UI BUILDER™ FOR ACCESS – ENTERPRISE EDITION VERSION 3.2

Application Guide

Version 05.04.2008

TABLE OF CONTENTS

1 GENERAL CONCEPTS AND RESOURCES 4

1.1 User INterface 4
1.2 Resources 4

2 MIGRATING YOUR APPLICATION 5

2.1 Migration Options 5
2.2 Migration steps 5
2.2.1 Importing your forms, queries, tables, macros, and code modules 5
2.2.2 Importing UI Builder Into your Database 6
2.3 Validate Objects 7
2.4 Object Compatability 7
2.4.1 Form References 7
2.5 Configuring Default Menus 8
2.5.1 Configuring the Default Menu 8
2.5.2 Configuring Sub-Menus 9
2.5.3 Advanced Menu Configuration 9
2.6 Managing Roles 10
2.6.1 Adding roles 10
2.6.2 Testing Roles 11
2.6.3 Removing Roles 11
2.6.4 Administrator security 11
2.7 Managing Users 11
2.7.1 Adding users 12
2.7.2 Deleting users 12
2.7.3 Anonymous Users 12
2.7.4 Restricting users 13
2.7.5 Allowing users to switch roles 13
2.8 Resize Forms 14

3 VBA TOOLBOX 14

3.1 Progress Bar 14
3.2 Event Logging 15
3.2.1 Configuring the log 16
3.2.2 Initializing the Log 16
3.2.3 Writing to the Log 17
3.2.4 Closing the Log 18
3.2.5 Viewing the Log 18
3.3 Table Record Count 19
3.4 Table Record Sum 20
3.5 Table Record Value 21
3.6 File Check 22
3.7 Operating System Name 22
3.8 Network UserName 23
3.9 Machine Name 23
3.10 Create Outlook Task – Direct function call 23
3.11 Mail Merge API – Direct Function call 24
3.12 Note Editor 25
3.13 Toast Popups 25

4 UPGRADING UI BUILDER 27

5 ONE-CLICK MAIL MERGE 27

5.1 Mail Merge Profiles 27
5.2 Outlook security warnings 29

5.3 CONFIGURING MENU COMMANDS 30

6 DYNAMIC USER HELP 31

6.1 CREATING USER HELP TOPICS 31
6.2 ADDING VISUAL BASIC FOR HOTKEYS 32
6.3 ADDING VISUAL BASIC FOR COMMAND BUTTONS 32

7 RECORD-LEVEL AUDITING 33

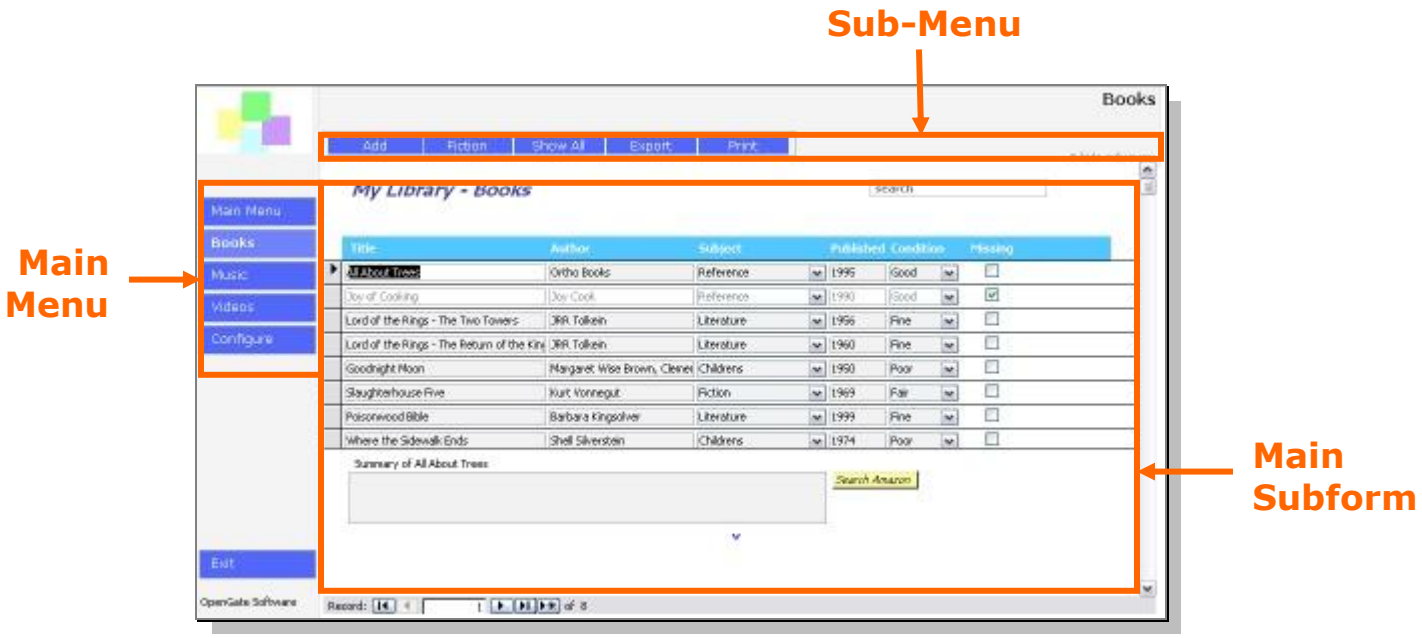
7.1 AUDIT HISTORY SETUP 33
7.2 CREATING A NEW AUDIT TABLE 34
7.2.1 Creating new Audit tables 34
7.2.2 Creating VB code for your form 35
7.2.3 Activating auditing for your form 36
7.2.4 Activating auditing for Other forms that use the same table 36
7.3 MANAGING AUDIT HISTORY TRACKING 37
7.3.1 Enabling/Disabling auditing 37
7.3.2 Changing audit table names 37
7.3.3 Changing audit table fields 38

RELEASE HISTORY – UI BUILDER ENTERPRISE EDITION 39

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 do not have 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

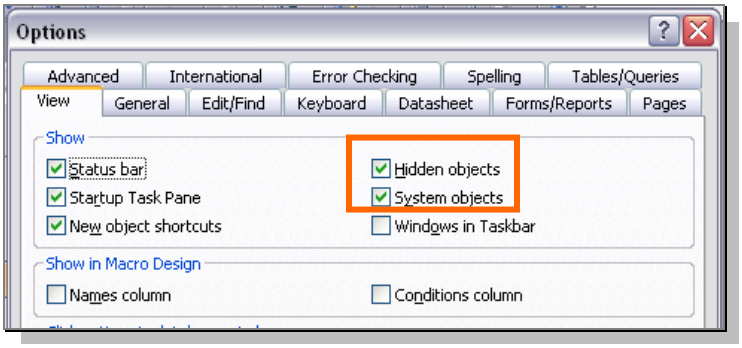
To import your forms, queries, tables, macros, and code modules into the UI Builder file, follow these steps:

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.

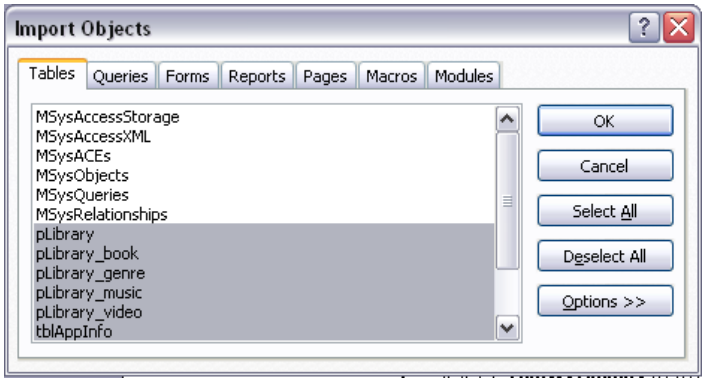
2.2.2 Importing UI Builder Into your Database

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. You are now ready to proceed to the next step in Section 2.3.

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

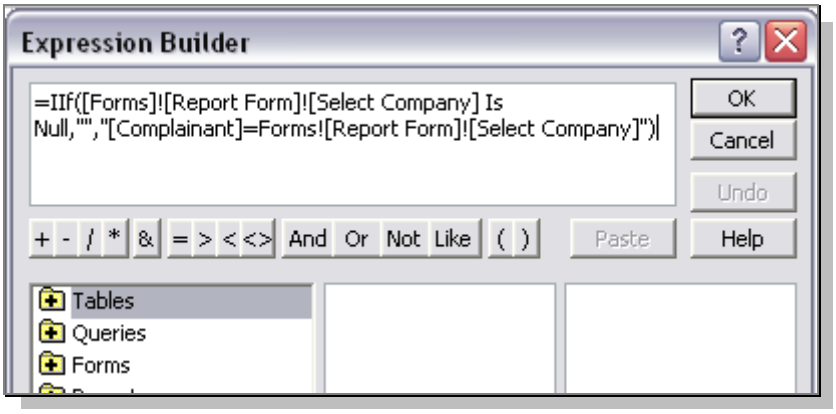
In most cases, you will not need to make significant changes to your forms, queries, tables, macros, and code modules. The largest change will be to resize your forms to take on the appearance you want within the UI Builder Main Subform.

2.4.1 FORM REFERENCES

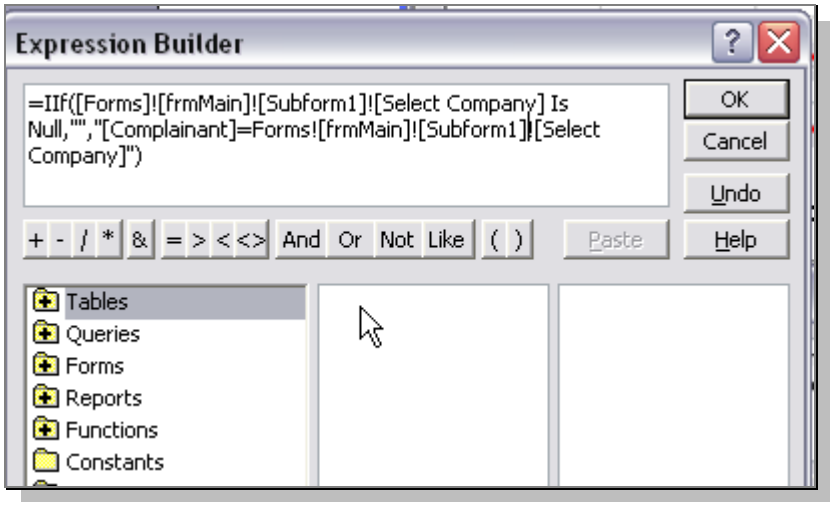
If you know your existing application has references to your forms, and you want those forms to appear in the subform of the main menu, you may need to reconfigure how the forms are referred to.

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.

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

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

In this case, you simply need to make the following change:

Forms!**frmMain!Subform1**.FilterOn = False
Or
[Forms]!**[frmMain]![Subform1]**.FilterOn = False

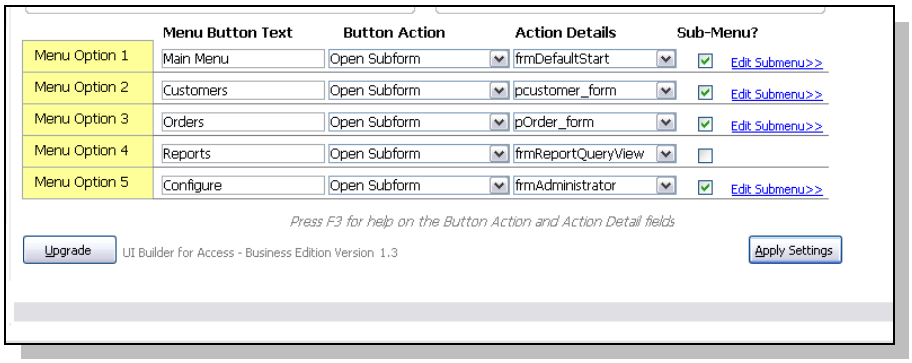
You no longer need to reference your form's name, but rather, Subform1, which is the subform displayed within the main form (frmMain).

2.5 CONFIGURING DEFAULT MENUS

2.5.1 CONFIGURING THE DEFAULT MENU

UI Builder – Enterprise 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.

2.5.2 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.

2.5.3 Advanced Menu Configuration

UI Builder 3.2 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

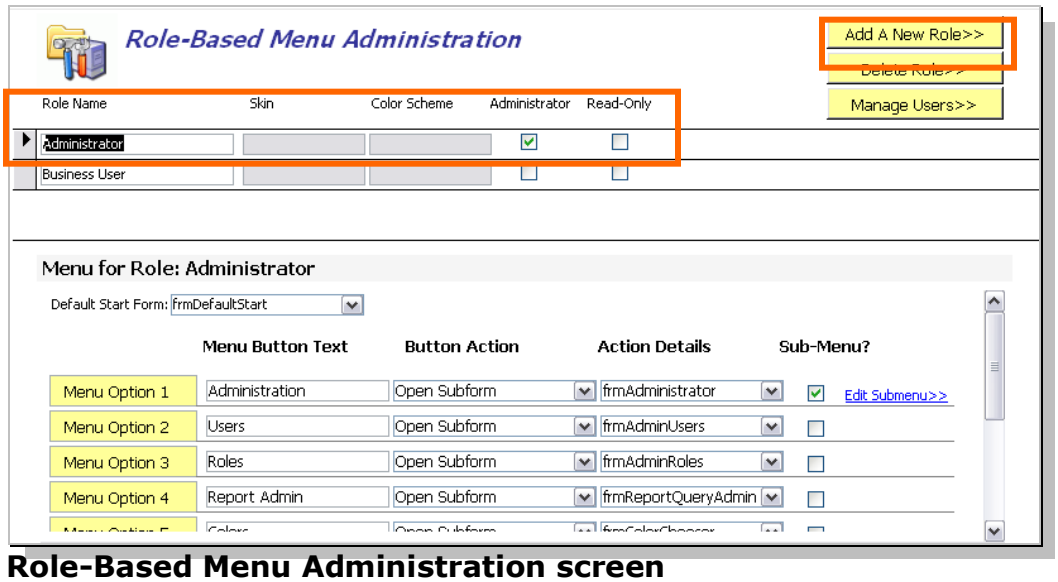
In the example above, the form "frmCustomer" will be opened in the Main Subform window and filtered to only show those records where Active equals True. Use this method where you are not concerned about users clearing the form filter and viewing other records in the table.

2.6 MANAGING ROLES

UI Builder simplifies menus at a user-level by allowing you to assign one or more users to a role, which in turn defines a set of menu options for that role. For example, all users assigned to the role "Sales Rep" may be able to add new customers, whereas user associated with the role "Customer Service Rep" may only be able to view customers, but not modify the customer records.

2.6.1 ADDING ROLES

To add a new role, or change an existing one, navigate to the Role-Based Menu Administration screen. The screen is accessible from the main Administrator form, or by opening the hidden form "frmAdminRoles."



Role-Based Menu Administration screen

Select "Add a New Role>>" at the top right of the screen. In the dialog prompt, enter the name of the role. The new role will appear in the role list. Then simply configure the menu and submenu button options at the bottom of the screen for the new role.

Note that if the "Read-Only" check box is checked, users associated with this role will be able to open forms you define for their menu options, but all forms will open as read-only.

2.6.2 TESTING ROLES

UI Builder 3.0 and above include the ability for administrators to test other roles without logging in as a different Windows NT user. Simply select "Test Role>>" from the list of roles and select the role you want to switch to in the popup form. You can test the role's menus and actions, then close the popup form when you want to return to your assigned role. This functionality is only available to administrators.

2.6.3 REMOVING ROLES

To remove a role, select the "Delete Role>>" at the top right of the screen. If there are users associated with the role, you will not be able to delete the role until you remove the users, or reassign them to another role.

2.6.4 ADMINISTRATOR SECURITY

When you first open the UI Builder application, the NT login currently in use will be assigned as the default administrator. You can see the designated default administrator in the main Administration screen. The default administrator is allowed to access and update information in any of the administration screens. All others are not able to open these forms, unless they are associated with a role where the "Administrator" check box is checked.

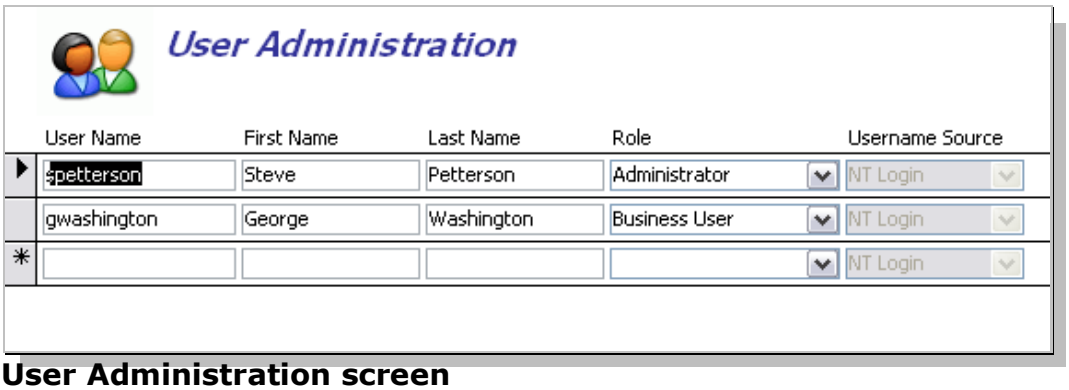
2.7 MANAGING USERS

UI Builder Enterprise Editions 3.2 and higher support two user login methods. You can rely on each user's NT (Windows) Login to authenticate them and automatically log in authorized users, or you can employ the UI Builder login prompt. If each user will log in from their own machine, we recommend you employ the NT Login method. However, if you have a shared machine that is used by multiple individuals without logging in separately, the UI Builder login prompt can provide user level menus despite the fact all users are logged in with the same Windows username.

Please refer to the *UI Builder Security Guide* for detailed information on how to set up the UI Builder login prompt method.

2.7.1 ADDING USERS

To add a new user, select the "Manage Users" button from the main Administration form, or open "frmAdminUsers."



User Administration screen

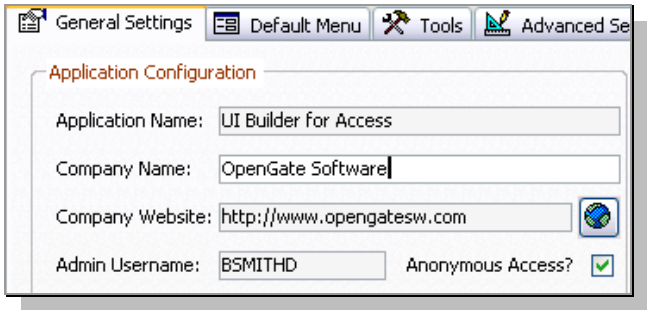
You must minimally specify each user's NT login username, and assign a role to the user from the drop-down box of available roles.

2.7.2 DELETING USERS

To delete a user, select the user's record, and then select "Delete [username]>>" from the bottom of the User Administration form.

2.7.3 ANONYMOUS USERS

If desired, you can allow anonymous users to access the system. That is, users that do not have a user record established in the UI Builder user table. These individuals will have access to the default menus you set up in the Administration form "Default Menu" tab. To allow anonymous users, simply check the "Anonymous Access" box in the General Settings tab of the Administration form.



Note
UI Builder will still capture the Windows login username for any anonymous user's activities. They are anonymous only insofar as they do not have a formal user profile with an assigned role in your application.

2.7.4 RESTRICTING USERS

UI Builder 3.1 and above support the ability to lock a user out of the application, while allowing anonymous users. For example, you may have a specific individual with a Windows Login that should not be allowed to access your database, but other users should be able to access the application even if they do not have a user record created in the UI Builder user table. To prevent a user from accessing the system, leave their user record in the use table, but clear the role field for that user.

2.7.5 ALLOWING USERS TO SWITCH ROLES

UI Builder 3.2 and higher allow you to optionally configure one or more users to be able to access multiple roles without logging in as a different user. For example, you may have a need for several complex menu schemes and have several staff that need access to both the "Sales" and "Operations" roles.

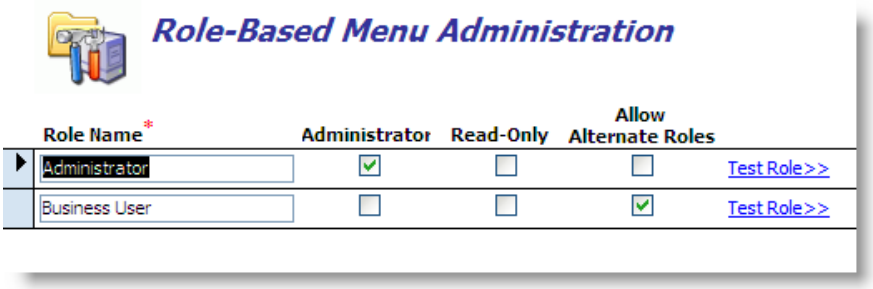


Figure 1: Role-Based Menu Administration Form

As shown above, you may define for each role whether to allow alternate roles to be presented to the user. If the **Allow Alternate Roles** box is checked, UI Builder will attempt to locate a record in the table usysUserAltRole table with the user's User ID. If an entry exists, UI Builder will display the Alternate Role popup form shown below:

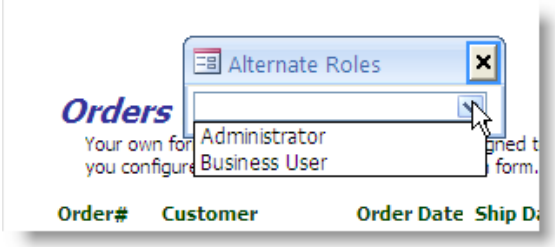


Figure 2: Alternate Role Popup Form

The Alternate Role popup form allows the user to quickly toggle between different roles you have set up for them in the usysUserAltRole table.

2.8 RESIZE FORMS

UI Builder – Enterprise 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.

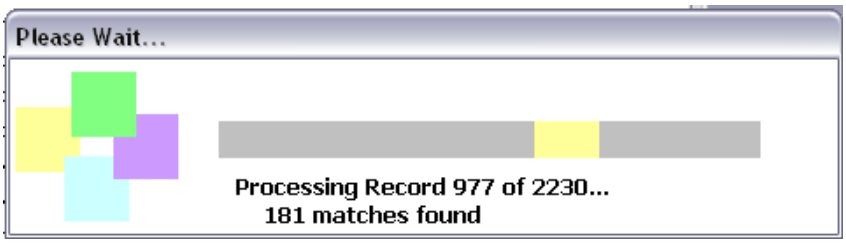
3 VBA Toolbox

The UI Builder – Enterprise 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 included as part of the price of UI Builder, but simply 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.

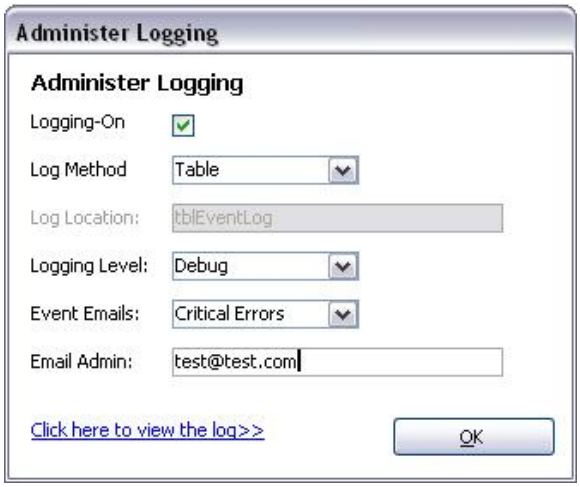
If you choose to write to the log table ("tblEventLog"), you can locate the table locally on the PC where your database application is used, or copy the table to a file server and then link the table back to your user's database.

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

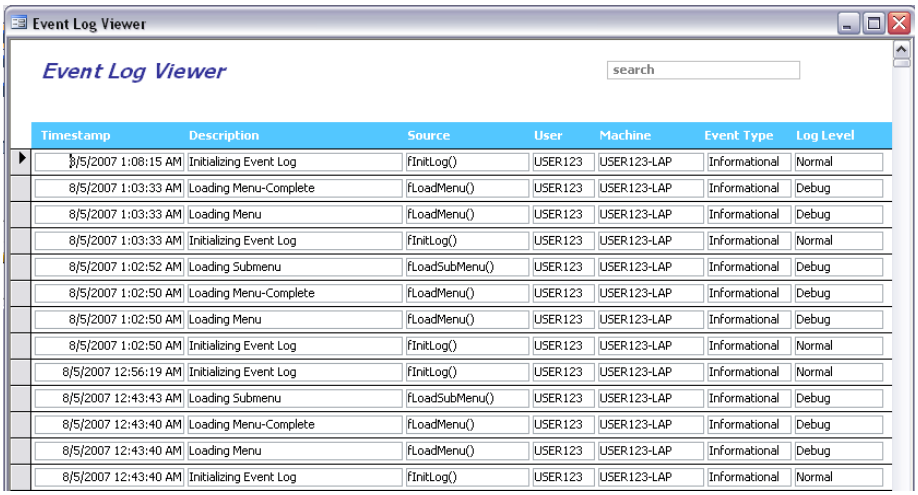
Parameters and Use

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.

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.

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.

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 CREATE OUTLOOK TASK – DIRECT FUNCTION CALL

Description
UI Builder helps users create Outlook tasks quickly and efficiently, without ever leaving your database application. Simply configure a menu button to use the "Create Outlook Task" command, and UI Builder takes care of the rest. When clicked, the user is prompted to add details about the task, click "Create Task>>", and they continue working on what they were doing without the normal multi-step process. To call the create task function directly, use the fAddOutlookTask function. For example, you might add an event that automatically creates an Outlook task to confirm an order shipped up five days in the future when a new order is entered.

Function Name
fAddOutlookTask

Parameters and Use

strSubject – The subject of the Outlook Task.

strBody – The details for the task created.

dtDueDate – Optional due date. If not supplied, the current date will be used.

Return Value – Yes/No value indicating if the merge was successful or failed.

Example

See the form frmExamples for an example of the methods used to create Outlook tasks.

3.11 MAIL MERGE API – DIRECT FUNCTION CALL

Description

UI Builder provides the ability to let users click a menu or submenu button and generate a mail merge letter or email based on a predefined Mail Merge Profile. If you want full control over how the mail merge will function, use the fMerge() function call directly.

Function Name

fMerge

Parameters and Use

strMergePath – Provide the full filename for the mail merge template document, or "Prompt User" to have the user select a template when the mail merge is run.

strDataSource – Provide the text name of the table or query to be used in the mail merge. Be sure the data source contains all the data fields that the mail merge template also uses to prevent user error messages from Microsoft Word.

strWhereStatement - Provide a valid SQL statement, beginning with "WHERE ", or leave as "" to select all records from the specified data source.

strMergeOption - Specify the type of mail merge. *Valid values are*

mrgPrint – Print the merged document immediately
mrgPreview– Open the merged document for editing
mrgMail – Send the merged email immediately
mrgMailPreview– Open the merged email for editing

strSubject – Email subject line. If the merge option mrgMail or mrgMailPreview is selected, this variable must be populated.

strEmail - Email recipient(s). Use semi-colons for multiple recipients. If the merge option mrgMail or mrgMailPreview is selected, this variable must be populated.

blnKeepOpen - True/False value that indicates if the document should remain open if using the mrgPrint method.

Return Value – Yes/No value indicating if the merge was successful or failed.

3.12 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.

3.13 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 Upgrading UI Builder

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

5 One-Click Mail Merge

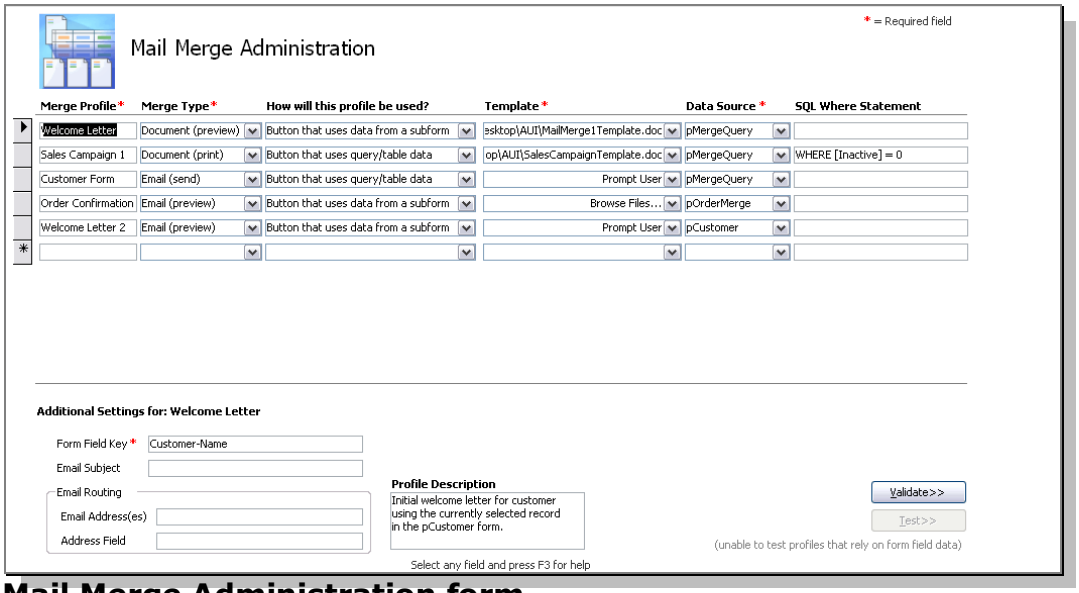
Microsoft Access allows you to create formatted reports for printing, but in today's business environment, where formatted emails are far more effective and more frequently needed, MS Access falls short. Reports must be emailed as RTF or Snapshot format attachments, with lack the same appearance to the recipient, or require the recipient to download the Microsoft Snapshot viewer.

UI Builder gives you the ability to offer users One-Click Mail Merge to email. You establish a profile for each Mail Merge button you will let the user choose from, which includes the location of the mail merge template, the merge data source, what action to take (print, email, display for editing), SQL filters to apply, and which fields on the displayed form to use to pull merge data. For example, you may want users to be able to view a customer record, click a menu button, and have an email message automatically sent to the customer displayed on screen. With UI Builder's One-Click Mail merge, you can add that capability to your database applications without a single line of Visual Basic code.

> **Important!**
> Due to limitations in Office 2000 and below, One-Click Mail Merge is only supported in Office 2002 and higher.

5.1 MAIL MERGE PROFILES

The first step is to create one or more Mail Merge Profiles that will drive the command button behaviour for a mail merge activity. Navigate to the Mail Merge Administration form either by clicking on the button in the Tools tab of the main Administrator form, or by opening the hidden form "frmMergeAdmin" directly.



Mail Merge Administration form

Merge Profile – Enter a brief name for the Merge Profile.

Merge Type – Specify whether the merge command will open a document to print, preview, create a single email that is sent immediately, opened for editing, or generate a mass email merge. Depending upon your selection, several fields may be required to complete the merge profile, and a red asterisk (*) will appear next to these fields.

How Will This Profile be Used? – Indicate how you plan to use the Mail Merge Profile. Valid values are:

Button that uses query/table data – This option means you plan to perform a mail merge that will not be directly related to data currently displayed to the user on a form. Instead, the merge will operate based on the data source and any SQL Where statement you provide.

Button that uses data from a subform – This option means that you will specify a field on a subform that will be used to select the right data to merge when the user selects a menu button or command button you create. If you select this option, several fields will be required to complete the merge profile, and a red asterisk (*) will appear next to these fields. For example, you may want to have a submenu button that allows the user to merge the currently displayed customer record to an email document, using the CustomerID field as the merge filter criteria. Note that this option is not available for mass email merges.

Template – Specify the location of a valid mail merge template document, or choose "Prompt User" to have UI Builder ask the user to locate the template document each time they run the mail merge.

Data Source – Select a table or query that will be used to obtain the mail merge data.

SQL Where Statement – Optionally provide a valid SQL statement, beginning with "WHERE ", or leave blank to select all records from the specified data source.

Form Field Key – If you will be using information on the subform displayed to the user at the time they initiate the mail merge, specify the data field name on that form that should be used at runtime to filter the mail merge data. For example, if there is a field "Customer-ID" on the form that you want to use when merging data from the pCustomer table, you would populate the Form Field Key with **Customer-ID**.

> **Important!**
> The Form Field Key name used must correspond to the name of the subform field and table field name. In other words, if you will be filtering the mail merge on Customer-Name, the table field name must be Customer-Name, and so must the field name on the form. Otherwise you will receive an error that no records match the merge criteria, or an error that the field doesn't exist in the data source.

Email Subject – If the merge type will be to an email, provide a subject line to be used in the email.

Email Address(es) – If the merge type will be to an email, provide one or more email addresses separated by semi-colons, or complete the Address Field described below.

Address Field – If you will be using an email address from the subform displayed to the user at the time they initiate the mail merge, specify the email address field name on that form that should be used at runtime as the recipient's email address. If you plan to generate a mass email merge, you must specify the data field that contains email addresses in the table or query that you specified in the Data Source section of the Merge Profile.

Profile Description – This field can contain any notes you might want to keep about the merge profile for future reference. It is not used in the actual mail merge process.

5.2 OUTLOOK SECURITY WARNINGS

Microsoft Outlook 2002 and later will display warning messages when another application attempts to send email messages automatically on behalf of the user. We recommend you review the Microsoft documentation for information on ways to disable the warning, and decide whether doing so makes sense for your organization. For Access 2007 users, you can avoid receiving the security warning message by ensuring you have an active Anti-Virus client on your PC with email security. Access 2007 will only display the warning message for each email if your anti-virus is out of date or disabled.

5.3 CONFIGURING MENU COMMANDS

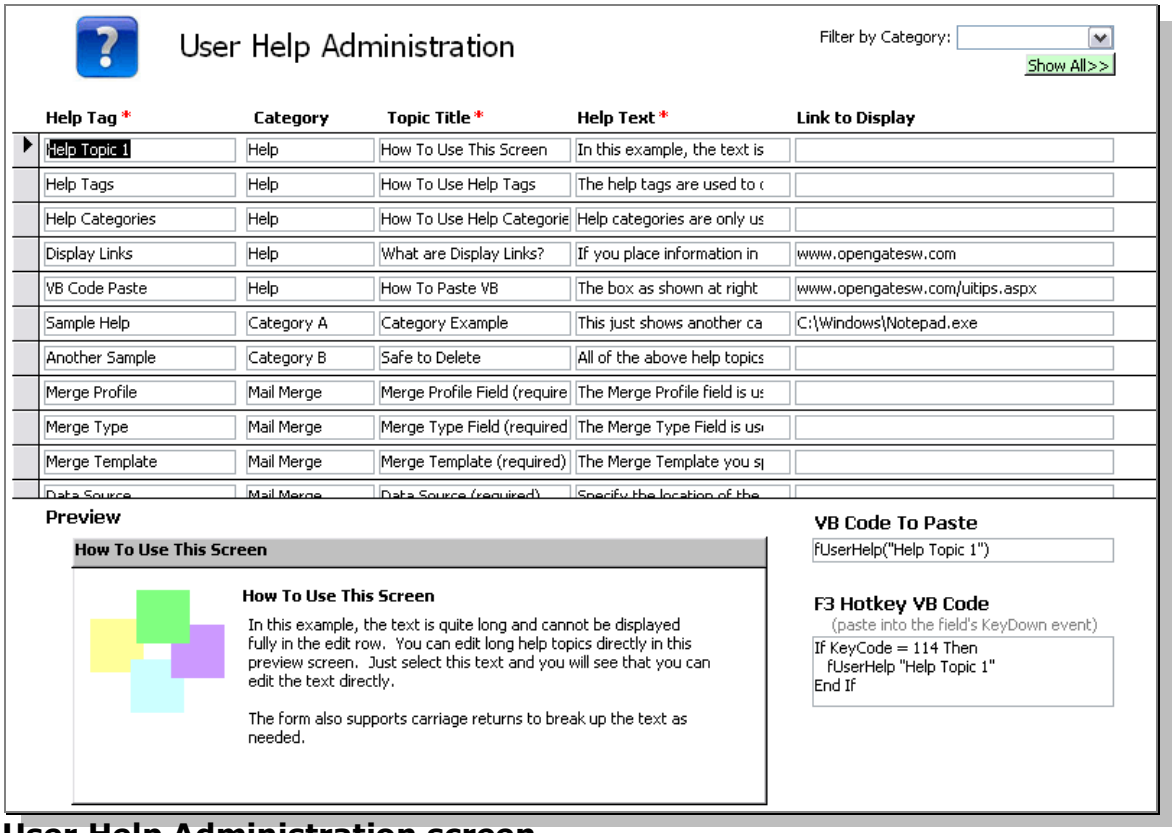
Once you have created a Mail Merge Profile, follow the normal steps to configure a menu or submenu button. In the Button Action field, specify "Mail Merge," and then select the appropriate Mail Merge Profile in the "Action Details" field.

6 Dynamic User Help

Dynamic User Help provides a way to create user help topics that can be accessed via a hotkey (such as F3) or by adding a command button to your form. Once you have created your user help topics, and added the necessary VB code to your form(s), it is easy to change the user help without changing the VB code on each form. Simply change the user help topic in the User Help Administration screen. For multi-user environments, you can place the table tblUserHelp in a linked database that is centrally located, making it easy to change help topics once instead of for each user's database.

6.1 CREATING USER HELP TOPICS

Navigate to the User Help Administration form either by clicking on the button in the Tools tab of the main Administrator form, or by opening the hidden form "frmUserHelpAdmin" directly.



User Help Administration screen

Help Tag – Help tags are used to determine which help topic to display to the user when the perform a specific action. You define each help tag, and they must be unique. There are no rules as to what the tag must be.

Category – Help categories are only used to help you categorize your user help topics. You can assign a category to your help topics and then filter by

those categories as shown in the top right hand side of the User Help Administration screen.

Topic Title – Dialog box title and help topic title.

Help Text – The actual help text that will be displayed to the user.

Link to Display – Information in the display link field, it will appear to the user at the bottom of the help window. When they click on the link, it will attempt to open the associated file or website.

6.2 ADDING VISUAL BASIC FOR HOTKEYS

You can enable a hot-key for a field such that if the user selects that key (we suggest F3), it will automatically display your help topic. Paste the text in the box at the right-hand side of the screen labeled "F3 Hotkey Code" into any fields KeyDown event to have that topic display to the user.

6.3 ADDING VISUAL BASIC FOR COMMAND BUTTONS

Paste the text in the box at the right-hand side of the screen labeled "VB Code to Paste" into any button's OnClick event to have that topic display to the user.

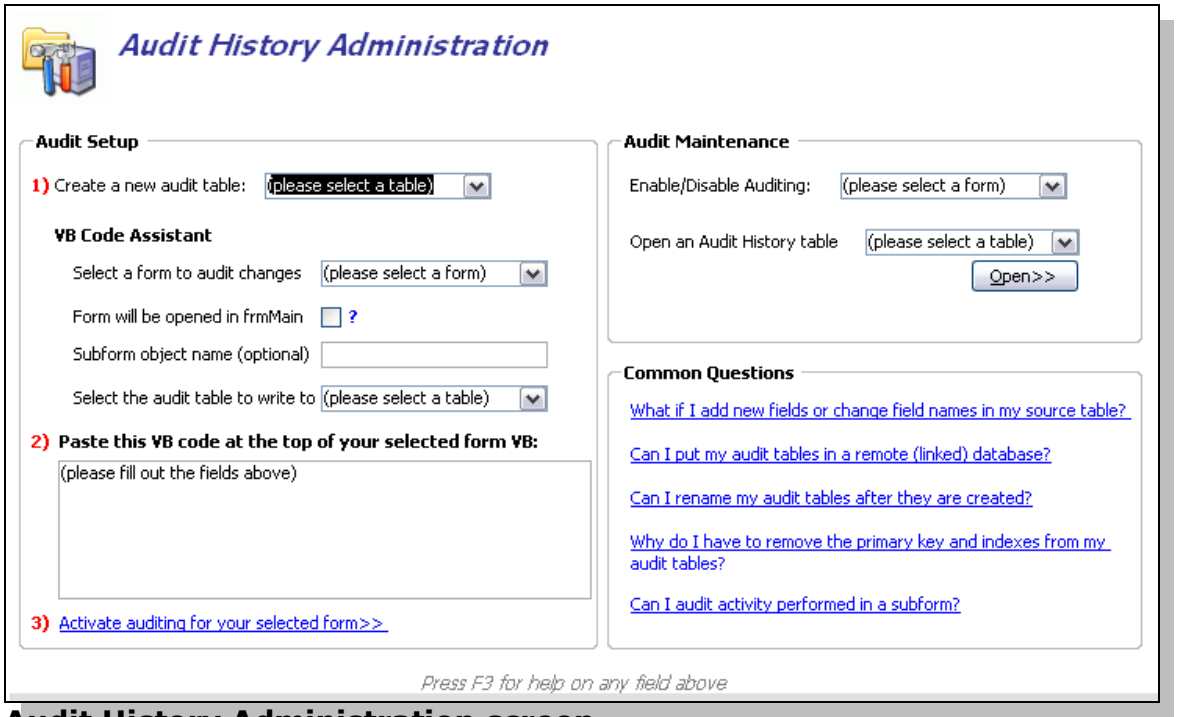
7 Record-Level Auditing

UI Builder provides the core functions needed to audit record-level changes in your forms. Using the step-by-step setup process, you can create individual audit tables that record adds, changes, and deletes made by your users.

It is important to know that the audit history functionality will only work for the forms that you set up audit history tracking. If users have direct access to tables or queries within your database, UI Builder will not be able to create audit history records for those changes. OpenGate Software recommends you take all necessary steps to prevent unauthorized users from gaining direct access to tables and queries where their changes cannot be audited. This includes the following: hiding the database window and preventing the user from using Access Special Keys. Both settings can be located in the Startup Options for your database.[1]

7.1 AUDIT HISTORY SETUP

To set up or manage audit history settings, navigate to the Audit History Administration form by navigating to the main UI Builder Administration form (frmAdministrator), select the Tools tab, and then click the Audit History Setup button. Alternatively, you can open the form frmAdminAudit directly from the database window. For help, you can select a field and then press the F3 button to view help related to that field.



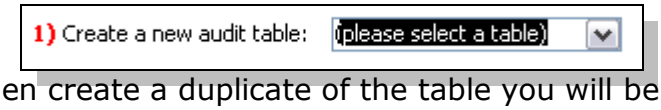
Audit History Administration screen

[1] In Access 2000/2002/2003, select Tools>>Startup…

7.2 CREATING A NEW AUDIT TABLE

7.2.1 CREATING NEW AUDIT TABLES

The first step to setting up audit history tracking is to create a table to store the audit records. To do so, select the table which contains the data you want to audit in Step 1 of the Audit Setup section of the Audit History Administration form.



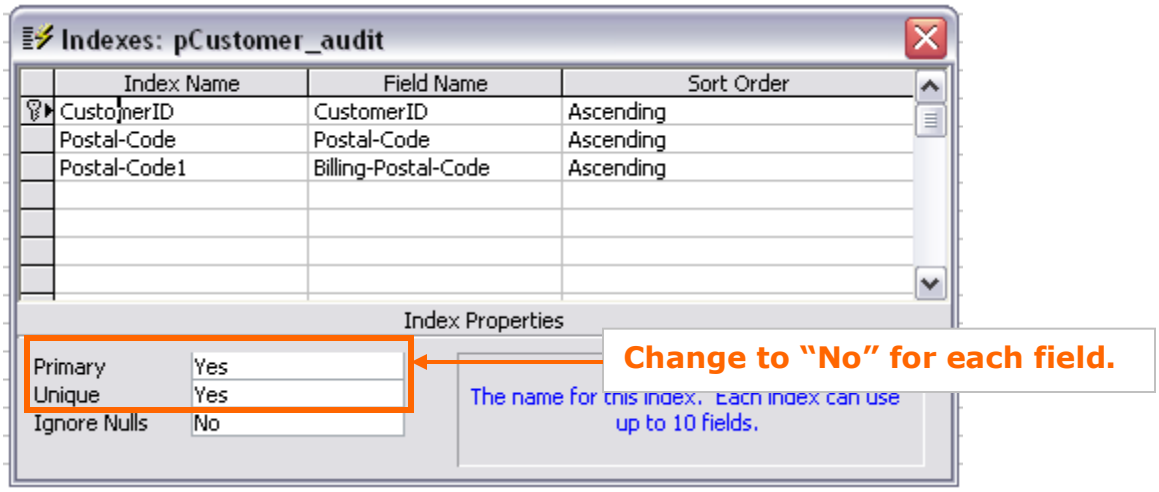
UI Builder will then create a duplicate of the table you will be auditing and automatically add key audit information fields. These fields are as follows:

- **zMachine** – This field will contain the machine from which any changes are made.
- **zUser** – This field will contain the Windows NT username of the individual that made the change.
- **zTimestamp** – This field will contain the timestamp of when the change was made.
- **zAction** – This field will indicate if the record was added, changed, or deleted.
- **zUpdateSource** – This field will indicate

After the audit table is created, which will be named YOURTABLENAME_audit, you will need to remove the primary key and any unique indexes from the audit table. This is to ensure that when audit records are created, the contain the same unique identifier data which will help you locate audit records in the future. For example, if the primary key for your table CUSTOMER is CustomerID, then the audit table will need to contain numerous audit records for the same Customer record which is identified by the same CustomerID. To remove unique indexes and the primary key, follow these steps:

1) When UI Builder opens your new audit table in design mode, click on the indexes icon. 

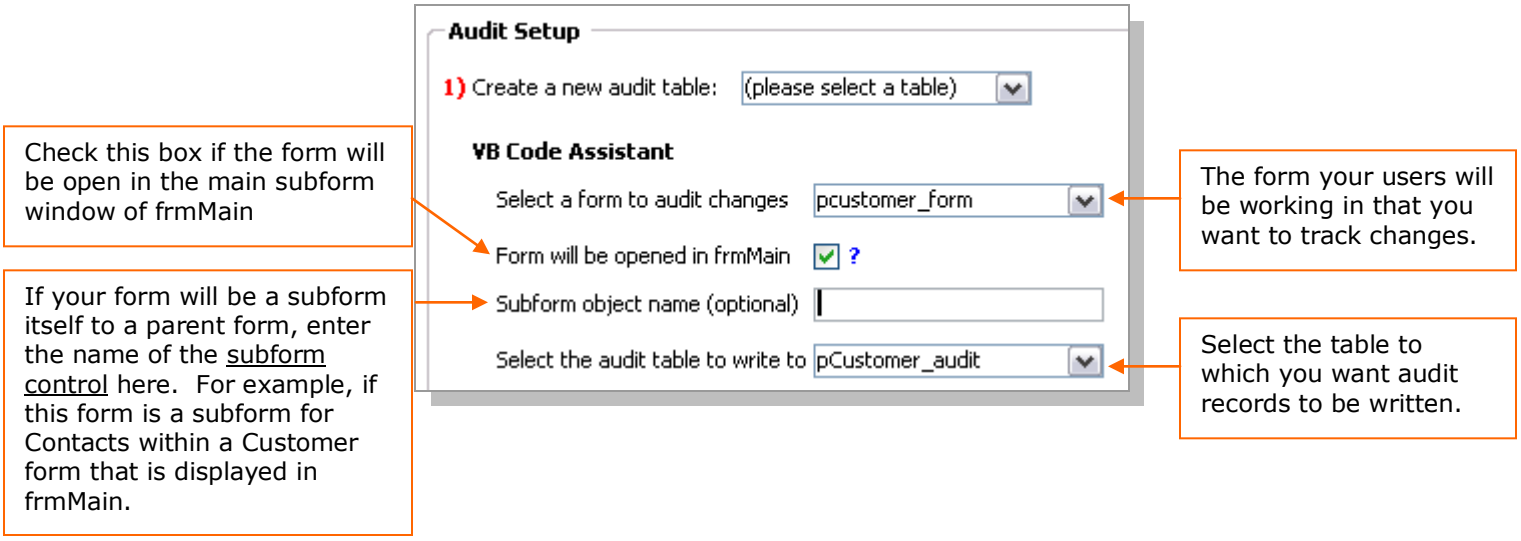
2) In the Indexes dialog, ensure the Primary and Unique attributes for each field listed contains a "No".



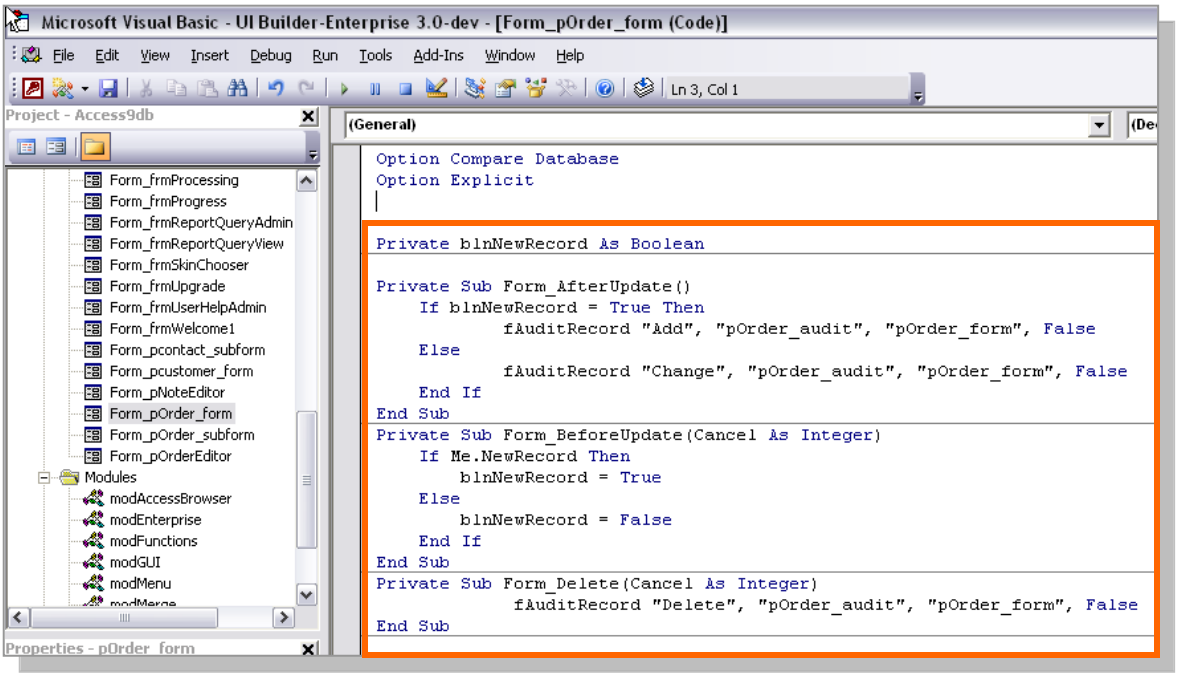
7.2.2 Creating VB code for your form

Once you have created an audit table, you can use UI Builder's VB Code Assistant feature to set up auditing and paste the automatically created VB code into your form by following these steps:

1) Complete each of the fields identified in the VB Code Assistant section as shown below.



2) Copy the text displayed in the "Paste this VB code…" text box into your form. Follow these steps to do so:
   a. Select the form you will be auditing changes in the Access Database Window.
   b. Select the VBA Code Editor icon on the toolbar.
   c. Paste the VB code provide in the VBA code window just below the line beginning with "Option Explicit" as shown below.



VBA Code Editor Window

d. To ensure the pasted code will function without error, select **Debug>>Compile Project** from the menu bar. If you receive an error, please contact support (support@opengatesw.com) with the specific error message and we will be happy to help you resolve the issue.
e. Click **File>>Save** from the menu bar, or the save icon.

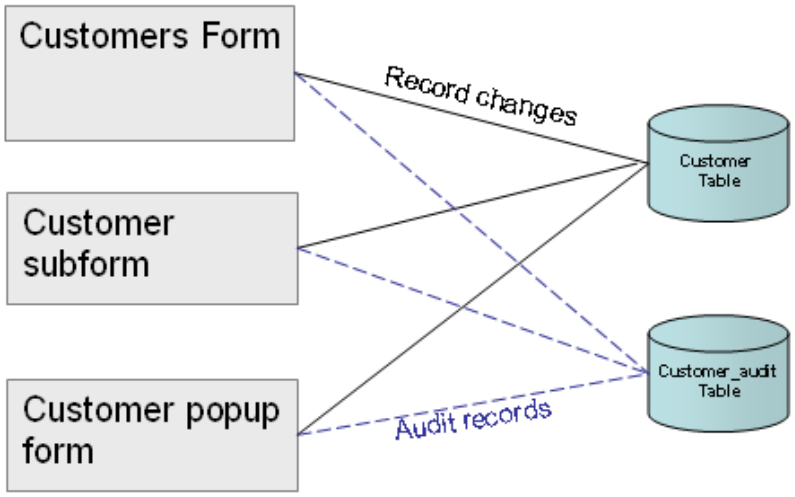
7.2.3 Activating Auditing for your form

Once you have pasted your code and saved changes to your form, select the "Activate Auditing for your selected form>>" option in Step 3 in the Audit Setup section of the Audit History Administration form. This will activate auditing for your form.

> **Important!**
> The form you plan to audit should have every database table field on the form that you want to write to the audit table. The fields do not need to be visible to the user. If a field from the table is not placed on the form, the information stored in that field will not be written to the audit table when an add/change/delete is made by a user.

7.2.4 Activating Auditing for Other forms that use the same table

You can easily add audit history tracking for other forms that use the same table by simply following Steps 1 through 3 of the Audit Setup process. You do not need to create a new audit table. The example below shows three separate forms that all read from and write to a table named "Customers." Each one also has the necessary VB code to write to a table named "Customers_audit."



Example Audit Setup

7.3 MANAGING AUDIT HISTORY TRACKING

To manage audit history settings, navigate to the Audit History Administration form by navigating to the main UI Builder Administration form (frmAdministrator), select the Tools tab, and then click the Audit History Setup button. Alternatively, you can open the form frmAdminAudit directly from the database window. For help, you can select a field and then press the F3 button to view help related to that field.

7.3.1 ENABLING/DISABLING AUDITING

You can enable or disable auditing at the form level by selecting the form name in the Audit Maintenance section of the Audit History Administration screen. If disabled, you do not need to remove the VB code from your form. UI Builder will simply stop writing changes to your designated audit table.

7.3.2 CHANGING AUDIT TABLE NAMES

If you need to rename one or more audit tables, be sure to change the table names in your Visual Basic procedures that call the fAuditRecord() function. The specific variable in the function call is named "strAuditTable." For example, here is a before and after look at the fAuditRecord code when you change a table name:

Before: `fAuditRecord "Add", "pOrder_audit", "pOrder_form", False`

After: `fAuditRecord "Add", "pOrder_history", "pOrder_form", False`

Note that in the example above, the audit table was renamed from "pOrder_audit" to "pOrder_history."

7.3.3 CHANGING AUDIT TABLE FIELDS

You can add new fields to your table, or change existing field names. Be sure to add or change the fields in your audit table as well as the "live" table. Otherwise, UI Builder will skip those fields when writing the audit record (but also log an error event).

Release History – UI Builder Enterprise Edition

Version 1.4
- Initial Release

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
- One-Click Mail Merge
- Create Outlook Tasks
- Repair Broken Table Link capability
- Note Editor
- Added Examples form
- Eliminated main form open background loading with black buttons
- Updated Role-Menu Admin form to include delete button and match functionality of default Menu Admin form
- 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 3.0
- Ability to audit record changes (Record Level Auditing)
- Ability for administrators to test different user roles without changing their NT username.
- New VBA Toolbox function - fTableRecordSum()
- 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
- New Processing form (similar to Google Analytics)
- Minor changes to sample Order and Customer forms
- Added new button for Access 2007 to show or hide the navigation pane from the main form

Version 3.0.1
- Update to "Open Form (read only)" mode to allow users to filter in read-only forms.

- Access 2007 compatibility updates
  - Fix to "Migrate Your Application" buttons
  - Fix to "Apply Settings" feature in Administration form for where the menu buttons changed, but the associated actions did not change until the frmMain form was completely re-opened

Version 3.1

- Enhancements
  - New mass email merge capability.
  - 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.
  - Enhanced user security such that if the user exists but with no role, they don't get to access the database (essentially a way to remove access to the database if you allow anonymous users).
  - The subform administration (frmAdmin_submenu) form's "Action Type" dropdowns now support quick entry without requiring the user to display the full list.
  - Table Link verification will no longer run if the database file extension is ADP/ADE. This is because ADP/ADE databases do not technically have linked tables.
  - Added improved validation to Mail Merge profiles to test that fields specified exist in the data source, general usability of the administration form has also been improved.
  - 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)
  - Fixed issue where user was allowed to delete records even when they were in read-only mode
  - Fixed issue where the user was allowed to open a new form window in edit mode

Version 3.2

- Enhancements
  - New UI Builder login prompt enables you to prompt users to provide a login/password when all users share the same Windows login.
  - 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 Alternate Role popup allows users to switch to different roles if authorized.
  - 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).

- Defect Resolutions

  Fixed issue where menu buttons did not appear if the frmMain form was saved with a button's Visible property set to False.
  Fixed issue where submenu buttons for roles may not appear if you import a copy of your tblRoleMenu table from another version of your database.