your own database application, we recommend you compile the project to ensure there are no duplicate function names between your application and the UI Builder VBA.

2.4 OBJECT COMPATABILITY

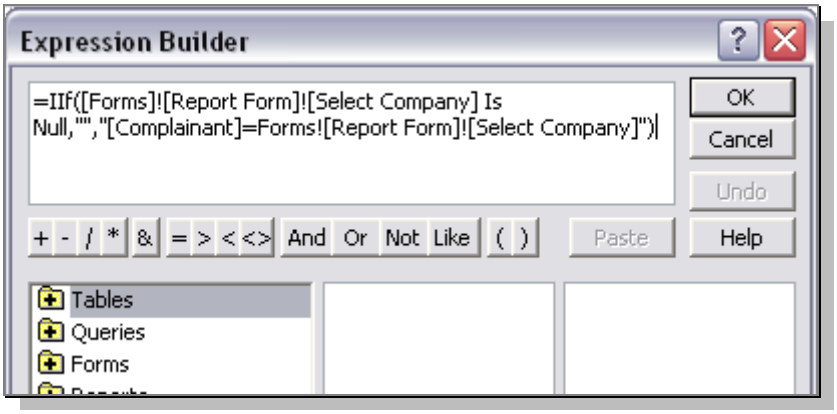
In most cases, you will not need to make significant changes to your forms, queries, tables, macros, and code modules. The largest change will be to resize your forms to take on the appearance you want within the UI Builder Main Subform.

2.4.1 FORM REFERENCES

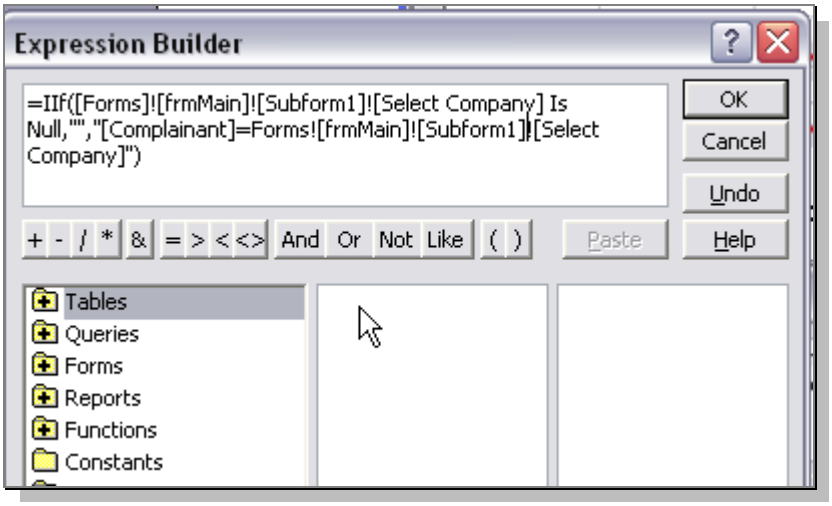
If you know your existing application has references to your forms, and you want those forms to appear in the subform of the main menu, you may need to reconfigure how the forms are referred to.

Example 1

You have subform fields or macros that refer to the parent form as follows:



You will need to change them to refer to the main form's (frmMain) subform (Subform1) as follows:



Note that "[Report Form]" in the original expression has been replaced by "[**frmMain**]![Subform]" as shown above. This is because your form (and any subforms within that form) become

a subform to the UI Builder's main window. You will need to replace **frmMain** as shown above with the name of the **frmMain** layout you choose to use, as shown in the table below:

Layout	Main Form Name
Left buttons, 1024x768	frmMain_left
Right buttons, 1024x768	frmMain_right
Top buttons, 1024x768	frmMain_top
Left buttons, 1028x 1024	frmMain_leftlg
Right buttons, 1028x 1024	frmMain_rightlg
Top buttons, 1028x 1024	frmMain_toplg
Resizing menu, 1028x 1024	frmMain_expandlg

Example 2
You have VBA code that refers to one of your forms as follows:

Forms!**MyFormName**.Form.FilterOn = False
Or
[Forms]![**MyFormName**].Form.FilterOn = False

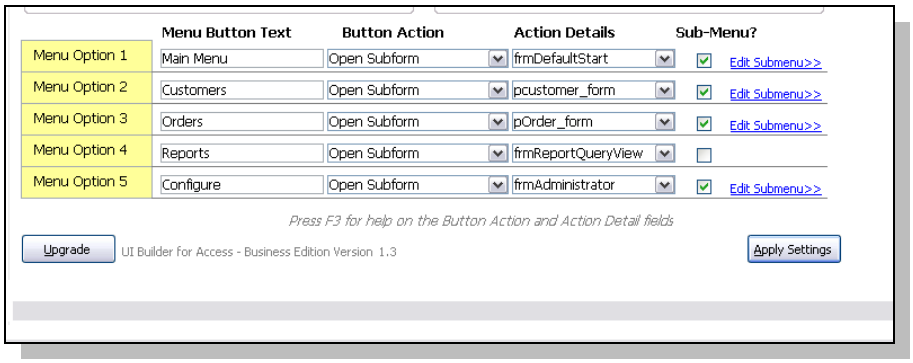
In this case, you simply need to make the following change:
Forms(gobjMain.Name)!Subform1.Form.FilterOn = False

2.5 CONFIGURING MENUS

2.5.1 CONFIGURING THE MAIN MENU

UI Builder – Starter Edition supports eight Main Menu items. Each Main Menu item can be configured to display a label you specify, and carry out an action you define. To configure the application, open the frmAdministrator, or select "Configure" from the Main Menu (if you have not already replaced the Configure menu option with one of your own).

As shown in the screen image below, there are five Main Menu options, each with corresponding Menu Button Text, Button Action, and Action Details fields.



If you do not want a specific Main Menu button to display to the user, simply clear the text from the Menu Button Text field for that button. For assistance with how to complete the Button Action and Action Details fields, press F3 on any specific field to display the help for that action.

2.5.1.1 Menu Actions

Button Action	Description	Action Detail Options
Create Email	Create a new email message in draft mode.	Optionally, specify the email address to default into the email.
Mail Merge (Business and Enterprise editions only)	Executes a pre-defined Mail Merge profile selected in the Action Details field.	Choose the Mail Merge profile you want to run.
Open Dialog Box	Displays a dialog box with the text you supply in the Action Details field.	Provide the message text you want to display to the user.
Open File	Opens a file specified in the Action Details field.	Provide the file name you want to open. Select "Browse..." to locate the file.
Open Form (Dialog mode)	Open a form in a new window in dialog mode (does not allow the user to navigate to any other part of the screen unless they close your form).	Select the name of a form in your database.
Open Form (New Window)	Open a form in a new window.	Select the name of a form in your database.
Open Report	Open a report in print mode. Note that the report will print automatically.	Select the name of a report in your database.
Open Report (preview)	Open a report in preview mode.	Select the name of a report in your database.
Open Subform	Open a form within the UI Builder menu screen.	Select the name of a form in your database.

Button Action	Description	Action Detail Options
Open Subform (add only)	Open a form within the UI Builder menu screen in Add-only mode. This means the user will not be able to see existing records in the table, only add new records.	Select the name of a form in your database. Refer to the "Advanced Menu Configuration" section for options on filtering records when the button is clicked by the user.
Open Subform (read-only)	Open a form within the UI Builder menu screen in read-only mode. This means the user will not be able to edit the records displayed.	Select the name of a form in your database. Refer to the "Advanced Menu Configuration" section for options on filtering records when the button is clicked by the user.
Open URL	Opens a web URL specified in the Action Details field.	Provide the URL, beginning with HTTP://
Run Code	Runs a VB function that you enter into the Action Details field. Note that Subroutines cannot be run with this option.	Pass any desired parameters to the Function in the Action Detail field, e.g.: fMyCode("parameter")
Run Code (delivered)	No longer used.	No longer used.
Run Macro	Executes a macro that you specify in the Action Details dropdown.	Select the name of a macro in your database.

2.5.2 CONFIGURING SUB-MENUS

Each Main Menu button can have it's own Sub-Menu. To activate a Sub-Menu, simply check the "Sub-Menu?" checkbox in the Administrator form. Select "Edit Submenu>>" to edit the Sub-Menu for that particular Main Menu item.

There are over 18 different actions you can perform for each Sub-Menu button. For assistance with how to complete the Button Action and Action Details fields, press F3 on any specific field to display the help for that action. Finally, similar to the Main Menu buttons, if you do not want a specific Sub-Menu button to display to the user, simply clear the text from the Menu Button Text field for that button.

2.5.2.1 Sub-Menu Actions

Button Action	Description	Action Detail Options
Add Record	Add a new record to the current form.	None.
Clear Subform Filter	Clear a filter on the current form. This action is usually used after the Filter Subform action (below) is used.	Specify the form record source you want to use. For example: "Select * from tblProjects" will select all records from tblProjects.
Copy Record	Copy the current record to a new record.	None.

Button Action	Description	Action Detail Options
Create Email	Create a new email message in draft mode.	Optionally, specify the email address to default into the email.
Create Outlook Task (Business and Enterprise editions only)	Create a new Microsoft Outlook task by prompting the user for the task details.	None.
Delete Record	Delete the currently selected record. The user will be prompted to confirm the action first.	None.
Delete Record (no confirm)	Delete the currently selected record without prompting the user to confirm.	None.
Export to CSV	Exports a table/query to comma-separated value file (CSV). The user will be prompted to provide the filename for the exported file.	Select the table or query to export.
Export to Excel	Exports a table/query to Excel-compatible file. The user will be prompted to provide the filename for the exported file.	Select the table or query to export.
Filter Subform	Filter the currently displayed form.	Specify the form record source you want to use. For example: "Select * from tblProjects WHERE [Project-Status] <> 'Cancelled'" will select all records from tblProjects where the Project-Status field is not equal to Cancelled.
Mail Merge (Business and Enterprise editions only)	Executes a pre-defined Mail Merge profile selected in the Action Details field.	Choose the Mail Merge profile you want to run.
Open Dialog Box	Displays a dialog box with the text you supply in the Action Details field.	Provide the message text you want to display to the user.
Open File	Opens a file specified in the Action Details field.	Provide the file name you want to open. Select "Browse..." to locate the file.
Open Form (Dialog mode)	Open a form in a new window in dialog mode (does not allow the user to navigate to any other part of the screen unless they close your form).	Select the name of a form in your database.
Open Form (New Window)	Open a form in a new window.	Select the name of a form in your database.
Open Report	Open a report in print mode. Note that the report will print automatically.	Select the name of a report in your database.
Open Report (preview)	Open a report in preview mode.	Select the name of a report in your database.
Open Subform	Open a form within the UI Builder menu screen.	Select the name of a form in your database.

Button Action	Description	Action Detail Options
Open Subform (add only)	Open a form within the UI Builder menu screen in Add-only mode. This means the user will not be able to see existing records in the table, only add new records.	Select the name of a form in your database. Refer to the "Advanced Menu Configuration" section for options on filtering records when the button is clicked by the user.
Open Subform (read-only)	Open a form within the UI Builder menu screen in read-only mode. This means the user will not be able to edit the records displayed.	Select the name of a form in your database. Refer to the "Advanced Menu Configuration" section for options on filtering records when the button is clicked by the user.
Open URL	Opens a web URL specified in the Action Details field.	Provide the URL, beginning with HTTP://
Print Screen	Print the current screen. Note that it will only print the visible screen, including menu. We recommend creating a report if you wish to print in a specialized format.	None.
Run Code	Runs a VB function that you enter into the Action Details field. Note that Subroutines cannot be run with this option.	Pass any desired parameters to the Function in the Action Detail field, e.g.: fMyCode("parameter")
Run Code (delivered)	No longer used.	No longer used.
Run Macro	Executes a macro that you specify in the Action Details dropdown.	Select the name of a macro in your database.
Run Query	Opens a query you specify in a new window.	Select the name of a query in your database.

2.5.3 Advanced Menu Configuration

UI Builder 2.4 and higher support the ability to configure a menu button to open a form in the Main Subform window using a filtered or limited recordset. You may want to set two menu buttons to display the same form, but each button should display different information in the form. For example, one button opens the Accounts form and shows only records where a field "Active" is equal to true. Another button opens the Accounts form, but shows records where the Account-Type field is equal to "Prospect." To accomplish this, you would set the Action Details field of the UI Builder menu setup form as follows:

```
frmCustomer:RECORDSOURCE:SELECT * FROM
[tblCustomer] WHERE [Account-Type] = 'Prospect'
```

In the example above, the form "frmCustomer" will be opened in the Main Subform window with the recordset limited to only those records where Account-Type equals 'Prospect.' Use this method where you do

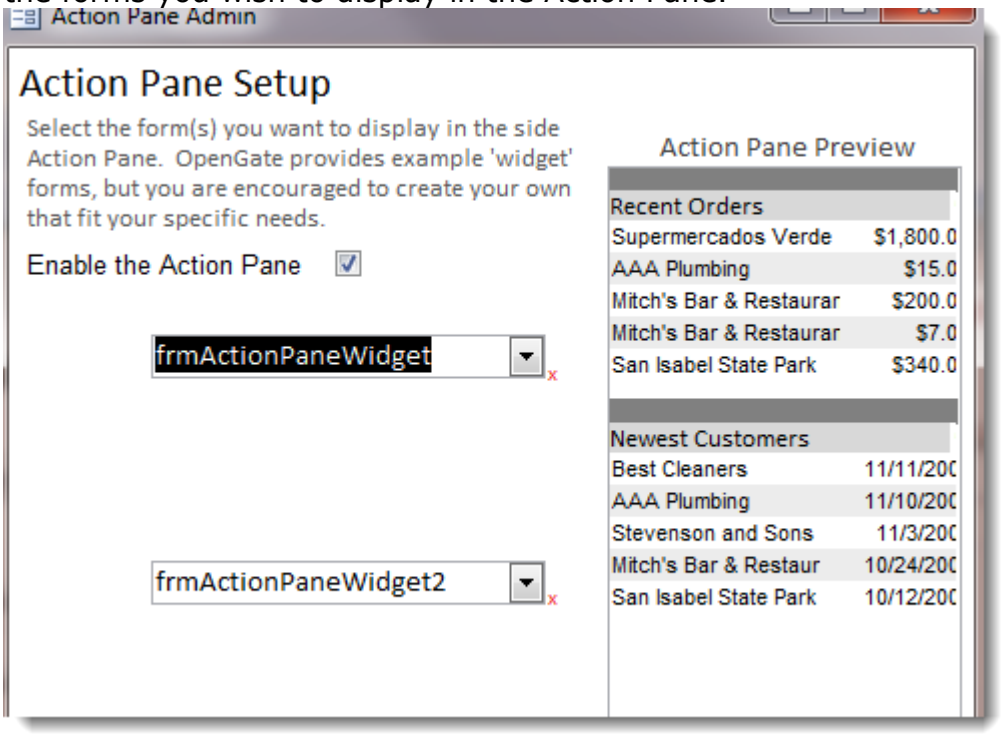
not want users to be able to clear the form filter and view other records in the table.

frmCustomer:FILTER:[Active] = -1

In the example above, the form "frmCustomer" will be opened in the Main Subform window and filtered to only show those records where Active equals True. Use this method where you are not concerned about users clearing the form filter and viewing other records in the table.

2.6 RIGHT-HAND ACTION PANE

Several UI Builder menu layouts support a persistent right-hand Action Pane that you can configure to display small forms as individual resizable panels within the Action Pane. For example, you may want to display a form with a list of recent new orders or statistics. To set up the Action Pane, make sure you are using one of the following menu layouts: Left (large), Expanding Menu (left), or the Top (large) menu layout. In the UI Builder Administration form, you can launch the Action Pane setup screen (frmActionPaneAdmin). In the dialog, select the forms you wish to display in the Action Pane.



Several example forms are provided with UI Builder. All begin with "frmActionPane...Widget." You can use these forms as examples for how to create your own Action Pane forms. Two example forms "frmActionPaneWebWidget#" can be changed to display different websites by selecting the "Edit" button when displayed.

Note
After enabling the Action Pane, you will need to restart UI Builder in order to see the Action Pane if you have upgrade from an earlier version..

2.7 RESIZE FORMS

UI Builder – Starter Edition offers three distinct skins (layouts) in two resolutions. If you wish to resize and save the main menu form(s), open the form in Form Design mode to accommodate a different screen resolution. Any changes to the "frmMain_" forms will be overwritten when you upgrade UI Builder, unless you override the upgrade function for that object. Refer to section 4 of this document for instructions on overriding the upgrade process for a specific form.

2.8 UNIVERSAL SEARCH

A new feature in UI Builder 3.4 is a universal search box that allows your users to search your forms.

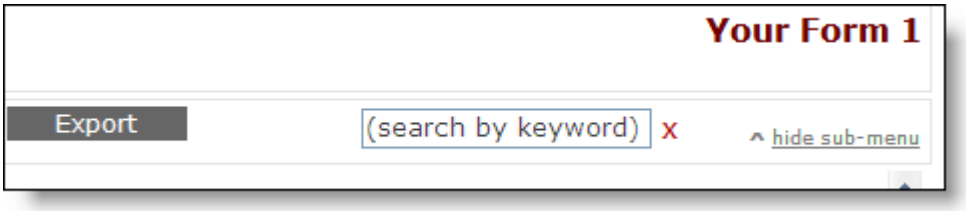


Figure 1: Universal Search Box

The search box will appear when a user opens any form in the main subform window, the form is bound to a data source, and one or more fields on the form has the tag of 'search'.

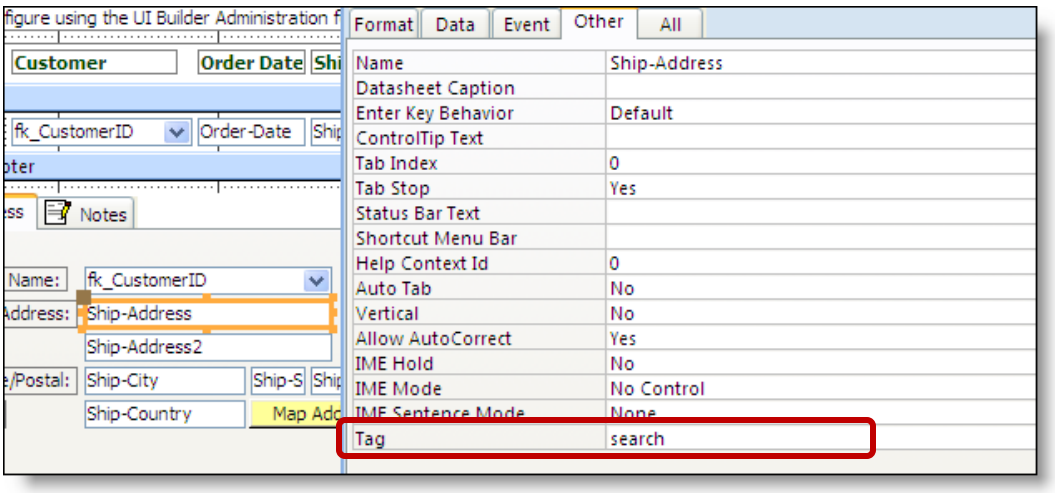


Figure 2: Control Tag Property

If a form is not bound to a data source, or no fields have the tag "search", the search box will not be visible.

> **Note**
> You can disable the universal search if needed by changing the constant gblnEnableUniversalSearch to 'False'. The constant is located in the module modMenu.

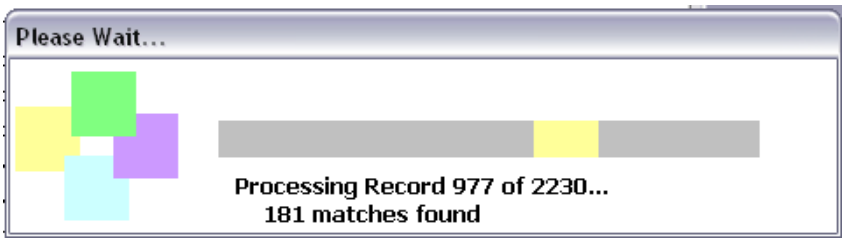
3 VBA Toolbox

The UI Builder – Starter Edition comes with a set of pre-packaged VBA functions in the module "modVBAToolbox." Several of these functions are collected from other sources, and are not included as part of the price of UI Builder, but simply as a convenience to our customers.

3.1 PROGRESS BAR

Description

The progress bar is used to notify the user of the present state of the application. The progress bar is more visible than the status bar (acSysCmdSetStatus) function, and does not require an ActiveX control, which can sometimes be problematic in a multi-user environment where each machine has different versions of the Microsoft Common Controls ActiveX file. Note that the progress bar allows you to specify the text to display to the user, but does not provide a % complete visualization, just an indication that the system is still working.



Progress Bar Example

Function Name

fProgressUpdate

Parameters and Use

strUpdateText – String Variable. Populates the text shown on the progress bar. Up to two lines of text can be displayed as shown in the example above.

iPercent – A whole number that represents the percent of the progress to represent on the bar.

Note that you must open and close the form in our code using the Docmd.OpenForm and Docmd.CloseForm commands.

Depending upon how intense your processing is, you may need to also insert the "DoEvents" command into your code to let Access refresh the form occasionally.

Example

```
Docmd.OpenForm "frmProgress"

fProgressUpdate "This is the text that would be displayed" & vbCrlf &
"on two different lines in the progress bar"
        (your own code doing something here)
fProgressUpdate "Next status update you want to provide"
DoEvents
        (your own code doing something here)
fProgressUpdate "Next status update you want to provide"
DoEvents
(your own code doing something here)

        Docmd.CloseForm "frmProgress"
```

3.2 EVENT LOGGING

The Event Log function in UI Builder provides a way for you to log important events to a local or remote table, or to an XML output file in the location you specify. There are three functions that are used for event logging, one to initialize the log, the second to write to the log, and the third to close the log when finished. To optimize performance, if you are using a table to log events, the table will remain open for the duration of the user's session, or until you intentionally close the log with the fCloseLog() function.

If you choose to write to the log table ("tblEventLog"), you can locate the table locally on the PC where your database application is used, or copy the table to a file server and then link the table back to your user's database.

Similarly, you can choose to store the XML event log on the local PC where the database is being used, or to a central location such as a file server.

> **Important!**
> Each copy of your database application can write to the same remote log table (linked) or central file. Note that each copy of your database application will have it's own settings and if you want to change them universally, you will need to do so for each copy.

3.2.1 CONFIGURING THE LOG

To configure the log, open the Administration form/subform, or open the form "frmLoggingAdmin" from the database window.

Logging Administration Form

You can choose to turn logging on or off for the current Access database, specify if log events should be written to the event log table, or a file location you specify. Note that if you specify a file name, you will need to provide a fully qualified path name and file name in the "Log Location" field. The file format will be XML (without header/footer), and will not depend upon the file name extension you supply in the "Log Location" field.

The logging level will dictate whether certain events are written to the log or not. When you define a log event you can indicate if it is a Normal or Debug event. If you have the Logging Level set to "Debug," all events will be written to the log. If set to "Normal," only normal events will be written to the log, Debug events will be ignored.

Finally, you can send an email to a designated administrator when certain events occur. There are three values for the "Event Emails" setting:

Never – Events will never be emailed to the Administrator.

Defined Events – Only events where the parameter "blnEmailAdmin" is set to True in the fLogEvent() function.

Critical Errors – Any event where the parameter "intEventType" is set to "auiCriticalError" will be emailed to the designated administrator.

3.2.2 INITIALIZING THE LOG

Description

The function fInitializeLog() obtains logging settings from the table tblAppInfo, such as where log events should be written to (table or file), what level of log events will be written, and what email address should be used to email log events, if any. The function is automatically called whenever the main form (frmMain) is opened. If you want to log specific events that might happen before the main form is ever opened, you will need to call fInitializeLog, otherwise it is done automatically for you.

Function Name

fInitializeLog

Parameters and Use

No parameters.

3.2.3 WRITING TO THE LOG

Description

Writing to the log can occur at any time after the log is initialized.

Function Name

fLogEvent

Parameters and Use

strDescription - Description of the event which will be passed to the user and/or table/file.

intEventLevel - Determines if the event will be written to the table/file based on the application's logging level (normal/debug).

blnAlarmUser - If set to True, the user will see a popup alert with the text of the event.

strSource - The source of the event.

intEventType - Category of the event to help you understand if it is informational, an error, or critical error.

blnEmailAdmin - Determines if the event is emailed to an administrator. There are three modes, the default being only those events where they blnEmailAdmin is explicitly set to True. Alternatively, you can set the system to email no events, or critical errors, in which case it doesn't matter whether blnEmailAdmin is True or False when passed in. Note that blank = Never email events.

Example 1

```
fLogEvent "Loading Menu", auiDebug, False, "fLoadMenu()",
auiInformation
```

In this example, we are logging an event with description "Loading Menu" that will only be captured if the Logging Level = Debug, we won't alert the user, and the origin is "fLoadMenu()". This is an informational event. No email will be sent to the administrator. If the Logging Level = Debug, this is how the event will appear in the log table:

8/5/2007 1:03:33 AM	Loading Menu	fLoadMenu()	USER123	USER123-LAP	Informational	Debug

Example 2

```
fLogEvent err.Description, auiNormal, True, "fMyFunction(" &
strMyVariable & ")", auiCriticalError, True
```

In this example, we are logging an event with description of the Access error that occurs (generally when you are using error handling in a function). It will only be captured regardless of the Logging Level, and we will alert the user. The origin is "fMyFunction" and we will also log the information that was passed into the function through strMyVariable. This is a critical error.

User Alert Dialog

3.2.4 Closing the Log

Description

The function fCloseLog() simply closes down the log function and, if necessary, the log table recordset as well. The function is automatically called whenever the main form (frmMain) is closed.

Function Name

fCloseLog

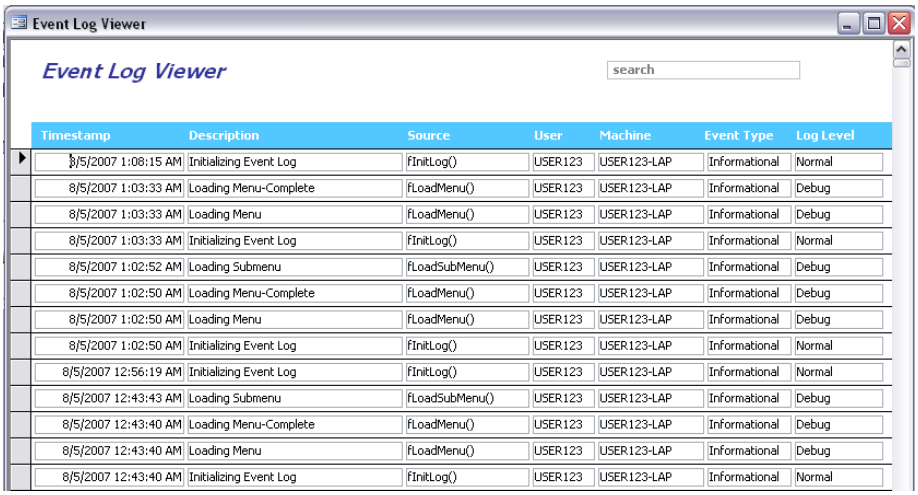
Parameters and Use

No parameters.

3.2.5 Viewing the Log

If you save log events to the XML output file, you will need to open the file by navigating to the location and opening it. If you write log events to the event log table, you can view the event records by opening the hidden form

frmLogViewer or by selecting "Click Here To View Event Log>>" from the Logging Administrator form.



Event Log Viewer

3.3 TABLE RECORD COUNT

Description

The function fTableRecordCount enables you to quickly obtain a count of records in a specified table using an SQL string you supply. You can call this function from a form field, or from another VBA function. This function provides a simple alternative to the Access DCount() function, with embedded error handling and event logging.

For example, you may have a field on a form "Customers" that you want to show the user the number of customers currently in your database. You could insert this function into the read-only field to display that information to them. Of you may have a function where your processing depends upon whether there are records in a specific table, or a specific number of records in that table meeting a given criteria.

Function Name

fTableRecordCount

Parameters and Use

strTable –Provide a valid table name (linked or local) for the current database

strSQL – Provide a valid SQL string beginning with "WHERE…" If you want to count all records in the table, simply pass "" in the variable.

blnDisplayErrors– Optional. Set to False if you do not want this function to display an error to the user if something goes wrong. For example, if

you have this function called multiple times, you may not want to alert the user every time an error occurs. If left blank, UI Builder will display any error messages.

Example 1

lngRecords = fTableRecordCount("tblMyTable","")

In this example, a value will be returned to lngRecords for the number of records contained in the table "tblMyTable." No SQL criteria will be applied.

Example 2

lngRecords = fTableRecordCount("tblTableX","WHERE [CustomerID] = " & me.customerid.value)

In this example, a value will be returned to lngRecords for the number of records contained in the table "tblTableX" where the CustomerID field is equal to a value passed from the current form for the field "customerid."

3.4 TABLE RECORD SUM

Description

The function fTableRecordSum enables you to quickly obtain a sum of records in a specified table field using an SQL string you supply. You can call this function from a form field, or from another VBA function. This function provides a simple alternative to the Access DSum() function, with embedded error handling and event logging.

For example, you may have a field on a form "Revenue" that you want to show the user the amount of revenue this month. You could insert this function into the read-only field to display that information to them. Or you may have a function where your processing depends upon whether you have generated a specific amount of revenue from orders.

Function Name

fTableRecordSum

Parameters and Use

strTable –Provide a valid table name (linked or local) for the current database

strSumField – Designate the name of the field in *strTable* that contains the data you want to sum together.

strSQL – Provide a valid SQL string beginning with "WHERE…" If you want to count all records in the table, simply pass "" in the variable.

blnDisplayErrors– Optional. Set to False if you do not want this function to display an error to the user if something goes wrong. For example, if you have this function called multiple times, you may not want to alert the user every time an error occurs. If left blank, UI Builder will display any error messages.

Example 1

lngRecords = fTableRecordSum("tblMyTable","MyField","")

In this example, a value will be returned to lngRecords for the sum of values contained in the field "MyField" in table "tblMyTable." No SQL criteria will be applied.

Example 2

lngRecords = fTableRecordSum ("tblTableX","MyField", "WHERE [Order-Date] > #1/1/2007#")

In this example, a value will be returned to lngRecords for the sum of values contained in the field "MyField" in table "tblTableX" where the Order-Date field is greater than 1/1/2007.

3.5 TABLE RECORD VALUE

Description

The function fTableRecordValue enables you to quickly obtain a value from a field in a specified table field using an SQL string you supply. You can call this function from a form field, or from another VBA function. This function provides a simple alternative to the Access DLookup() function, with embedded error handling and event logging.

For example, you may have want to fetch the current value of a field that is in a table other than the one your form uses as the data source. You could insert this function into the read-only field to display that information to them. UI Builder uses this function, as an example, to change the color scheme on some forms to conform to the color scheme you choose. The form load event obtains your currently active color scheme's menu button color to paint the menu administration form buttons the same color.

Important!
This function will return the first value found in the data source for the field you specify. If you happen to have multiple records in the data source that match your SQL expression, UI Builder will only return the first record found.

Function Name

fTableRecordValue

Parameters and Use

strTable –Provide a valid table name (linked or local) for the current database

strField – Designate the name of the field in *strTable* that contains the data you want to fetch.

strSQL – Provide a valid SQL string beginning with "WHERE…" If you want to count all records in the table, simply pass "" in the variable.

blnDisplayErrors– Optional. Set to False if you do not want this function to display an error to the user if something goes wrong. For example, if you have this function called multiple times, you may not want to alert the user every time an error occurs. If left blank, UI Builder will display any error messages.

Example 1

```
lngButtonColor = fTableRecordValue("tblColorScheme", "MenuButtons",
"WHERE [ActiveScheme] = -1", False)
```

In this example, a value will be returned to lngButtonColor for the first value contained in the field "MenuButtons" in table "tblColorScheme" where the scheme is currently active. Errors will not be displayed to the user if they occur.

3.6 FILE CHECK

Description

The function fFileExists allows you to validate whether a file exists or not. When you call fFileExists with a filename, it will respond with a True/False to your function to indicate if the file exists or not.

Function Name

fFileExists

Parameters and Use

strFileName – Supply the name of the file you want to determine if it exists or not.

Example

```
blnMyVariable = fFileExists("C:\Win.ini")
```

The function will return a True/False value to your calling function for the file "C:\Win.ini"

3.7 OPERATING SYSTEM NAME

Description

The code for this function is courtesy of Dev Ashish. Call the function within any other function to retrieve the name of the operating system upon which your Access database application is running.

Function Name

fOSName

Parameters and Use

No Parameters.

3.8 NETWORK USERNAME

Description

The code for this function is courtesy of Dev Ashish. Call the function within any other function to retrieve the network username for the machine on which your Access database application is running.

Function Name

fOSUserName

Parameters and Use

No Parameters.

3.9 MACHINE NAME

Description

The code for this function is courtesy of Dev Ashish. Call the function within any other function to retrieve the name of the machine on which your Access database application is running.

Function Name

fOSMachineName

Parameters and Use

No Parameters.

3.10 NOTE EDITOR

Description

Note Editor gives you the ability to condense a memo or long text field on screen, saving valuable form real estate. You place a button next to the field, or create an OnDoubleClick event, that calls the fNoteEditor function to present a small form that allows users to view and/or edit the contents of the field. The results of the user's edits are returned in the function's return variable.

Function Name

fNoteEditor

Parameters and Use

strNote - The current text contained in the field, if any.

blnReadOnly- True/False. Specify if the Note Editor should allow the user to edit the notes, or just read them.

Return Value – Text supplied by the user, or original text if no changes were made.

3.11 TOAST POPUPS

Description

Toast popups allow you to display an informational message to the user for a short period of time similar to email message notifications now available in most email clients.

Function Name

fDisplayPopup

Parameters and Use

strHeader - The header/title text that will appear at the top of the popup message in bold. Similar to the title bar of a message box.

strText- The main text to display to the user.

iDisplaySeconds- Optional. Indicates how long the popup message should show. By default, the message will show for 5 seconds.

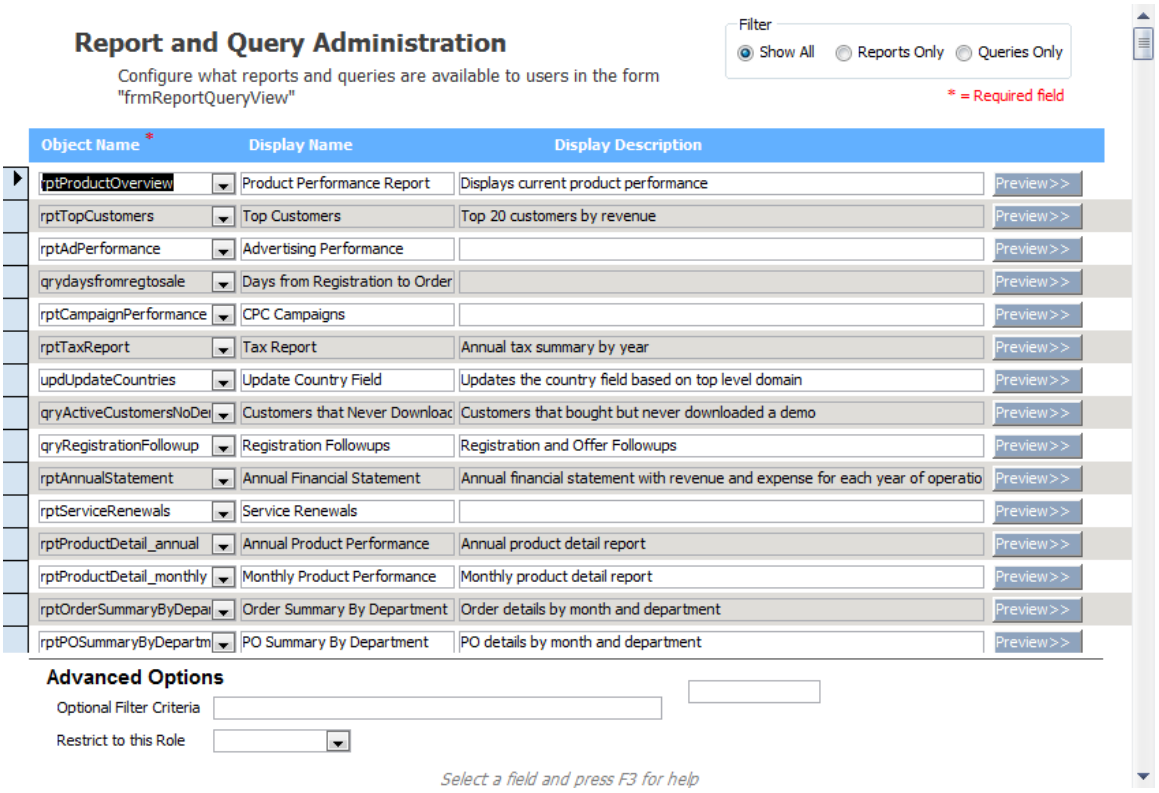
blnLegacyMode- Optional. Indicates that UI Builder should use the expanding popup mode which was present in UI Builder 3.0.

4 Report/Query Directory Menu

The Report/Query Directory menu (frmReportQueryViewer) allows you to present a simple menu of available reports and queries in your database to users, with plain-language names and descriptions. The and queries may not be important enough to consume one of your main menu or submenu items, but users need access to the reports and queries on occasion.

4.1 REPORT/QUERY DIRECTORY SETUP

To add reports and queries to the reports menu, open frmReportQueryAdmin, or access the form from the Tools tab in the Administration form (frmAdministrator).



Object Name The report or query to display on the menu.

Display Name The name to display to the end user.

Description The description to display to the user.

Optional Filter Criteria You can optionally specify the filter criteria to use when the user opens, prints, or saves the report.

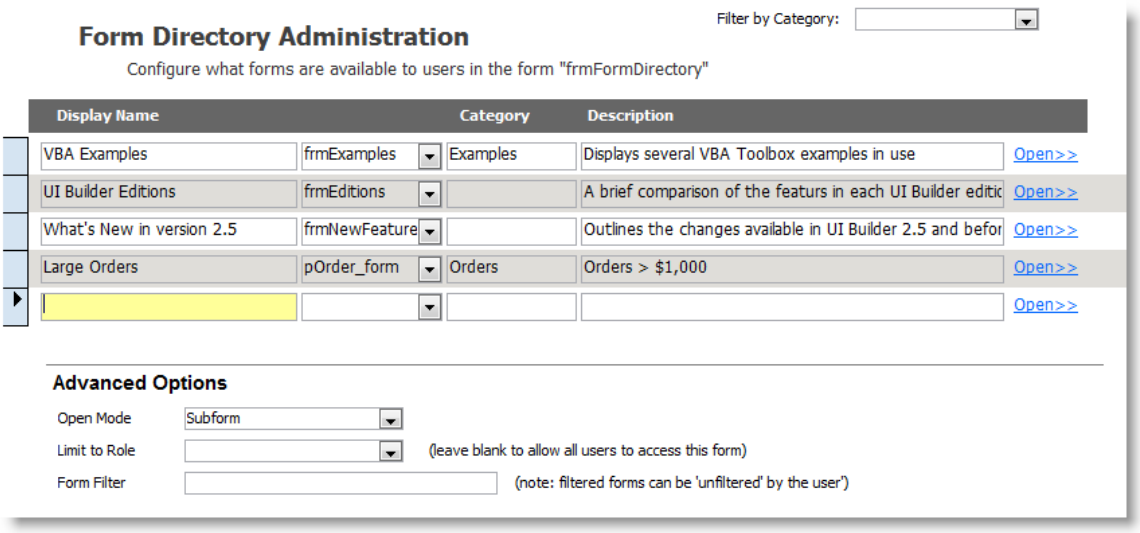
Restrict to Role Available in the Enterprise Edition of UI Builder only.

5 Form Directory Menu

The Form Directory menu (frmFormDirectory) allows you to present a simple menu of available forms in your database to users, with plain-language names and descriptions. The forms may not be important enough to consume one of your main menu or submenu items, but users need access to the form(s) on occasion.

5.1 FORM DIRECTORY SETUP

To add forms to the form directory, open frmFormDirectoryAdmin, or access the form from the Tools tab in the Administration form (frmAdministrator).



Object Name	The form to display on the menu.
Display Name	The name to display to the user.
Category	A self-assigned category. Type any text you want into the field to help users filter by category.
Display Description	The description to display to the user.
Optional Open Mode	How you want the form to open on screen.
Optional Filter Criteria	You can optionally specify the filter criteria to use when the user opens, prints, or saves the report.
Limit to Role	Available in the Enterprise Edition of UI Builder only.

6 Developer Tools and Tips

6.1 DEVELOPER MENU

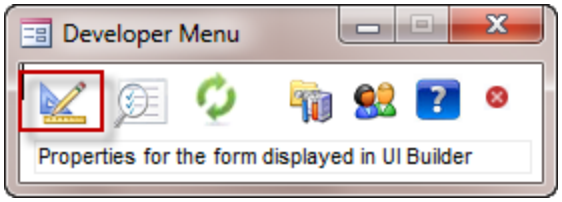
The UI Builder Developer Menu is a tool to help you design and manage your forms within UI Builder with greater ease. The floating menu can be dismissed at any time, and allows you to configure which users (by username) are authorized to see the Developer Menu.

6.1.1 ENABLING THE DEVELOPER MENU

The Developer Menu is available by default, but can be dismissed at any time by clicking the small red "close" icon in the menu. To restore the menu, select the "Enable UI Builder Developer's Menu" from the main administration form (frmAdministrator) "Advanced Settings" tab.

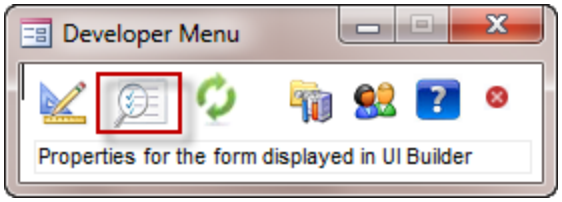
6.1.2 FORM DESIGN

Select the Design icon in the Developer Menu to be able to design the form currently displayed within the UI Builder menu. Select the same icon to restore your form to display within UI Builder.



6.1.3 FORM PROPERTIES

Select the Properties icon to view the properties of the form currently displayed within UI Builder. This is helpful when you need to research or troubleshoot your form's behaviour. For example, the form may have "Allow Additions" set to "False," which would explain why your users are not able to add new records. You can then research whether the menu button is configured to open the form in read-only mode, or code in your form is affecting the form's behaviour.



6.2 CHANGING THE FORM DISPLAYED IN UI BUILDER FROM CODE

To programmatically change the subform displayed within the UI Builder menu, you can use the sChangeSubform() command.

Parameters and Use

iButton - Always use the number 0 for this argument.

strCommandDetail- The form you want to display.

blnFromSubcommand- *Always use True for this argument*.

The following example will change the form displayed within UI Builder to the form "frmMyForm":

```
sChangeSubform 0, "frmMyForm",True
```

Advanced Use

You can also tell UI Builder to perform an action after sChangeSubform changes the subform displayed in UI Builder.

To filter the form when it opens, use the following convention:

```
sChangeSubform 0, "frmMyForm:FILTER:[CustomerID] = " & CustomerID.Value ,True
```

In the example above, the form "frmMyForm" will be displayed, and then filtered to only show records where the CustomerID on the record equals the CustomerID on the form that executed the command.

To change the form's recordsource when it opens, use the following convention:

```
sChangeSubform 0, "frmMyForm:RECORDSOURCE:SELECT * FROM [tblCustomer] WHERE [CustomerID] = " & CustomerID.Value ,True
```

In the example above, the form "frmMyForm" will be displayed, and the recordsource will be changed to the SQL statement provided.

To go to a specific record in the form when it opens, use the following convention:

```
sChangeSubform 0, "frmMyForm:GOTO:[CustomerID] = " & CustomerID.Value ,True
```

In the example above, the form "frmMyForm" will be displayed, and the focus will change to the first record with a matching Customer ID to that specified by the form that executed the command.

6.3 ADDING MENU BUTTONS

OpenGate has created a whitepaper with information about how to add more menu and submenu buttons to the UI Builder menu, as well as alternate options. Click here to open the whitepaper.

7 Upgrading UI Builder

This section has been moved to a separate guide. Please refer to the separate UI Builder Upgrade Guide for information.

Release History – UI Builder Starter Edition

Version 1.0 – Initial Release

Version 1.1 – Minor changes to resolve various errors

Version 1.2 – Enhancements to the Upgrade feature

Version 1.3

- Fixed modUpgrade function to prevent application from crashing during upgrade
- Fixed issue where upgrade function did not pull over new version number
- Added progress bar to VBA toolkit and upgrade
- Added dropdown to the Action Details fields in the Sub-Menu Administrator
- Fixed issue with custom color chooser if you didn't select a new color

Version 1.3.1

- Fixed logo not found error when opening the frmMain forms
- Added skins for 1280x1024 resolutions (previously optimized for only 1024x768)
- Fixed issue where upgrade function failed if an object to be upgraded had been removed from the current version
- Added capability to upgrade the modUpgrade module

Version 1.3.2

- Fixed flashing menu issue
- Automatically reloads color scheme if you enter VBA break mode
- Added new menu button option to open forms in read-only mode

Version 1.4

- Added ability to send email messages to application administrator when the event log is called for specific event types
- Added 3 more main menu buttons
- Fixed issue where subform tried to use Linked Child/Master Ids when changing subforms.

Version 1.4.2

- New Administrator form with tabs for easier navigation

Version 1.4.3

- Added ability to hide database window in new Advanced tab within Administrator form

Version 2.0

- User Help Administration
- Note Editor
- Added Examples form
- Eliminated main form open background loading with black buttons
- Progress Bar now can be incremented based on % completion

Version 2.1

- Ability to minimize main menu bar on left and right hand layouts (skins)
- Incorporated PowerBrowser capability
- Introduced Toast Popups (fDisplayPopup)

Version 2.2

- New Processing form (similar to Google Analytics)
- New VBA Toolbox function - fTableRecordSum()
- Minor changes to sample Order and Customer forms
- New Dashboard demonstrates fTableRecordCount and fTableRecordSum capabilities
- Enhanced Report/Query screen to allow users to send email with object attached and view count of records contained in a query

Version 2.3

- Enhancements
  - New splash screen.
  - New "Open Form (add only)" menu option.
  - Add optional parameter to suppress errors in fTableRecordCount, fTableRecordSum, and fTableRecordValue functions. Suppressing errors ensures user's aren't prompted numerous times that an error exists if you create a dashboard or form that uses many invalid fTableRecord*** calls. By default errors are shown.
  - New "File Browse…" ability when "Open File" is chosen from the menu administration screens.
  - The subform administration (frmAdmin_submenu) form's "Action Type" dropdowns now support quick entry without requiring the user to display the full list.
  - Enhanced the Color Scheme functionality support an unlimited number of user-defined color schemes. The color scheme editor also allows you to define the color for the Application Name, Screen Name, and Company Name text on the frmMain screen.
  - New Toast Popup form that fades in and out.

- Defect Resolutions
  - Fix to better support upgrade from Business to Enterprise (adds records to tblSubMenu for each role pre-defined in the Enterprise Edition brought over)

Version 2.4

- Enhancements
  - The Report/Query Viewer allows users to email reports as PDF documents in Access 2007.
  - New filter by event type feature to the Event Log viewer.
  - New Upgrade function is faster and more gracefully handles missing objects in the local database.
  - New feature allows you to configure a menu button to open a form with a filtered or alternate recordset.
  - New diagnostics utility to identify menu setup problems if you have manually imported different menu tables from another UI Builder-based database.

- The Upgrade function now prevents a user from attempting to upgrade the database if they are in a locked database (MDE, ACCDR, ACCDE).

Version 3.3

(note: for simplicity, all UI Builder editions have been placed on the same version number starting with 3.3. Functionality will differ between editions, however the core objects are shared)

- Enhancements
  - The Report/Query Administration screen now allows you to define an SQL Where statement to be used when opening a report, allowing you to list the same report in the Report/Query Viewer menu (frmReportQueryView) with unique filter criteria.
  - The Report/Query Viewer can display a record count when a Query is selected.
  - The progress bar has a new look and feel.
  - The function fChangeSubform will now open a form in a new window if frmMain is not open.

- Defect Resolutions
  - Removed an error message from fFileExists (the function returns the Boolean value, which should be sufficient)
  - The popup form (frmPopup) would raise error 2450 if opened in succession too quickly.
  - A new "State Loss" message will tell users when frmMain needs to reload. This occurs when Access enters the Break Mode in VB.
  - If the Start Form is missing, an unhandled error would occur rather than displaying a popup message.
  - The Subform Administration form's Action Detail dropdown boxes were not displaying available forms when a form-related Action was chosen.
  - If a user selected "Apply Settings" and frmMain is not open, an unhandled error would occur.
  - If a user changes a color scheme or the menu layout when frmMain is not open, UI Builder will no longer open frmMain or raise an error.
  - The function fCheckLinks raised an error if the back end database was not found (it should just prompt the user to re-link the tables instead of raising an error dialog).
  - Added popup error messaging to frmReportQueryAdmin and frmReportQueryView when a report or query is not found.

Version 3.4

- Enhancements
  - New Universal Search box allows you to add open text search without any coding.

Version 3.5

- Enhancements
  - New Form Directory feature allows you to display an unlimited number of forms in a directory form to users (similar to the Report/Query menu). Each form can be configured to open a specific way (dialog,

filtered). The Enterprise Edition enables you to limit visibility for a specific for to a designated role.
    - A new menu layout where the main menu buttons expand or collapse to display submenu buttons below them.
    - Updated layout selection form.
    - The main form is now referenced as the object gobjMain. This means rather than deleting the form "frmMain" and replacing with a copy of the desired layout (e.g., "frmMain_leftlg"), UI Builder directly opens the form that corresponds to the chosen layout. This allows you to let users change the menu layout even when your database is compiled as an MDE/ACCDE file.
    - Hidden object are automatically displayed when migrating your existing database into UI Builder.
    - The function to count records in a table (fTableRecordCount) has been enhanced to provide 10-20% better performance on local tables.
    - New setup wizard simplifies the startup process for new UI Builder databases.
    - The main form area now expands vertically as users resize their Access window to give even more vertical screen real estate to your forms.
    - Large format layouts now have wider submenu buttons.

- Defect Resolutions
    - The mail merge profile data source dropdown was corrected to include linked tables.

Version 4.0

- Enhancements
    - New Developer Menu.

Version 4.1

- Enhancements
    - New right-hand Action Pane.
    - You can now change the menu font family and font sizes in the Color Scheme editor.