Create a user interface (UI) macro

Access for Office 365, Access 2019, Access 2016, Access 2013, Access 2010

In Microsoft Access, macros that are attached to user interface (UI) objects such as command buttons, text boxes, forms, and reports are known as UI macros. This distinguishes them from data macros, which are attached to tables. You use (UI) macros to automate a series of actions, such as opening another object, applying a filter, starting an export operation, and many other tasks. This article introduces you to the newly-redesigned macro builder, and shows you the basic tasks involved in creating a UI macro.

Note: This article doesn't apply to Access web apps.

In this article

Control program flow with If, Else If, and Else

Create submacros

Group related actions together

Expand and collapse macro actions or blocks

Copy and paste macro actions

Share a macro with others

Run a macro

Debug a macro

Convert a macro to VBA code

Overview

Macros can be contained in macro objects (sometimes called standalone macros), or they can be embedded into the event properties of forms, reports, or controls. Embedded macros become part of the object or control in which they are embedded. Macro objects are visible in the Navigation Pane, under **Macros**; embedded macros are not.

Each macro is made up of one or more macro actions. Depending on the context you are working in, some macro actions may not be available for use.

Top of Page

Macro Builder

Here are some of the main feature highlights of the Macro Builder.

- **Action Catalog** Macro actions are organized by type and searchable.
- **IntelliSense** When typing expressions, IntelliSense suggests possible values and lets you select the correct one.

- **Program flow** Create more readable macros with comment lines and groups of actions.
- **Conditional statements** Allow for more complex logic execution with support for nested If/Else/Else If.
- **Macro reuse** The Action Catalog displays other macros you have created, letting you copy them into the one you're working on.
- **Easier sharing** Copy a macro, then paste it as XML into an email, newsgroup post, blog, or code sample web site.

Here's a video that walks you through the main areas of the Macro Builder.

Top of Page

Create a standalone macro

This procedure creates a standalone macro object that will appear under **Macros** in the Navigation Pane. Standalone macros are useful when you want to reuse the macro in many places of the application. By calling the macro from other macros, you can avoid duplicating the same code in multiple places.

1. On the **Create** tab, in the **Macros & Code** group, click **Macro**.

   Access opens the Macro Builder.

2. On the Quick Access Toolbar, click **Save**.

3. In the **Save As** dialog box, type a name for the macro, and then click OK.

4. Continue with the section Add actions to a macro.

Top of Page

Create an embedded macro

This procedure creates a macro that is embedded in an event property of an object. Such a macro does not appear in the Navigation Pane, but can be called from events such as **On Load** or **On Click**.

Because the macro becomes part of the form or report object, embedded macros are recommended for automating tasks that are specific to a particular form or report.

1. In the Navigation Pane, right-click the form or report that will contain the macro, and then click **Layout View**.

2. If the property sheet is not already displayed, press F4 to display it.

3. Click the control or section that contains the event property in which you want to embed the macro. You can also select the control or section (or the entire form or report) by using the drop-down list under **Selection Type** at the top of the property sheet.

4. In the Property Sheet task pane, click the **Event** tab.

5. Click in the property box for the event you want to trigger the macro. For example, for a command button, if you want the macro to run when the button is clicked, click in the **On Click** property box.

6. If the property box contains the words **[Embedded Macro]**, this means a macro has already been created for this event. You can edit the macro by continuing with the remaining steps in this procedure.

7. If the property box contains the words **[Event Procedure]**, this means that a Visual Basic for Applications (VBA) procedure has already been created for this event. Before you can embed a macro in the event, you will need to remove the procedure. You can do this by deleting the words **[Event Procedure]**, but you should first examine the event procedure to make sure that removing it will not break needed functionality in the database. In some cases, you can recreate the functionality of the VBA procedure by using an embedded macro.

8. Click the **Build** button  .

9. If the **Choose Builder** dialog box appears, make sure **Macro Builder** is selected, and then click **OK**.

   Access opens the Macro Builder. Continue with the next section to add actions to the macro.

Top of Page

Add actions to a macro

Actions are the individual commands that make up a macro, and each is named according to what it does, for example, **FindRecord** or **CloseDatabase**.

Step 1: Browse or search for a macro action

The first step in adding an action is finding it in the **Add New Action** drop-down list or in the Action Catalog.

Notes:

- By default, the **Add New Action** drop-down list and the Action Catalog only display the actions that will execute in non-trusted databases. To see all actions:

- If the Action Catalog is not displayed, on the **Design** tab, in the **Show/Hide** group, click **Action Catalog**.

To find an action, use one of the following methods:

- Click the arrow in the **Add New Action** drop-down list, and scroll down to find the action. Program flow elements are listed first, and then the macro actions are listed alphabetically.
- Browse for the action in the Action Catalog pane. The actions are grouped by category. Expand each category to view the actions. If you select an action, a short description of the action appears at the bottom of the Action Catalog.
- Search for the action in the Action Catalog pane by typing in the Search box at the top of the pane. As you type, the list of actions is filtered to show all macros that contain that text. Access searches both the macro names and their descriptions for the text you enter.

Step 2: Add an action to a macro

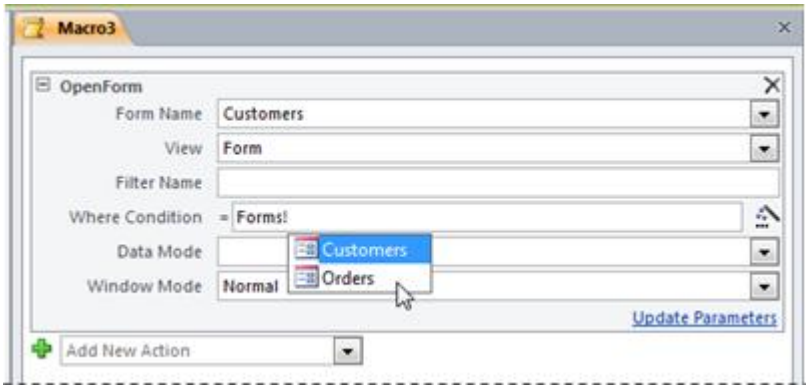
Once you have found the macro action you want, add it to the macro by using one of these methods:

- Select an action in the **Add New Action** list, or just begin typing the action name in the box. Access adds the action at the point where the **Add New Action** list was displayed.
- Drag the action from the Action Catalog to the macro pane. An insertion bar appears to show you where the action will be inserted when you release the mouse button.
- Double-click the action in the Action Catalog.
  - If an action is selected in the macro pane, Access adds the new action just below the selected one.
  - If a **Group**, **If**, **Else If**, **Else**, or **Submacro** block is selected in the macro pane, Access adds the new action to that block.
  - If no action or block is selected in the macro pane, Access adds the new action to the end of the macro.

- If you have already created one or more macros, they are listed under the **In this Database** node in the Action Catalog.
  - Dragging a standalone macro (one that is listed under **Macros**) into the macro pane creates a **RunMacro** action that runs the macro you dragged in. You can then use the drop-down list to call submacros, if present.
  - If you just want to copy the actions from a standalone macro into the current macro (instead of creating a **RunMacro** action), right-click it in the Action Catalog, and then click **Add Copy of Macro**.
  - Dragging an embedded macro (one that is listed under a form or report object) into the macro pane copies the actions from that macro into the current macro.
- You can also create an action by dragging a database object from the Navigation Pane to the macro pane. If you drag a table, query, form, report, or module to the macro pane, Access adds an action that opens the table, query, form, or report. If you drag another macro into the macro pane, Access adds an action that runs the macro.

Step 3: Fill in arguments

Most macro actions require at least one argument. You can view a description of each argument by selecting the action and then moving the pointer over the arguments. For many arguments, you can select a value from a drop-down list. If the argument requires you to type in an expression, IntelliSense helps you enter the expression by suggesting possible values as you type, as shown in the following illustration:



When you see a value that you want to use, add it to your expression by double-clicking it or using the arrow keys to highlight it and then pressing the TAB or ENTER key.

For more information about creating expressions, see the article Introduction to expressions.

When you are creating an embedded UI macro on a web-compatible form, IntelliSense allows you to add any form property to an expression. However, in a web database, only a subset of form properties can be accessed by using UI macros. For example, given a control named Control1 on a form named Form1, IntelliSense will let you add [Forms]![Form1]![Control1].[ControlSource] to an expression in a UI macro. However, if you then publish the database to Access Services, the macro containing that expression will generate an error when it is run on the server.

The following table shows the properties that you can use in UI macros in web databases:

	Properties that you can use
Form	Caption, Dirty, AllowAdditions, AllowDeletions, AllowEdits
Tab Control	Visible
Label	Caption, Visible, Fore Color, Back Color
Attachment	Visible, Enabled
Command Button	Caption, Visible, Enabled, Fore Color
Tex tBox	Enabled, Visible, Locked, Fore Color, Back Color, value
Check Box	Enabled, Visible, Locked, Value
Image	Visible, Back Color
Combo Box	Enabled, Visible, Locked, Value
List Box	Enabled, Visible, Locked, Value

	Properties that you can use
Web Browser	Visible
Subform	Enabled, Visible Locked
Navigation Control	Enabled, Visible

Move an action

Actions are executed in order, from the top to the bottom of the macro. To move an action up or down in the macro, use one of the following methods:

- Drag the action up or down to where you want it.
- Select the action, and then press CTRL + UP ARROW or CTRL + DOWN ARROW.
- Select the action, and then click the **Move Up** or **Move Down** arrow on the right side of the macro pane.

Delete an action

To delete a macro action:

- Select the action, and then press the DELETE key. Alternatively, you can click the **Delete** (X) button on the right side of the macro pane.

Notes:

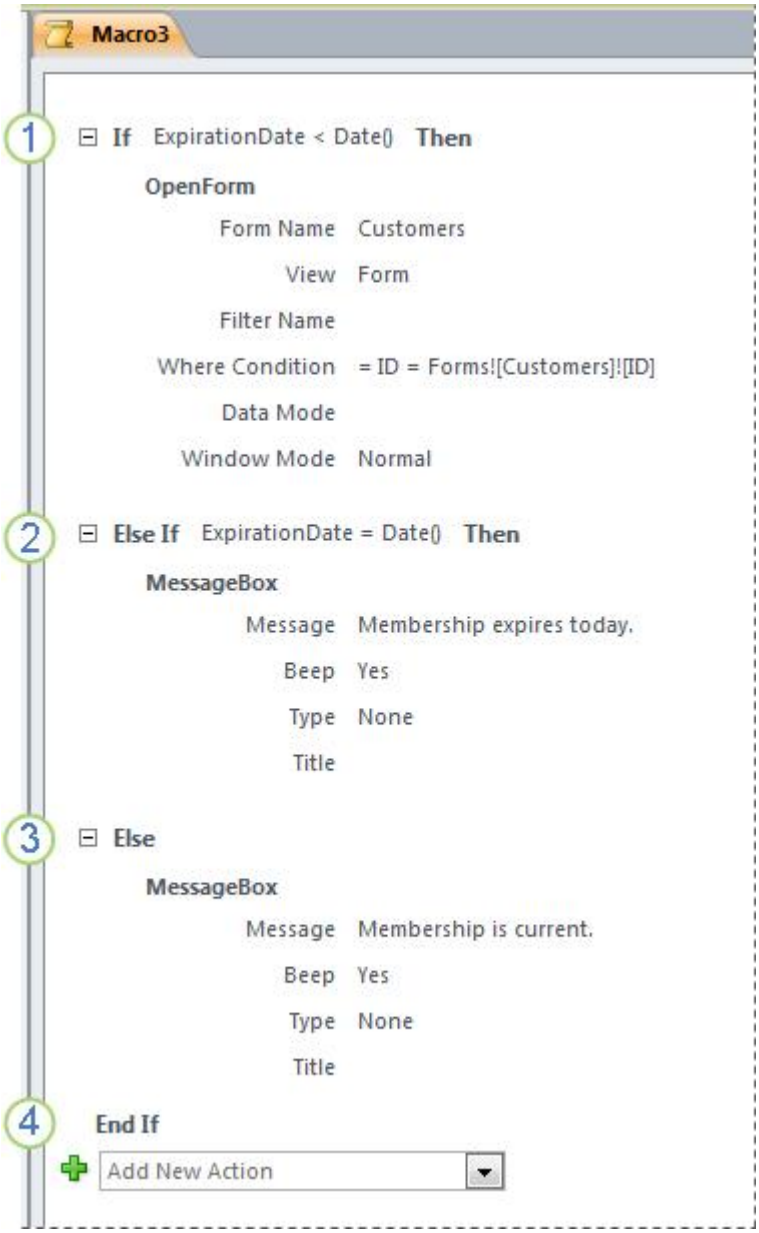
- If you delete a block of actions, such as an **If** block or a **Group** block, all the actions in the block are deleted as well.
- The **Move up**, **Move down**, and **Delete** commands are also available on the shortcut menu that appears when you right-click a macro action.

Top of Page

Control program flow with If, Else If, and Else

To execute macro actions only when certain conditions are true, you use an **If** block. This replaces the **Condition** column that was used in earlier versions of Access. You can extend an **If** block with **Else If** and **Else** blocks, similar to other sequential programming languages like VBA.

The following illustration shows a simple **If** block, including **Else If** and **Else** blocks:



The **Else If** block executes if the ExpirationDate field is equal to the current date.

The **Else** block executes if none of the previous blocks do.

The **If** block ends here.

Add an If block to a macro

1. Select **If** from the **Add New Action** drop-down list, or drag it from the Action Catalog pane to the macro pane.

2. In the box at the top of the **If** block, type an expression that determines when the block will be executed. The expression must be Boolean (that is, one that evaluates to either Yes or No).

3. Add actions to the **If** block by selecting them from the **Add New Action** drop-down list that appears within the block, or by dragging them from the Action Catalog pane to the **If** block.

Add Else or Else If blocks to an If block

1. **Select the If block , and then on the lower right corner of the block, click Add Else** or **Add Else If**.

2. If you are adding an **Else If** block, type an expression that determines when the block will be executed. The expression must be Boolean (that is, one that evaluates to either True or False).

3. Add actions to the **Else If** or **Else** block by selecting them from the **Add New Action** drop-down list that appears within the block, or by dragging them from the Action Catalog pane to the block.

Notes:

- The commands to add **If**, **Else If**, and **Else** blocks are available on the shortcut menu that appears when you right-click a macro action.

Top of Page

Create submacros

Each macro can contain multiple submacros. A submacro is designed be called by name from the **RunMacro** or **OnError** macro actions.

You add a **Submacro** block to a macro in the same way that you a macro action, as described in the section Add actions to a macro. Once you have added a **Submacro** block, you can drag macro actions into it, or select actions from the **Add New Action** list that appears within the block.

Notes:

- You can also create a **Submacro** block by selecting one or more actions, right-clicking them, and then selecting **Make Submacro Block**.
- Submacros must always be the last blocks in a macro; you cannot add any actions (except more submacros) below a submacro. If you run a macro that only contains submacros without specifically naming the submacro you want, only the first submacro will run.
- To call a submacro (for example, in an event property, or by using the **RunMacro** action or **OnError** action), use the following syntax:

  macroname.submacroname

Top of Page

Group related actions together

You can improve the readability of a macro by grouping actions together and assigning a meaningful name to the group. For example, you could group actions that open and filter a form into a group named “Open and filter form.” This makes it easier to see which actions are related to each other. A **Group** block does not affect how the actions are executed, and the group cannot be called or run individually. Its primary use is for labeling a group of actions to help you understand the macro as you read it. In addition, while editing a large macro, you can collapse each group block down to a single line, reducing the amount of scrolling you

If the actions you want to group are already in the macro, use this procedure to add them to a **Group** block:

1. Select the actions that you want to group.
2. Right-click the selected actions, and then click **Make Group Block**.
3. In the box at the top of the **Group** block, type a name for the group.

If the actions are not already present:

1. Drag the **Group** block from the Action Catalog to the macro pane.
2. In the box at the top of the **Group** block, type a name for the group.
3. Drag macro actions from the Action Catalog into the **Group** block, or select actions from the **Add New Action** list that appears within the block.

Group blocks can contain other **Group** blocks, and can be nested up to a maximum of 9 levels deep.

Top of Page

Expand and collapse macro actions or blocks

When you create a new macro, the macro builder displays macro actions with all arguments visible. Depending on the size of the macro, you might want to collapse some or all of the macro actions (and blocks of actions) while you are editing the macro. This makes it easier to get an overall view of the structure of your macro. You can expand some or all of the actions as needed to edit them.

Expand or collapse a single macro action or block

- Click the plus (**+**) or minus (**-**) sign to the left of the macro or block name. Alternatively, press the UP ARROW and DOWN ARROW keys to select an action or block, and then press

Expand or collapse all macro actions (but not blocks)

- On the **Design** tab, in the **Collapse/Expand** group, click **Expand Actions** or **Collapse Actions**.

Expand or collapse all macro actions and blocks

- On the **Design** tab, in the **Collapse/Expand** group, click **Expand All** or **Collapse All**.

> **Tip:** You can "peek" inside a collapsed action by moving the pointer over the action. Access displays the action arguments in a tooltip.

Top of Page

Copy and paste macro actions

If you need to repeat actions that you have already added to a macro, you can copy and paste the existing actions much as you would do with paragraphs of text in a word processor. When you paste actions, they are inserted just below the currently selected action. If a block is selected, the actions are pasted inside the block.

> **Tip:** To quickly duplicate selected actions, hold down the CTRL key and drag the action(s) to the location in the macro where you want them to be copied.

Share a macro with others

When you copy macro actions to the clipboard, they can be pasted as Extensible Markup Language (XML) into any application that accepts text. This enables you to send a macro to a colleague via an e-mail message, or post the macro on a discussion forum, blog, or other Web site. The recipient can then copy the XML and paste it into their Access 2010 Macro Builder. The macro is recreated just as you wrote it.

Run a macro

You can run a macro by using any of the following methods:

- Double-click the macro in the Navigation Pane.
- Call the macro by using the **RunMacro** or **OnError** macro action.
- Enter the macro name in an Event property of an object. The macro will run when that event is triggered.

Top of Page

Debug a macro

If you are having problems getting a macro to run, there are a couple of tools you can use to get to the source of the problem.

Add error-handling actions to a macro

We recommend adding error handling actions to each macro as you write it, and leaving them in the macro permanently. When you use this method, Access displays descriptions of errors as they occur. The error descriptions help you understand the error so that you can correct the problem more quickly.

Use the following procedure to add an error-handling submacro to a macro:

1. Open the macro in Design view.
2. At the bottom of the macro, select **Submacro** from the **Add New Action** drop-down list.
3. In the box just to the right of the word **Submacro**, type a name for the submacro, such as **ErrorHandler**.
4. From the **Add New Action** drop-down list that appears within the **Submacro** block, select the **MessageBox** macro action.

5. In the **Message** box, type the following text: **=[MacroError].[Description]**

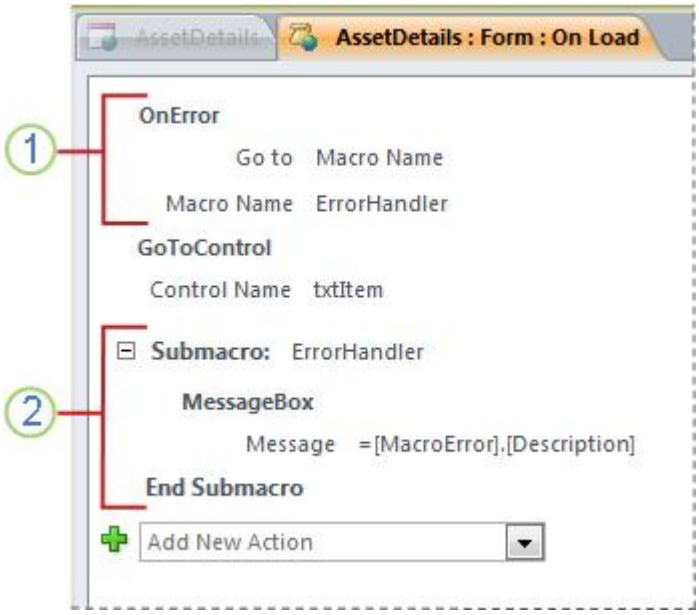
6. At the bottom of the macro, select **OnError** from the **Add New Action** drop-down list.

7. Set the **Go to** argument to **Macro Name**.

8. In the **Macro Name** box, type the name of your error-handling submacro (in this example, **ErrorHandler**).

9. Drag the **OnError** macro action to the very top of the macro.

The following illustration shows a macro with the **OnError** action and a Submacro that is named **ErrorHandler**.



The **OnError** macro action is placed at the top of the macro, and calls the **ErrorHandler** submacro in the event of an error.

The **ErrorHandler** submacro only runs if it is called by the **OnError** action, and displays a message box that describes the error.

Use the Single Step command

Single Step is a macro debugging mode that you can use to execute a macro one action at a time. After each action is performed, a dialog box appears that displays information about the

of the error in the Macro Single Step dialog box, we recommend using the error-handling submacro method described in the previous section.

To start Single Step mode:

1. Open the macro in Design view.
2. On the **Design** tab, in the **Tools** group, click **Single Step**.
3. Save and close the macro.

Next time you run the macro, the **Macro Single Step** dialog box appears. The dialog box displays the following information about each action:

- Macro name
- Condition (for If blocks)
- Action Name
- Arguments
- Error Number (an error number of 0 means no error occurred)

As you step through the actions, click one of the three buttons in the dialog box:

- To see information about the next action in the macro, press **Step**.
- To stop any macros that are currently running, click **Stop All Macros**. Single Step mode will still be in effect the next time you run a macro.
- To exit Single Step mode and continue running the macro, click **Continue**.

Notes:

- If you press **Step** after the last action in a macro, Single Step mode will still be in effect the next time you run a macro.
- To enter Single Step mode while a macro is running, press CTRL+BREAK.

- Single Step mode is not available in a Web database.

Top of Page

Convert a macro to VBA code

Macros provide a subset of the commands that are available in the Visual Basic for Applications (VBA) programming language. If you decide you need more functionality than macros can provide, you can easily convert a standalone macro object to VBA code, and then make use of the expanded feature set that VBA provides. Keep in mind, however, that VBA code will not run in a browser; any VBA code that you add to a Web database will only run when the database is open in Access.

Note: You cannot convert embedded macros to VBA code.

To convert a macro to VBA code:

1. In the Navigation Pane, right-click the macro object and then click Design view.
2. On the **Design** tab, in the **Tools** group, click **Convert Macros to Visual Basic**.
3. In the **Convert macro** dialog box, specify whether you want error handling code and comments added to the VBA module, and then click **Convert**.

Access confirms that the macro was converted, and opens the Visual Basic Editor. Double-click the Converted Macro in the Project pane to view and edit the module.

Top of Page