

IDE Contents

Basics

[Overview: The LotusScript IDE](#)

[Overview: Working in the LotusScript IDE Window](#)

[Overview: Using the Keyboard and Mouse in the IDE](#)

[Overview: Working with Outline Lists in the IDE](#)

[Overview: The Script Editor](#)

[Overview: The Script Debugger](#)

[Overview: Breakpoints, Browser, Output, and Variables Panels](#)

Creating, testing, and compiling scripts

[Overview: Working with \(Globals\) and Objects in the IDE](#)

[Overview: Creating Scripts in the Script Editor](#)

[Overview: Compiling and Testing Scripts in the Script Editor](#)

Setting breakpoints and debugging scripts

[Overview: Using Breakpoints in Debugging](#)

[Overview: Executing Scripts in the Debugger](#)

Using the Breakpoints, Browser, Output, and Variables panels

[Overview: Using the IDE Breakpoints Panel](#)

[Overview: Using the IDE Browser Panel](#)

[Overview: Using the IDE Output Panel](#)

[Overview: Using the IDE Variables Panel](#)

Working with error messages

[IDE Error Messages](#)

[Overview: Getting Help for Error Messages in the IDE](#)

Getting help

[Overview: Online Help in the IDE](#)

[Getting Help on the IDE](#)

[Getting Context-Sensitive Help in the Script Editor, Script Debugger, and Browser](#)

[Opening LotusScript Help in the IDE](#)

[Overview: Getting Help for Error Messages in the IDE](#)

Contents

Search

Print

Overview: The LotusScript IDE

The LotusScript Integrated Development Environment (IDE) is a set of tools you can use to create and debug scripts in Lotus applications.

- The Script Editor lets you write scripts and check their syntax. It also lets you set, clear, enable, and disable breakpoints used in debugging scripts.
- The Script Debugger lets you set, clear, enable, and disable breakpoints and step through scripts to locate the source of problems in script execution.
- The Breakpoints, Browser, Output, and Variables panels help you create and debug scripts:
 - The Breakpoints panel lists breakpoints set in scripts, and lets you navigate to breakpoints in scripts, as well as clear, enable, and disable breakpoints.
 - The Browser panel lists LotusScript keywords, application script components, and type libraries and classes for OLE Automation objects, and lets you paste keywords, component names, and OLE application-class identifiers into a script.
 - The Output panel displays output generated by LotusScript Print commands included in scripts.
 - During debugging, the Variables panel displays information about variables in a script and lets you change variable values.

 [See related topics](#)

Contents

Search

Print

Overview: Working in the LotusScript IDE Window

Each document in an application has its own IDE window. When the document is active, you can open its IDE window and view, write, compile, and debug scripts for document objects. To work with scripts in another document, you must make the other document active and open its IDE window.

When you open the IDE, the Script Editor and the Breakpoints, Browser, and Output panels are displayed in separate panes of the window. When you debug a script, the Script Debugger and Breakpoints, Browser, Output, and Variables panels are displayed.

You can display one pane at a time by dragging the splitter that divides the panes.

When you close or deactivate the IDE, the following are completed automatically: %Rem blocks, multiline strings, object subs, functions, and property blocks, and user-defined types and classes blocks. Scripts are also reformatted to fix lines entered out of sequence or not indented properly, and all scripts for the document are saved.

If you try to close the IDE while the Script Debugger is running, a message lets you know that a script is executing and asks whether you want to stop the script.

 [See related topics](#)

Contents
Search
Print

Closing the IDE Window

To close the IDE, double-click the Control-menu box in the top left corner of the IDE window.

Note The IDE closes automatically when you close its document.

 [See related topics](#)

Contents

Search

Print

Showing and Hiding Panes in the IDE Window

To show and hide panes in the IDE window, drag the pane splitter all the way up or down in the window. For example, to show only the Script Editor, drag the pane splitter down until the Editor fills the window.

To toggle between hidden and visible panes

Press F6.

To restore the default split

Simultaneously press Ctrl and click the pane splitter.

To toggle between single and double panes

Double-click the pane splitter.

 [See related topics](#)

Contents

Search

Print

Switching Between Panes in the IDE Window

To switch focus between the Script Editor or Script Debugger and the Breakpoints, Browser, Output, and Variables panels, press F6 or use Script and Debug menu commands. The pane or panel you switch to becomes the active pane or panel in the IDE window.

To switch to the Breakpoints, Browser, Output, and Variables panels with F6

1. From the Script Editor or Script Debugger, press F6.
Focus switches to the panel that last had focus.
2. To move focus to another panel, click the panel tab or press Shift+Ctrl+→ or Shift+Ctrl+←.

To switch to the Breakpoints, Browser, Output, and Variables panels with menu commands

1. From the Script menu, choose Breakpoints, Browser, Output, or Variables.
Focus switches to the specified panel.
2. To move focus to another panel, click the panel tab or press Shift+Ctrl+→ or Shift+Ctrl+←.

To switch from a panel to the Script Editor or Script Debugger

Do one of the following:

- Press F6.
- Click in the Script Editor or Script Debugger.
- On the Script or Debug menu, choose a command that is performed in the Script Editor or Script Debugger.

 [See related topics](#)

Switching Between IDE Windows

Click a window to give it focus, or use Next on the window Control menu.

When an IDE window loses focus, the following are completed automatically: %Rem blocks, multiline strings, object subs, functions, and property blocks, and user-defined types and classes blocks. Scripts are also reformatted to fix lines entered out of sequence or not indented properly, and all scripts for the document are saved.

 [See related topics](#)

Contents

Search

Print

Overview: Using the Keyboard and Mouse in the IDE

You can use the keyboard to perform commands, move between and within panes and panels in the IDE window, and select text in editable areas of the panes and panels. You can use the mouse to move quickly to any part of the IDE window, show or hide panes in the IDE window, select editable items, expand or collapse outline lists, manage breakpoints, and display right-click menus for areas of the window.

Some general uses of the keyboard

- To switch between the Script Editor or Script Debugger and the last-selected panel (Breakpoint, Browser, Output, or Variables), press F6.
- To display Help for the active area of the IDE or for a script keyword selected in the Script Editor or Browser, press F1.
- To expand or collapse items in an outline list, select the item and press + (plus) or - (minus).
- To move between fields in a pane, press Tab to move forward one field or press Shift+Tab to move backward one field.
- To move the insertion point in the Script Editor to the previous or next procedure that contains user-defined script, press Ctrl+PgDn or Ctrl+PgUp. The insertion point is moved within an object's scripts first and then from one set of object scripts to the next.

Some general uses of the mouse

- To expand or collapse list items, click the expand symbol or the collapse symbol.
- To set a breakpoint on a line, click next to the line in the breakpoint gutter.
- To clear a breakpoint, click the breakpoint in the gutter.
- To disable or enable a breakpoint, simultaneously press Ctrl and click the breakpoint.

 [See related topics](#)

Using the Keyboard in the Script Editor and Script Debugger

To perform commands on the Script and Debug menus

Use the command accelerator keys listed on the menus.

To move the insertion point

→ or ←	Moves the insertion point one character to the right or left.
or ↓	Moves the insertion point up or down one line.
Home	Moves the insertion point to the beginning of a line.
End	Moves the insertion point to the end of a line.
PgUp or PgDn	Moves the insertion point to the beginning or end of a block of script.
Ctrl+→	Moves the insertion point to the next word.
Ctrl+←	Moves the insertion point to the previous word.
Ctrl+	Moves the insertion point to the beginning of the current procedure.
Ctrl+↓	Moves the insertion point to the end of the current procedure.
Ctrl+Home	Moves the insertion point to the beginning of the current procedure.
Ctrl+End	Moves the insertion point to the end of the current procedure.
Ctrl+PgUp	Moves the insertion point to the previous procedure that contains user-defined script.
Ctrl+PgDn	Moves the insertion point to the next procedure that contains user-defined script.

 [See related topics](#)

Contents

Search

Print

Using the Keyboard in the Breakpoints, Browser, Output, and Variables Panels

To delete a breakpoint from the Breakpoints panel


Select the breakpoint item and press Del.

To cancel edits made to values in the Variables panel

Press Esc.

To move to the next panel

Press Shift+Ctrl+→ or Shift+Ctrl+←.

 [See related topics](#)

Contents

Search

Print

Using the Mouse in the IDE

To move the insertion point

Click the editable field you want to move the insertion point to.

To select text in a script

- To select a word, double-click the word.
- To select text from the insertion point to the mouse pointer, press Shift and click.
- To select a script block, click the beginning of the block and drag the highlight to the end of the block.

To view items in an outline list

- To expand an item, click the expand symbol.
- To collapse an item, click the collapse symbol.

To display right-click menus

- In the Script Editor or Script Debugger, right-click the breakpoint gutter or the script area.
- Right-click the Breakpoints or Output panel, or the values text box in the Variables panel.

To control breakpoints from the Script Editor and Script Debugger

- To set a breakpoint on a line, click next to that line in the gutter.
- To clear a breakpoint, click that breakpoint.
- To disable or enable a breakpoint, simultaneously press Ctrl and click that breakpoint.

To control breakpoints from the Breakpoints panel

- To scroll to a breakpoint in the Script Editor or Script Debugger pane and make the breakpoint current without activating the pane, click that breakpoint item in Breakpoints.
- To scroll to a breakpoint in the Script Editor or Script Debugger pane, make the breakpoint current, and make the pane active, double-click that breakpoint item in Breakpoints. The insertion point moves to the beginning of the breakpoint line.

 [See related topics](#)

Contents

Search

Print

Overview: Working with Outline Lists in the IDE

Outline lists contain items that can be expanded or collapsed to display other levels of items. There are several outline lists in the IDE, including the LotusScript Language list in the Browser. This list contains task-oriented categories, which you can expand to display keywords or other expandable subcategories.

Items you can expand are marked with the expand symbol. Items you can collapse are marked with the collapse symbol.

To expand items

- To expand one item, do one of the following:
 - Click the expand symbol for the item.
 - Select the item and press Enter.
 - Select the item and press + (plus).
- To expand all levels in the Browser panel, or the next level down in the Variables panel, simultaneously press Shift and + (plus).

To collapse items

- To collapse one item, do one of the following:
 - Click the collapse symbol for the item.
 - Click the item and press Enter.
 - Click the item and press - (minus).
- To collapse all items, simultaneously press Shift and - (minus).

 [See related topics](#)

Contents

Search

Print

Overview: The Script Editor

In the IDE Script Editor, you can write and edit scripts, check script syntax, and set breakpoints for debugging scripts. When the Script Editor is open:

- The Script menu is displayed in the main menu.
- The Browser, Output, and Breakpoints panels are accessible.
- A script is displayed in the script work area.

Using the Script menu

The Script menu contains commands you can use in the IDE to:

- Create new subs and functions
- Compile and test subs and scripts
- Access the Breakpoints, Browser, Output, and Variables panels

Opening the Script Editor from the main menu

When it is opened from the main menu, the Editor shows the script last edited:

- If no script has been edited, the Editor shows the sub for the current object's default event.
- If no default event is defined, the Editor shows (Declarations) for the object.

Opening the Script Editor from an object

When it is opened from an object's InfoBox or right-click menu, the Editor shows one of the following scripts:

- If the object has been selected during the programming session, the Editor shows the last script selected for that object, whether or not the script has been edited during the session.
- If the object has not been selected during the session, but contains a user-supplied script, the Editor shows (for the object) the first sub, function, or other group of statements that contains a user-supplied script.
- If the object has not been selected and does not contain a user-supplied script, the Editor shows the sub for the object's default event.
- If no default event is defined for the object, the Editor shows (Declarations) for the object.

 [See related topics](#)

Contents

Search

Print

Overview: Setting Script Properties for the IDE

You can set tab stops, enable and disable Smart Indenting, and change text color and font from the Script Editor or Script Debugger. You can also enable and disable a message box that reports the number of errors found during script compilation.

To set these script properties, choose Script Editor Preferences on the File menu and change settings in the preferences box.

Setting tab stops and Smart Indenting

The tab stop setting determines how wide a tab is (in terms of spaces). Smart Indenting indents statements within a block of script by one tab.

Displaying the number of compile-time errors

By default, the number of errors encountered at compile time is displayed in a message box. For example, when you choose one of the Check Scripts commands on the Script menu or an event handler is executed, the number of errors found for the object or the document is displayed. You can disable this message box so that no error count is shown.

Setting colors for script text

By default, comments, keywords, directives, identifiers, and text containing errors are displayed in different colors in the IDE. For example, keywords are displayed in blue and text containing errors is displayed in red. You can change these default colors.

Specifying fonts for script text

The default font for text in the IDE is black 9 point System font on a white background. You can change the font and font size.

 [See related topics](#)

Contents

Search

Print

Overview: Working with (Globals) and Objects in the IDE

The Object drop-down list box in the Script Editor lists the names of objects in the current document that you can associate scripts with. You select an object when you want to enter scripts for the object's events. Once you select the object, you can enter declarations in (Declarations) and define procedures that are used by the events. You can also set up variables declared in (Declarations) in the object's Initialize sub, clean up the variables in the Terminate sub, and enter options for your object scripts in (Options).

You select (Globals) in the Object drop-down list box when you want to enter declarations and procedures that are shared by all scripts in your document. You can set up variables declared in (Declarations) in the (Globals) Initialize sub, clean up variables in the Terminate sub, and enter options for your global scripts in (Options).

Displaying object scripts

When you select (Globals), one of the following is displayed:

- The last (Globals) script you accessed
- (Globals) declarations

When you select an object, one of the following is displayed:

- The last object script you accessed
- The sub for the object's default event
- Object declarations

When the expand symbol is displayed for an object in the list, you can click the symbol to display a group of related objects. If the expandable item is also an object with scripts, you can click the item to display its scripts.

Renaming objects

You cannot rename a product object in the IDE. If the object is renamed through the product, the object's scripts are automatically associated with the new name. You must, however, change references to the old name everywhere they appear in script text.

 [See related topics](#)

Contents

Search

Print

Selecting an Object to Work With in the Script Editor

Click an item in the Object drop-down list box.

If the expand symbol appears next to an item, click the symbol to display a group of related objects; then click one of the objects.

To enter or modify global (Declarations) or procedures, or to enter or modify (Options) for your global scripts

Click (Globals).

To enter or modify (Declarations), (Options), or procedures for an object

Click the object name.

 [See related topics](#)

Contents

Search

Print

Entering (Globals) in the Script Editor

1. In the Object drop-down list box, click (Globals).
2. In any script for (Globals), enter options, declarations, and subs, functions, or other procedures.
Option, *Deftype*, Use, and UseLSX statements are moved to (Options).

The following are moved to (Declarations): Dim statements not explicitly entered inside Sub...End Sub statements; Public, Private, Type, Class, and Declare Lib statements; and Const statements not explicitly entered in (Options).

Sub, Function, Property Set, and Property Get statements not entered inside Class...End Class statements are moved to their own section and their names are added to the Script list for (Globals).

 [See related topics](#)

Contents

Search

Print

Overview: Creating Scripts in the Script Editor

The Script drop-down list box in the Script Editor lists the options, declarations, and subs associated with the object selected in the Object drop-down list box.

By default, an object's (Options) and (Declarations) and event subs are empty. (Globals) (Options) is an exception; it contains the Option Public statement by default, which makes all (Declarations) and procedures you create for (Globals) available to all scripts in the document.

When you create a new sub, function, or other procedure, the IDE inserts its name in the list of user-defined procedures at the bottom of the Script list.

Selecting bold and italicized list items

Items in the Script drop-down list box are displayed in different fonts to help you make selections quickly:

- Predefined items, including (Options), (Declarations), and Initialize, Terminate, and object event subs, are displayed in normal font. These items are displayed in bold font when they have been edited.
- Procedures you create are listed in bold italics.

Entering statements for objects and (Globals)

(Options)

Put Option, Deftype, Use, and UseLSX statements, and Const statements associated with Use and UseLSX, in (Options).

(Declarations)

Put Private, Type, Class, and Declare (external C calls) statements, and Const statements not related to Use or UseLSX statements, in (Declarations). Put Public statements in (Globals) (Declarations). Put module-level Dim statements in (Declarations).

Initialize and Terminate subs

Use the (Globals) Initialize sub to set up data defined in (Globals) (Declarations). This sub executes before other procedures in (Globals) and before variables in (Globals) (Declarations) are accessed.

Use an object's Initialize sub to set up variables defined in the object's (Declarations). This sub executes before the object's events are executed.

Similarly, use the (Globals) and object Terminate subs to clean up variables defined in the corresponding (Declarations). The Terminate sub executes when a script is unloaded. An object script is unloaded when it is edited or when (Globals) is edited. A (Globals) script is unloaded when it is edited. Scripts are also unloaded when the application in which they are executed closes.

Event sub

In an object event sub, put statements you want executed when the event is raised for the object.

Procedure

In a sub, function, or other procedure you create, put statements you want executed when the procedure is called from another script.

 [See related topics](#)

Contents

Search

Print

Adding Comments to a Script in the Script Editor

To enter a one-line comment

Do one of the following:

- Type ' (single quotation mark) at the beginning of a line; then enter the comment line.
- Type Rem at the beginning of a line; then enter the comment text.

To add a comment to a line of script

At the end of a line, do one of the following:

- Add a Rem statement.
- Type ' (single quotation mark), then enter the comment.

To enter a block of comment lines

Use the %Rem...%End Rem directives:

1. Type %Rem at the beginning of a line.
2. Enter the lines of comment text.
3. At the beginning of a new line, type %End Rem.

 [See related topics](#)



Formatting Scripts in the Script Editor

To indent lines of script automatically

With the Script Editor Preferences command on the File menu, set the tab width, and enable Smart Indenting. Then, when you enter lines of script, they are automatically indented.

To create blocks of script automatically

Enter any valid beginning statement for a block structure and press Enter. The corresponding end statement is automatically inserted on a subsequent line.

Beginning and end statements are listed below.

Sub	End Sub
Function	End Function
Property Get	End Property
Property Set	End Property
Do	Loop
For	Next
ForAll	End ForAll
While	Wend
If	End If
Select	End Select
With	End With
Type	End Type
Class	End Class

When you close the IDE, take focus away from the IDE, or check script syntax, the IDE also completes any unterminated %Rem blocks, multiline strings, object subs, functions, and property blocks, and user-defined types and classes blocks. Scripts are also reformatted to fix lines entered out of sequence or not indented properly, and all scripts for the document are saved.



[See related topics](#)



Overview: Creating (Declarations) in the Script Editor

The following statements go in (Declarations): module-level Dim statements, Private, Type, Class, and Declare (external C calls) statements, and Const statements entered outside (Options). Public statements go in (Globals) (Declarations).

When you enter a Type, Class, Declare (external C calls), Dim, Public, or Private statement in any script for an object, the IDE automatically moves the statement to the object's (Declarations). The IDE moves a Const statement to (Declarations) when you enter the Const statement outside (Declarations) or (Options).

The IDE also forward declares object properties, subs, and functions for you.



[See related topics](#)



Entering Type Statements in the Script Editor

Type or paste a valid Type statement in any script for an object.

The IDE formats the Type statement and, if you enter it outside (Declarations), moves the statement to the end of the object's (Declarations).

Note You must enter at least a valid beginning Type statement. The IDE generates an End Type statement if you do not supply one.



[See related topics](#)



Entering Dim Statements in the Script Editor

To declare a variable for a sub

Type or paste a valid Dim statement in the sub.

To declare a variable for all scripts associated with an object

Type or paste a valid Dim or Private statement in the object's (Declarations).

To declare a variable for all scripts in a document

1. Make sure that (Options) for (Globals) contains the Option Public statement.
2. Select (Globals), then (Declarations).
3. Type or paste a valid Dim statement in (Declarations) for (Globals).



[See related topics](#)



Entering Class Statements in the Script Editor

Type or paste a valid Class statement in any script for an object.

The IDE formats the Class statement and, if you enter it outside (Declarations), moves it to the end of the object's (Declarations).

Note You must enter at least a valid beginning Class statement. The IDE generates an End Class statement if you do not supply one.



[See related topics](#)



Overview: Entering Options in the Script Editor

The following statements go in (Options): Option, Deftype, Use, and UseLSX statements, and Const statements associated with Use and UseLSX.

When you enter a valid Option, Deftype, Use, or UseLSX statement in any script for an object (including (Globals)), the IDE automatically moves the statement to the object's (Options).

To put a Const statement in (Options), you must enter it there explicitly; otherwise, the IDE moves the Const statement to (Declarations).

Option and Deftype statements you enter in (Globals) (Options) affect (Globals) scripts only; Option and Deftype statements you enter in an object's (Options) affect that object's scripts only.



[See related topics](#)



Entering Option Statements in the Script Editor

Type or paste a valid Option Base, Option Compare, or Option Declare statement in any (Globals) or object script. The IDE formats the Option statement and moves it to the end of the object's (Options).

Option Public

The IDE automatically includes the Option Public statement as the first statement in (Options) for (Globals). This statement makes all (Declarations) and procedures you create for (Globals) public by default.

The Option Public statement is not allowed in an object's (Options).



[See related topics](#)



Entering Deftype Statements in the Script Editor

Type or paste a valid DefCur, DefDbf, or other *Deftype* statement in any script for an object. The IDE formats the statement and moves it to the end of the object's (Options).



[See related topics](#)



Entering Use and UseLSX Statements in the Script Editor

Type or paste a valid Use or UseLSX statement in any script for an object.

The IDE formats the statement and moves it to the end of the object's (Options).



[See related topics](#)



Overview: Creating Procedures in the Script Editor

The IDE automatically provides empty Sub statement blocks for Initialize and Terminate subs associated with (Globals) and with all scriptable objects in a document. The IDE also provides empty Sub statement blocks for all product-defined object events that you can attach scripts to.

These subs are listed in the Script drop-down list box for an object. The object's event subs are listed by event name. The empty Sub statement block is displayed when you select the event from the list. Statements you enter in a sub are executed when the event is raised for the object.

You can also create subs, functions, and other procedures for objects, including (Globals). The names of the procedures you create are inserted in the list of user-defined procedures at the bottom of the Script list.

Typing a procedure script

One way you can create a procedure is to type or paste a valid Sub, Function, Property Set, or Property Get statement in the script area. The IDE formats a new statement block and provides the appropriate statement terminator. You can then type in statements you want to include in the procedure.

If you create the procedure in (Globals), and you have not deleted the Option Public statement that the IDE puts in (Globals) (Options), your new procedure is public by default.

If you create the procedure for an object, the procedure can be called from the object's Initialize or Terminate sub, object event subs, and other procedures you create for the object.

Note If you add a Sub, Function, Property Get, or Property Set statement inside a sub or function that defines a class method or property, the IDE does not move the statement but displays an error message to let you know that the procedure definition is illegal in that scope.

Using New Sub and New Function

You can also create a sub or a function by choosing New Sub or New Function from the Script menu. In the New Sub or New Function dialog box, you can supply a name for the sub or function and indicate whether you want it to be global. By default, the sub or function is associated with the current object.

Renaming an existing event sub

The IDE also creates a sub for an object when you try to rename one of the object's event subs. Since event subs cannot be renamed, the IDE creates a user-defined sub with the name you specify, moves the script from the event sub to the new sub, and leaves the event sub empty.



[See related topics](#)




Renaming Functions and Subs in the Script Editor

To rename a user-defined sub or function

In the Sub or Function statement, type or paste the new name.

Note You cannot rename product event subs. If you try to rename an event sub, the IDE creates a new user-defined sub with the name you specify and moves the script from the event sub to the new sub.

 [See related topics](#)



Overview: Working with Initialize and Terminate Subs in the Script Editor

Initialize and Terminate subs are predefined in the IDE for (Globals) and for all scriptable product objects in a document.

Statements that you enter in the (Globals) Initialize sub are executed before other procedures in (Globals) and before variables in (Globals) (Declarations) are accessed. Statements that you enter in an object's Initialize sub are executed before the object's events are executed.

Statements in Terminate subs are executed when a script is unloaded. An object script is unloaded when it is edited or when (Globals) is edited. A (Globals) script is unloaded when it is edited. Scripts are also unloaded when the application in which they are executed closes.

Initialize and Terminate subs can contain any LotusScript statements that are valid in subprograms. Invalid statements are reported in compile-time error messages.

Restrictions on using Initialize and Terminate

- There can be only one Initialize sub and only one Terminate sub per object, including (Globals).
- You can set debugging breakpoints in an Initialize sub but not in a Terminate sub.



[See related topics](#)



Overview: Working with Directives in the Script Editor

You can enter %Include and %Rem directives in scripts you create in the IDE, but not %If (%Else, %Elseif, %EndIf) directives. If you want to use %If directives, you must enter them in a file that you call with the %Include directive. When entered directly in IDE scripts, %If directives generate syntax and compile-time errors.

Entering %Rem and %End Rem

You can use the %Rem directive to enter a block of comments in any (Globals) or object script.

If you do not enter %End Rem to terminate the block, the IDE terminates it for you when you close the IDE, take focus away from the IDE, or check script syntax.

Working with %Include

When the IDE detects an error in a file called by %Include, it reports the error at the line that contains the directive. After you change an included file, you must press F2 to recompile all scripts for the object, or Shift+F2 to compile all scripts for the document, because there is no way for the IDE to know that these changes have been made.



[See related topics](#)



Overview: Managing Text in the Script Editor

The script area of the Script Editor is where you enter script statements for a document and for objects in the document. In this area of the Editor, you can perform tasks commonly performed in most text editors: cut, copy, and paste text, enter and delete text, undo edits and, if your application allows, redo edits, import text, and find and replace text.

You can also use Script Editor Preferences on the File menu to control the following script properties: tab stop spacing and line indenting in scripts and the fonts and colors used to display script text.

Automatic formatting

The IDE automatically formats each script so that:

- LotusScript keywords appear initial-capped; directives and the Rem keyword appear uppercased.
- Identifiers and commented lines appear exactly as you enter them.
- Lines beginning with labels are left-justified.
- If Smart Indenting is enabled, lines within a block are indented by one tab stop.

Entering script statements

In general, a line in a script can contain one script statement. You can enter multiple statements on a single line by putting a colon between the statements. When you move from the line, the colons are converted to carriage returns and the statements are split into separate lines. The lines are formatted automatically.

If you enter a colon after an identifier, the identifier is treated as a label.



[See related topics](#)



Selecting Text in the Script Editor

To select text in a script or drop-down list box, use the mouse or the keyboard.

To select text with the mouse

- To select an item in a drop-down list box, click the item.
- To select a word, double-click the word.
- To select a block of script, click the beginning of the block and drag the highlight to the end of the block.
- To select text from the insertion point to the mouse pointer, press Shift and click.

To select text with the keyboard

Shift+→	Selects the character to the right of the insertion point.
Shift+←	Selects the character to the left of the insertion point.
Shift+	Selects one line of text to the left of the insertion point.
Shift+↓	Selects one line of text to the right of the insertion point.
Shift+Home	Selects text from the insertion point to the beginning of the line.
Shift+End	Selects text from the insertion point to the end of the line.
Shift+Ctrl+→	Selects the next word or the rest of the word containing the insertion point.
Shift+Ctrl+←	Selects the previous word or the beginning of the word containing the insertion point.
Shift+Ctrl+	Selects text from the insertion point to the beginning of the current block of script.
Shift+Ctrl+↓	Selects text from the insertion point to the last line in the text display area.



[See related topics](#)



Cutting, Copying, and Pasting Text in the Script Editor

To cut or copy text from a script

Select the text and choose Edit Cut or Edit Copy.

To paste text into a script

Position the insertion point where you want to paste the text and choose Edit Paste.

To paste a keyword from a Browser list into a script

Position the insertion point in the script where you want to paste the keyword, select the keyword in the Browser, and click Paste Name at the top right of the Browser panel.



[See related topics](#)



Entering and Deleting Text in the Script Editor

To enter text in the script area

Do one of the following:

- Position the insertion point where you want to add text and type the text.
- Position the insertion point and choose Edit Paste.

If file importing is supported by the application you are working in, you may also be able to import scripts into the Script Editor and merge them with existing scripts.

To delete text

Select the text and choose Edit Clear or press Del.



[See related topics](#)



Overview: Compiling and Testing Scripts in the Script Editor

As you create scripts for a document, you can test the scripts to make sure they work as you want them to. When you finish entering each line of script and move the insertion point off the line, the IDE tests the syntax and structure of the line and reports errors in the Errors drop-down list box.

You can also test subs and check all scripts for an object or document.

Testing a sub

You can choose Run Current Sub on the Script menu to execute a sub that has no parameters and test its structure and syntax without having to create an object and call the sub from an object event.

Checking all scripts for an object or a document

You can check all scripts for an object or for a document by choosing Check Scripts for Object or Check Scripts for *document* on the Script menu. You should use these commands to force a compile after you change a file that is called from a %Include directive. There is no other way for the IDE to know about or check the changes.

When you use Check Scripts commands, the IDE automatically reformats scripts to fix lines entered out of sequence or not indented properly, and completes any unterminated %Rem statements.

Viewing error messages

The IDE reports all compile-time errors in the Errors drop-down list box. The number of errors found is also displayed in a message box if you enable the box with the Script Editor Preferences command on the File menu.



[See related topics](#)



Testing a Sub in the Script Editor

To execute a sub that has no parameters and check its structure and syntax, select the sub in the Script drop-down list box in the Script Editor, and choose Run Current Sub on the Script menu.

The IDE reports syntax and compile-time errors in the Errors drop-down list box.

Note You cannot use Run Current Sub to test a sub that has parameters.



[See related topics](#)



Checking a Single Line of Script in the Script Editor

To check the syntax of the current line of script, move the insertion point off the line.

Syntax and compile-time errors are reported in the Errors drop-down list box.



[See related topics](#)



Checking Scripts for an Object in the Script Editor

To check all uncompiled scripts for the current object, choose Check Scripts for Object on the Script menu.

Syntax and compile-time errors are reported in the Errors drop-down list box. The number of errors found is also displayed in a message box, if the message box is enabled with the Script Editor Preferences command on the File menu.



[See related topics](#)



Checking Scripts for a Document in the Script Editor

To check all uncompiled scripts for the current document, choose Check Scripts for *document* on the Script menu.

Syntax and compile-time errors are reported in the Errors drop-down list box. The number of errors found is also displayed in a message box, if the message box is enabled with the Script Editor Preferences command on the File menu.



[See related topics](#)



Finding and Fixing Compile-Time Errors in the Script Editor

Compile-time errors are reported in the Errors drop-down list box in the following format:

object: procedure: line#: message

To correct an error

1. Click the error message in the Errors drop-down list box.
The IDE scrolls to the script line that contains the error.
2. Correct the error.

To correct an error in an included file

1. Double-click the error message in the Errors drop-down list box.
The IDE scrolls to the line that contains the %Include directive for the file.
2. Open the file and correct the error.



[See related topics](#)



Overview: The Script Debugger

In the IDE Script Debugger, you can set, clear, enable, and disable breakpoints and step through scripts to locate the source of problems in script execution. The Debugger opens when the first enabled breakpoint is encountered during script execution. The Debugger for a document also opens when one of its procedures is called from an executing script in another document.

When the Script Debugger is open:

- The Debug menu is displayed in the main menu.
- The Browser, Output, Breakpoints, and Variables panels are accessible.
- The script containing the breakpoint is displayed in the Debugger, and the breakpoint line is the current line.

Using the Debug menu

The Debug menu contains commands you can use in the IDE to:

- Set, clear, enable, and disable breakpoints for debugging
- Step through a script and stop script execution



[See related topics](#)








Overview: Using Breakpoints in Debugging



You can set breakpoints in the Script Editor and Script Debugger on lines in all subs except the Terminate sub. When you set a breakpoint, it is, by default, enabled. When you are editing or debugging a script, you can disable, enable, and clear the breakpoint. You can also set other breakpoints.

Note For a continued line, a breakpoint is set on the last line.









Breakpoint symbols

-  Breakpoint
-  Current breakpoint
-  Disabled breakpoint
-  Current disabled breakpoint
-  No breakpoint

Current line indicators

-  Top current line
-  Nested current line

Combined breakpoint symbols and current line indicator

-  Disabled breakpoint, top current line
-  Current disabled breakpoint, top current line
-  Disabled breakpoint, nested current line
-  Current disabled breakpoint, nested current line
-  Breakpoint, top current line
-  Current breakpoint, top current line
-  Breakpoint, nested current line
-  Current breakpoint, nested current line



[See related topics](#)



Setting and Clearing Breakpoints in the IDE

To set and clear breakpoints in the Script Editor or Script Debugger

- To set a breakpoint on a line, click next to the line in the breakpoint gutter.
- To clear a breakpoint, click the breakpoint symbol in the breakpoint gutter.

To set and clear breakpoints using the Debug menu

- To set a breakpoint on a line, click anywhere in the line and choose Set Breakpoint in the Debug menu.
- To clear a breakpoint on a line, click anywhere in the line and choose Clear Breakpoint in the Debug menu.
- To clear all breakpoints for a document, choose Clear Breakpoints for *document* in the Debug menu.

To clear breakpoints using the Breakpoints panel list

Select a breakpoint in the breakpoints list, choose and press Del or choose Clear Breakpoint in the Debug menu.



[See related topics](#)



Enabling and Disabling Breakpoints in the IDE

To enable or disable breakpoints using the Debug menu

- To enable a breakpoint on a line, click anywhere in the line and choose Enable Breakpoint in the Debug menu.
- To disable a breakpoint on a line, click anywhere in the line and choose Disable Breakpoint in the Debug menu.
- To enable or disable all breakpoints for a document, choose Enable Breakpoints for *document* or Disable Breakpoints for *document* in the Debug menu.

To enable or disable breakpoints using the Breakpoints panel list

Select a breakpoint in the breakpoints list and choose Enable Breakpoint or Disable Breakpoint in the Debug menu.

To enable or disable breakpoints with the keyboard and mouse

Simultaneously press Ctrl and click the breakpoint symbol in the breakpoint gutter.



[See related topics](#)



Overview: Executing Scripts in the Debugger

The Script Debugger is opened when an enabled breakpoint is reached in an executing script. Once the Debugger is open, you can use commands in the Debug menu to step through a script, step into or over a procedure called in a breakpoint line, exit the procedure, continue script execution, or stop script execution.

You can also view information about the executing script in the Output and Variables panels.

Working with more than one Debugger

When a script you are debugging in one document steps into a script that belongs to another, unopened document, the Debugger for the second document opens and displays its script.

If Script Editors are already open for several documents when a breakpoint is reached in the executing document, Debuggers are displayed in place of the Editors, whether or not scripts for those documents are executing. The Calls drop-down box for each Debugger lists all executing procedures, regardless of the documents they belong to.

If script execution leads to a procedure for which source is not available, stepping continues until a displayable statement is reached or the script ends.



[See related topics](#)



Stepping Through a Script in the Debugger

To step through procedures in a script

When script execution stops at a line, choose Step in the Debug menu. The next procedure is executed. If the procedure contains a call, script execution steps into the called procedure.

To step out of the current procedure

When execution of a procedure stops, choose Step Exit in the Debug menu to exit from the procedure. When you exit from a called procedure, script execution steps into the line after the line containing the call. Otherwise, script execution continues until the next breakpoint is reached or until the script ends.

To step over the procedure called in a breakpoint line

When script execution stops at the line, choose Step Over in the Debug menu. The line and the procedure it calls are executed, and script execution steps to the line after the breakpoint line.



[See related topics](#)



Continuing and Stopping Script Execution in the Debugger

To continue script execution from a breakpoint

Choose Continue Execution in the Debug menu. Script execution continues to the next breakpoint or to the end of the script.

To stop script execution

Choose Stop Execution in the Debug menu. Script execution stops at the current breakpoint. If the Script Editor was open when debugging started, the Script Editor is redisplayed when debugging stops; otherwise, the Script Debugger closes.



[See related topics](#)



Overview: Switching to Another Script in the Debugger

When an enabled breakpoint is reached during debugging, script execution is stopped and the script for the procedure containing the breakpoint is displayed in the Script Debugger. You can display another procedure by selecting it from the Calls drop-down list box at the bottom of the Debugger pane, or from the Script drop-down list box at the top right.

Using the Calls drop-down list box

You can switch to another procedure in the Script Debugger execution stack by selecting it from the Calls drop-down list box.

The Calls box lists the procedures that are currently executing, including procedures that belong to other documents. Each list item contains the name of the object the procedure is associated with and the name of the procedure. When you select a procedure in the list, its script is displayed in the script area of the Debugger.

A procedure that belongs to another document is dimmed in the list. A procedure whose source is not available (for example, an external script that is called from the procedure) is also dimmed. You cannot select these procedures.

When you select a procedure, its variables and their values are displayed in the Variables panel. You can use the Variables panel to view information about a variable or change its value in the executing procedure.

Using the Script drop-down list box

If you want to switch to a procedure that is not in the execution stack, you can use the Object and Script drop-down list boxes to navigate to the procedure. The script you select is displayed in the Script Debugger, but the Calls drop-down box and the Variables panel continue to display information for the executing procedure last selected in the Calls box.



[See related topics](#)




Selecting a Script from the Calls Drop-down List in the Debugger

1. Click the Calls drop-down list box to display a list of procedures currently in the execution stack.
2. Click the procedure you want to display in the Script Debugger.

The procedure script is displayed in the Debugger script area, and information about procedure variables is listed in the Variables panel.

Note You cannot change a script in the Debugger script area; you must stop script execution and make changes in the Script Editor. You can change variable values in the Variables panel.

 [See related topics](#)



Selecting a Script from the Script List in the Debugger

1. Click the Object drop-down list box to display a list of objects in the document.
2. Select the object whose script you want to display.
3. Click the Script drop-down list box to display a list of scripts for the object.
4. Select the script you want to look at.

The script is displayed in the Debugger script area.

Note You cannot change a script in the Debugger script area; you must stop script execution and make changes in the Script Editor. You can change variable values in the Variables panel.



[See related topics](#)



Overview: Fixing Errors Reported in the Script Debugger

When a run-time error occurs, the error is reported in a message box, and the current line indicator in the Script Debugger is positioned at the line containing the error. The format of the message is shown below:

object: procedure: line#: message

Press F1 to view Help for the message.

All Debug menu items are disabled except Stop Execution. When you choose Stop Execution, the script containing the error is displayed in the Script Editor if the Editor was open when debugging started; otherwise, the Script Debugger is closed.

When a compile-time error is encountered during debugging, script execution is terminated, the script containing the error is displayed in the Script Editor, and a message about the error is displayed in the Errors drop-down list box. You can view Help for the message by selecting the message and pressing F1.



[See related topics](#)



Overview: Selecting and Copying Text in the Script Debugger

You can select text and copy it to the Clipboard in the same way you select and copy it in the Script Editor; for example, by dragging the mouse over the text you want to copy and choosing Edit Copy.

You cannot type or paste text into a script, and you cannot cut text from a script, in the Script Debugger.



[See related topics](#)



Overview: Closing the Script Debugger

You can close the Script Debugger for a document as long as the document doesn't contain the currently executing script. If you try to close the Debugger while its script is executing, a message tells you that closing the Debugger will stop script execution and asks you if you want to continue. If you click OK, script execution is stopped, and the Debugger is closed.

Closing a Debugger for the executing script affects other open Debuggers as follows:

- Debuggers that replaced Script Editors (when the Debugger was opened for the executing script) revert to being Editors.
- Debuggers that were opened automatically during script execution are closed.

You can close a Debugger by choosing Stop Execution in the Debug menu, by double-clicking the Control-menu box in the top left corner of the Debugger window, or by closing the document in which the Debugger is open.



[See related topics](#)



Overview: Breakpoints, Browser, Output, and Variables Panels

The Breakpoints, Browser, Output, and Variables panels display information about scripts that you create and test in the Script Editor and the Script Debugger. You can use them to change scripts and to manage breakpoints for debugging scripts.

Browser panel

The Browser panel displays lists of keywords, application script components, and type libraries and classes for OLE Automation objects. You can select items in these lists and paste their names into the current script. The Browser panel is selected by default when the Script Editor is active. You can view Browser listings when the Script Editor or the Script Debugger is active; you can select items and paste them into the current script only when the Script Editor is active.

Breakpoints panel

The Breakpoints panel lists breakpoints set in scripts, and lets you navigate to breakpoint lines in scripts, as well as clear, enable, and disable breakpoints. You can use the Breakpoints panel when the Script Editor or the Script Debugger is active.

Output panel

The Output panel displays output generated by LotusScript Print statements included in scripts. This output is generated when scripts execute and can be viewed when the Script Editor or the Script Debugger is active.

Variables panel

The Variables panel displays information about variables in a script and lets you change variable values. The Variables panel is available only when the Script Debugger is active.



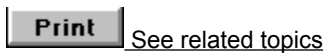
[See related topics](#)



Overview: Using the IDE Breakpoints Panel

The Breakpoints panel displays a list of breakpoints set for all scripts in a document. You can use this list to see which objects, procedures, and lines contain breakpoints, to determine whether a breakpoint is enabled, and to clear, enable, and disable breakpoints.

You can also navigate to a breakpoint line in the Script Editor or Script Debugger by clicking its entry in the Breakpoints list. You can navigate to the breakpoint line and make the Script Editor or Script Debugger active by double-clicking the Breakpoints entry or selecting the entry and pressing Enter.





Overview: Viewing a List of Breakpoints in the Breakpoints Panel

Breakpoints are listed in the order in which they are entered in a document's scripts. Information for each breakpoint is displayed in the following format:

object: procedure: line# (Disabled)



[See related topics](#)



Overview: Managing Breakpoints from the Breakpoints Panel

From the Breakpoints panel, you can navigate to a breakpoint line in the Script Editor or Script Debugger, you can remove a breakpoint from a line of script, and you can enable and disable a breakpoint.

Scrolling to lines in the Script Editor or Script Debugger

You can click an item in the Breakpoints list to navigate to a breakpoint line in the Script Editor or Script Debugger:

- Clicking a breakpoint item in the list scrolls to the breakpoint line in the Script Editor or Script Debugger without making the Editor or Debugger active.
- Double-clicking a breakpoint item (or selecting the item and pressing Enter) navigates to the breakpoint line in the Script Editor or Script Debugger, puts the insertion point at the beginning of the line, and makes the Editor or Debugger active.

Removing breakpoints

To remove a breakpoint, select it in the Breakpoints panel and press Del or choose Clear Breakpoint on the Debug menu.

Clearing a breakpoint does not change the pane focus or the script displayed in the Script Editor or Script Debugger.

Enabling and disabling breakpoints

To enable or disable a breakpoint, select it in the Breakpoints panel and choose Enable Breakpoint or Disable Breakpoint on the Debug menu.



[See related topics](#)



Overview: Using the IDE Browser Panel

The Browser displays LotusScript keywords and statement syntax, application script components, and type libraries and classes for OLE Automation objects, and lets you paste keywords, component names, and OLE application-class identifiers into the current script.

The Browser panel is available from the Script Editor and the Script Debugger, but you can select items and paste them into the current script only when the Script Editor is active.

When you are working in the Browser, you can display Help for a LotusScript keyword or other list item by selecting the keyword or item and pressing F1.



[See related topics](#)



Viewing LotusScript Keywords in the Browser

You can display LotusScript keyword syntax in the Browser when the Script Editor or Script Debugger is active. You can also display Help for a keyword by selecting it and pressing F1.

When the Script Editor is active, you can select a keyword to paste into the current script.

To display keyword syntax

1. In the Category drop-down list box, choose LotusScript Language.
2. To display a list of LotusScript keywords in alphabetic order, expand All.
To display a list of keywords for another category, expand the category.

To paste a keyword into the current script

1. Select a keyword entry in a category list.
2. Click Paste Name at the top right of the Browser panel.
The keyword is pasted into the current script in the Script Editor.



[See related topics](#)



Viewing Information about Application Components in the Browser

You can display lists of application classes, constants, variables, subs, and functions in the Browser when the Script Editor or the Script Debugger is active. You can paste their names into the current script only when the Script Editor is active.

To display a list of classes, constants, variables, or subs and functions

1. In the Category drop-down list box, choose an application category.
2. If there are subcategories, expand the one(s) you want to look at.

To paste a component name into the current script

1. Select an item from an expanded category.
2. Click Paste Name at the top right of the Browser panel.

The component name is pasted into the current script in the Script Editor.



[See related topics](#)



Viewing Registered Type Libraries for OLE Classes in the Browser

You can display lists of OLE Automation object classes in the Browser when the Script Editor or the Script Debugger is active. If a class can be instantiated in scripts, you can also paste its application-class identifier into the current script when the Script Editor is active.

Note When Help is associated with an OLE class, you can display Help for the class by selecting the class name in the Script Editor, Script Debugger, or Browser and pressing F1.

To display information about OLE classes

1. In the Category drop-down list box, choose OLE Classes.
2. From the list of registered OLE type libraries, select the one you want to look at.
3. From the list of classes in the type library, select the class you want to view information for; then select Properties or Methods to view a list of properties or methods for the class.

To paste an OLE class identifier into the current script

When an OLE class can be instantiated in a script with the CreateObject function, the application-class identifier for the class is shown in parentheses next to the class name.

1. Select the class you want to work with in the current script.
2. Click Paste Name at the top right of the Browser panel.
The application-class identifier is pasted into the script.



[See related topics](#)



Pasting from the Browser into a Script

When the Script Editor is active, you can paste a list item from the Browser into the current script.

1. In a Browser list, select an entry for a LotusScript language keyword, OLE class, or application component.
2. Click Paste Name at the top right of the Browser panel.

The keyword, OLE application-class identifier, or application component name you selected is pasted into the current script in the Script Editor.

In the script, complete the statement for the keyword, identifier, or name.



[See related topics](#)




Overview: Using the IDE Output Panel

When you execute scripts in a document, up to 1024 characters of output from each Print statement included in the scripts is displayed in the Output panel. This output is displayed in the order it is generated in the document.

You can review the output list to determine whether scripts are executing as you want them to.

When the document is closed, up to 500 lines of output are saved for the document.

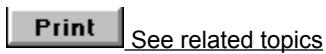
 [See related topics](#)



Overview: Using the IDE Variables Panel

The Variables panel displays information about variables in a script and lets you change variable values. This panel is active during debugging only and displays information about variables in the current script (that is, the script for the object and procedure shown in the Calls drop-down list in the Script Debugger).

Variables are listed by variable name in the order they are declared. The variable's value and current data type are also listed. When a variable is a type with members, the variable is presented as an expandable entry and its name and type are listed. Expand the entry to view its members and their values and data types.





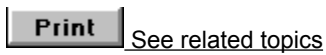
Viewing Variables for the Current Script

1. In the Calls drop-down list box in the Script Debugger, select the procedure whose variables you want to look at.
2. In the Variables panel, expand variable entries (as necessary) to display their members.

Member variables and their values and data types are displayed.

If a value contains more than 42 characters, the first 40 characters, plus ... (ellipses), are displayed.

Note When a list displays the contents of a collection, numbers appear next to the values. These numbers represent the order of the contents; they cannot be used as indexes to the collection.





Viewing Globals for the Current Script

Expand the Globals entry in the Variables panel.

The following variables and their values and data types are displayed:

- Variables defined in (Globals) and referenced in the current script
- Variables defined in modules called by Use and UseLSX statements in the current script
- Variables declared in the current object's (Declarations)



[See related topics](#)

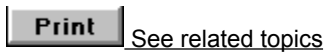


Changing Variable Values in the IDE

You can change a variable's value when you are debugging a script.

To change a variable value during debugging

1. In the Calls drop-down list box in the Script Debugger, select the procedure that contains the value definition.
2. In the Variables panel, select the variable whose value you want to change.
If the variable value can be changed, the value text box is enabled.
3. In the value text box, enter a new value for the variable.
If a variable contains more than 1024 characters, you can view and edit only the first 1024 characters in the text box.
4. Press Enter or click the check mark next to the text box.
The value is changed for the variable.





Overview: Online Help in the IDE

Help provides information about the IDE and the ways you can use it to create and test scripts. You can access this information by pressing F1 to display context-sensitive Help for the active part of the IDE window, or by using the Help menu to select a topic.

Context-sensitive Help

In the IDE, context-sensitive Help is available for:

- The Script Editor and Script Debugger panes
- The Browser, Breakpoints, Output, and Variables panels
- LotusScript keywords and application components (and OLE Automation classes for which Help is available) displayed in the Script Editor, Script Debugger, and Browser
- Compile-time messages displayed in the Errors drop-down list box in the Script Editor
- Run-time messages displayed in message boxes in the Script Debugger

Help Contents and Search

You can also use the IDE Help Contents topic and Search to view Help for error messages and to find out how to use menus, the keyboard and mouse, the Object and Script drop-down list boxes, and the Breakpoints, Browser, Output, and Variables panels to create and manage scripts.



[See related topics](#)




Opening LotusScript Help in the IDE

In the Script Editor or Script Debugger

1. Select a LotusScript keyword.
2. Press F1.
Help for the keyword is displayed.

In the Browser

1. Select an entry for a LotusScript keyword.
2. Press F1.
Help for the keyword is displayed.

 [See related topics](#)



Getting Context-Sensitive Help in the Script Editor, Debugger, and Browser

In the Script Editor or Script Debugger

1. Select a LotusScript keyword or the name of an application class, method, property, or other component.
2. Press F1.
Help for the keyword or application component is displayed.

In the Browser

1. Select an entry for a LotusScript keyword or an entry for an application class, method, property, or other component.
2. Press F1.
Help for the keyword or component is displayed.



[See related topics](#)



Overview: Getting Help for Error Messages in the IDE

Context-sensitive Help is available for compile-time and run-time messages displayed in the IDE. Compile-time messages are displayed in the Errors drop-down list box in the Script Editor. To view Help for one of these messages, select the message and press F1.

Run-time messages are displayed in message boxes. To view Help for one of these messages, press F1 while the message box is displayed.

You can also view Help for these messages when you are in IDE Help by searching for "messages" and going to the IDE Error Messages topic. This topic contains a list of compile-time and run-time messages. When you select a message, Help for the message is displayed.



[See related topics](#)



Getting Help on the IDE

To display general information for the Script Editor or Script Debugger

Click anywhere in the Script Editor or Script Debugger pane (except in a script) and press F1.

To display general information for the Breakpoint, Browser, Output, and Variables panels

Click anywhere in a panel and press F1.

If you click an item in a Browser list and press F1, Help for the item is displayed.

To display information about menus, drop-down list boxes, and activities you can perform

Use Help Search or the IDE Help Contents topic.



[See related topics](#)



IDE Error Messages

Click a message to display Help.

[.. not valid outside of class scope](#)

A - E

[Cannot forward declare user-defined class or data type](#)

[CASE ELSE must be the last CASE in a SELECT statement](#)

[Compiler stack overflow at: <token name>](#)

[Compiler statement stack overflow at: <token name>](#)

[DIM required on declarations in this scope](#)

[Duplicate procedure name: <procedure name>](#)

F - J

[Illegal character after %INCLUDE directive](#)

[Illegal character after continuation character](#)

[Illegal directive](#)

[Illegal duplicate END statement](#)

[Illegal executable code in Declarations](#)

[Illegal executable code in Options](#)

[Illegal executable code outside procedure](#)

[Illegal on declarations in this scope](#)

[Illegal range specifier](#)

[Illegal statement](#)

[Illegal type suffix on keyword: <keyword>](#)

[Illegal use of escape character](#)

[Illegal use of escape character in identifier](#)

[Illegal use of parentheses](#)

[Invalid type for procedure](#)

K - O

[ME not valid outside of class scope](#)

[Name too long](#)

[Named product class instance not valid here](#)

P - T

[Out of memory](#)

[Procedure definitions illegal in this scope](#)

[Procedures may not be forward declared](#)

[PUBLIC not allowed in this module](#)

[SET required on class instance assignment](#)

[Statement illegal in CLASS block: <keyword>](#)

[Statement illegal in TYPE block: <keyword>](#)

[Statement is illegal in this scope](#)

[Syntax checking buffer overflow](#)

U - Z

Unexpected: <token>; Expected: <token>

Unmatched block terminator

Unterminated block statement

Unterminated square bracket reference

Unterminated string constant

Unterminated <keyword> block



[See related topics](#)



Duplicate procedure name: <procedure name>

You assigned the same name to more than one Sub, Function, Type, Property Set, or Property Get statement assigned to an object or (Globals). No two subs assigned to an object or (Globals) can have the same name, no two functions can have the same name, and so on.

Remove the duplicate name.



[See related topics](#)




Illegal executable code in Declarations

In (Declarations), you included a statement that is not allowed in the section.

Only the following are allowed in object (Declarations): comments, the Private keyword, the Declare (external C calls), and the Const, Dim, Type, and class statements.

(Global) (Declarations) can also contain the Public keyword.

Remove the invalid statements from (Declarations).

 [See related topics](#)



Illegal statement

You used a colon (:), followed by an underscore (_), to separate and then recombine statements in a line of script. This is not allowed in the IDE.

Use a colon (:) between multiple statements in a line; use an underscore (_) at the end of a line to continue the line.



[See related topics](#)



Procedures may not be forward declared

You tried to use the Declare statement to forward declare a sub, function, or property for an object. This is not necessary because the IDE generates forward declares for you.



[See related topics](#)



Cannot forward declare user-defined class or data type

You tried to use the Declare statement to declare a type or class before defining it. This is not allowed in the IDE.



[See related topics](#)



Unterminated block statement

You omitted the ending statement for a block statement that begins with one of the following keywords:

Class

Do

For

ForAll

Function

If...Then...Else...EndIf

Property Get

Property Set

Select

Sub

Type

While

With

Enter the appropriate ending statement.



[See related topics](#)



Unmatched block terminator

You omitted the statement that begins with one of the following keywords and marks the beginning of a statement block:

Class

Do

For

ForAll

Function

If...Then...Else...EndIf

Property Get

Property Set

Select

Sub

Type

While

With

Enter the appropriate beginning statement.



[See related topics](#)




Syntax checking buffer overflow

A statement you entered exceeds the limit of 32K bytes. Split the statement into multiple units, each with fewer than 32,768 bytes of data.

In general:

- Each line of script (including each line in a multiline statement) can contain approximately 1000 characters.
- Each statement can contain approximately 2400 language elements.

 [See related topics](#)



PUBLIC not allowed in this module

You defined a public variable, sub, function, property, or constant for an object other than (Globals). You can do one of the following to correct this problem:

- Move the definition to (Globals).
- Leave the definition where it is, but do one of the following:
 - If you are declaring a variable, replace the Public keyword with Private or Dim.
 - Otherwise, delete the Public keyword or change it to Private.



[See related topics](#)



Procedure definitions illegal in this scope

You tried to define a sub, function, or property within a sub, function, or property that defines a class method or property. This is not allowed.

Move the definition outside the scope of the class method or property.



[See related topics](#)



Illegal duplicate END statement

You added an End Sub, End Function, or End Property statement to a procedure that already contains the statement. Delete the duplicate statement.



[See related topics](#)



Illegal executable code outside procedure

You entered an executable statement outside a sub, function, or other procedure. Delete the statement or move it inside a procedure definition.

If you want the statement to be executed when scripts for an object are loaded, move the statement into the Initialize sub. If you want the statement to be executed when scripts for an object are unloaded, move the statement into the Terminate sub.



[See related topics](#)



Statement is illegal in this scope

You tried to enter a statement in a scope where it is not allowed. Check LotusScript Help for information about the statement you tried to enter and whether it can be used in module, procedure, user-defined data type, or class scope.



[See related topics](#)



Invalid type for procedure

You tried to change a product-defined procedure to another type of procedure. This is not allowed.

You cannot, for example, change a product-defined sub to a function by changing the Sub statement to a Function statement. Similarly, you cannot change a product-defined function or procedure to another procedure type.



[See related topics](#)



Illegal directive

Any of the following could have caused this error:

- You used an unrecognized directive. For example:

```
%EndRem    ' Illegal
%End Rem   ' Legal
```

- You nested a %Rem...%End Rem block inside another %Rem...%End Rem block.
- You used an %End Rem without a preceding %Rem.
- You entered a %If, %Else, %Elseif, or %EndIf directive in a script you created in the IDE.

If you want to use %If directives, you must enter them in a file that you call with the %Include directive.



[See related topics](#)



CASE ELSE must be the last CASE in a SELECT statement

You used a CASE clause after CASE ELSE in a Select Case statement. No other Case clause may follow a CASE ELSE clause.

Make CASE ELSE the last clause in the SELECT Case statement, or omit the keyword Else.



[See related topics](#)



DIM required on declarations in this scope

You declared a variable at module level without the Dim, Public, or Private keyword, or you declared a variable inside a procedure without the Dim or Static keyword. One of these is required.

Add the appropriate keyword to the declaration.




[See related topics](#)



Illegal character after continuation character

The line-continuation character underscore (`_`) is followed on the same line by a character that is not the comment character (`#`). The line-continuation character must be the last character on a line, except for an optional comment, beginning with the comment character.

Remove everything following the line-continuation character on the line, or insert a comment character after it to comment out the rest of the line.

 [See related topics](#)



Illegal use of escape character

You included an escape character at the end of a line. This is not allowed. For example:

```
aString$ = "This is a tilde: "  
anotherString$ = aString$~  
' This is illegal
```

Remove the escape character.



[See related topics](#)



Illegal use of escape character in identifier

You included an escape character in one of the following contexts in which that character is not allowed:

- In a declared name (a variable, constant, procedure, class, or user-defined data type)
- In the name of an implicitly declared variable
- In a label definition or reference
- In the name of the reference variable in a ForAll statement

For example:

```
Dim fo~x As Integer ' Illegal
```

Remove the escape character.



[See related topics](#)



Named product class instance not valid here

In one of the following statements, you used the name of a product object in a context in which it is not allowed:

- An assignment statement (Let or =) in either of the following forms:

Let *name* = ...

name = ...

- A Set statement in either of the following forms:

Set *name* = NEW...

Set *name* = ...

Set *name* = Bind...

- A Delete statement
- An Erase statement
- A ForAll statement
- A Get or Put statement
- An Input # or Line Input # statement
- An LSet or RSet statement
- A Mid or MidB statement
- A ReDim statement

Replace the name with an appropriate name, or remove the invalid statement.



[See related topics](#)



Illegal range specifier

You used a Def`type` range in one of the following illegal ways:

- No range was specified.
- The beginning of the range was not a single character between A and Z (ASCII uppercase or lowercase), inclusive.
- The end of the range was not a single character between A and Z (ASCII uppercase or lowercase), inclusive.

Correct the error.



[See related topics](#)



.. not valid outside of class scope

You used "dotdot" syntax outside of a procedure within a class. The "dotdot" syntax is only valid inside procedures within a class. You use "dotdot" notation when referring to a procedure in a base class when the derived class has a procedure of the same name, as in the following example:

```
CLASS BaseClass
  SUB MySub
    PRINT "In BaseClass's MySub"
  END SUB
END CLASS

CLASS DerivedClass AS BaseClass
  SUB MySub
    PRINT "In DerivedClass's MySub "
  END SUB

  SUB MyOtherSub
    CALL MySub
    CALL BaseClass..MySub
  END SUB
END CLASS
```

'Print "In DerivedClass's MySub "

'Print "In BaseClass's MySub "

Remove the "dotdot" syntax and use an object reference variable in its place.



[See related topics](#)



ME not valid outside of class scope

You used the keyword `ME` outside of a procedure within a class. Use the keyword `ME` only inside procedures within a class. You use `Me` within the definition of a class when referring to members of that class.

Remove the keyword `ME`. If you are referring to a class member, use an object reference variable instead of `Me`.



[See related topics](#)



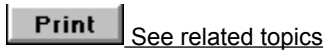
SET required on class instance assignment

You attempted to assign an object reference to a variable but omitted the SET keyword. (An object reference can be a reference to an instance of a user-defined class, a product object, an OLE automation object, or the constant NOTHING). The SET keyword is required in object reference assignments. For example:

```
Class MyClass
' ...
End Class
Dim MyObj As New MyClass
Dim varV As Variant
varV = MyObj           ' Illegal syntax
```

Insert the SET keyword in the assignment statement:

```
Class MyClass
' ...
End Class
Dim MyObj As New MyClass
Dim varV As Variant
Set varV = MyObj      ' Legal syntax
```






Out of memory

You must free enough memory to perform the operation that caused this error message. To free memory in your computer, do one of the following:

- If you have other programs in memory, end one or more of those programs.
- Reduce the amount or size of PUBLIC data.
- Activate extended memory.

 [See related topics](#)



Compiler stack overflow at: <token name>

The statement being compiled is too complex. It may contain a complex expression, or deeply nested block statements, such as a Do or For statement.

Reduce the nesting level, or break up the statement into multiple, less complex statements.



[See related topics](#)



Unexpected: <token>; Expected: <token>

The compiler encountered an unexpected language element.

If the unexpected language element is a number appearing inside square brackets, it represents the ASCII code of an unprintable character. For example, if you enter the Backspace character in a statement where a name is expected, the following error message appears when you compile the script:

```
Unexpected: [8]; Expected: Identifier
```

For more information, refer to the list of expected language elements following the unexpected language element in the error message.



[See related topics](#)



Illegal use of parentheses

You called a sub or function and enclosed its argument list in parentheses. You can only do this under the following circumstances:

- The sub or function is the target of a Call statement. For example:

```
Call MySub ()                ' Legal
Call MyOtherSub("ABC", 4)    ' Legal
Call MyFunction()           ' Legal
Call MyOtherFunction(123, "XXX") ' Legal
```

- The sub or function has a single parameter that the caller is passing by value. For example:

```
MySub("ABC")                ' Legal
MyFunction(anInt%)         ' Legal
```

- The target is a function that is included in a statement. For example:

```
X% = MyFunction(123, "XXX") ' Legal
```

The following are illegal:

```
MySub()                    ' Illegal
MyFunction()               ' Illegal
MyOtherSub("ABC", 4)      ' Illegal
MyOtherFunction(123, "XXX") ' Illegal
```

Remove the parentheses from around the argument list or call the sub or function with the Call statement.



[See related topics](#)



Compiler statement stack overflow at: *<token>*

The statement being compiled is too complex. It may contain deeply nested block statements, or single-line If statements.

Reduce the nesting level, or break up the statement into multiple, less complex statements.



[See related topics](#)

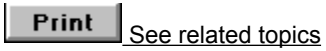


Illegal type suffix on keyword: <keyword>

You included an illegal data type suffix character in the name of a LotusScript built-in function. Certain LotusScript built-in functions can end in the \$ type suffix character; no other data type suffix character is valid on these functions. The names of other functions cannot end in a data type suffix character. For example:

```
Print Date()      ' Legal
Print Date$()    ' Legal
Print Date#      ' Illegal
Print CDat(Date) ' Legal
Print CDat$(Date) ' Illegal
```

Remove the suffix character.





Unterminated <keyword> block

You omitted the keyword that marks the end of one of the following block statements:

Class

Do

For

ForAll

Function

If...Then...Else...EndIf

Property Get

Property Set

Select Case

Sub

Type

While

Terminate the block with the appropriate statement.



[See related topics](#)



Unterminated string constant

You omitted the double quotation mark that signals the end of a quoted literal on a single line. Double quotation marks must be paired on the same line. For example:

```
Print "Hi,          ' Illegal because end quotation mark is missing
Martin."
```

```
Print "Hi, " _      ' Legal because string is properly quoted
"Martin."          ' Legal because string is properly quoted and
                   ' preceded by line-continuation character
' Output: Hi, Martin.
```

Terminate the string with double quotation marks on the same line where it starts.



[See related topics](#)



Unterminated square bracket reference

A square bracket reference was not terminated by a close square bracket (]) on the same line. Square brackets are used in some cases when referring to the names of product objects.

Terminate the square bracket reference with a close square bracket on the same line. Make sure that the product you are using supports square bracket notation for references.



[See related topics](#)



Illegal character after %INCLUDE directive

The %Include directive is followed on the same line by something other than the name of the file to include. The name of the file to include must be the only thing following %Include on a line, except for an optional comment, beginning with the comment character.

Remove everything following %Include and the name of the file, or insert a comment character after it to comment out the rest of the line.



[See related topics](#)



Illegal executable code in Options

In (Options), you included a statement that is not allowed in the section.

Only the following are allowed in (Options): Option, Deftype, Use, and UseLSX statements and Const statements associated with Use and UseLSX.

Remove the invalid statements from (Options).



[See related topics](#)



Illegal on declarations in this scope: <name>

The following conditions could have caused this error:

- You used the keyword Dim, Public, Private, or Static when defining a member variable in a Type statement. For example:

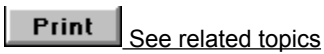
```
Type MyType
  Public X As Integer      'Illegal: Public keyword not allowed here.
End Type
```

Remove the Dim, Public, Private, or Static keyword.

- You used the Dim keyword when defining a member variable in a Class statement. For example:

```
Class MyClass
  Dim X As Integer        ' Illegal: Dim keyword not allowed here.
End Class
```

Remove the Dim keyword.





Name too long

The specified name is too long (it is truncated in the error message). The maximum length of a LotusScript name is 40 characters.

Shorten the name to 40 or fewer characters.



[See related topics](#)



Statement illegal in CLASS block: <keyword>

You used an illegal statement in a Class...End Class block.

The only legal statements in a Class. ...End Class block are:

- Declarations of variables without the keyword Dim or Static
A variable may be declared Public or Private, or with no leading keyword.
- Definitions without the keyword Static
- Definitions of the constructor and destructor subs (Sub New and Sub Delete) for the class
- The Rem statement
- The directives %Rem...%End Rem and %Include

By extension, when you use the %Include directive in a Class...End Class block, the file to which it refers must not contain any statements that are illegal inside a Class...End Class block.

Remove the illegal statement from the Class...End Class block.



[See related topics](#)



Statement illegal in TYPE block: <keyword>

You used an illegal statement in a Type...End Type block. The only legal statements in a Type...End Type block are declarations of variables without the leading keyword Dim, Public, Private, or Static; the Rem statement; and the directives %Rem...%End Rem and %Include. All other statements are illegal.

By extension, when you use the %Include directive in a Type...End Type block, the file to which it refers must not contain any statements that are illegal inside a Type...End Type block.

Remove the statement from the Type...End Type block.



[See related topics](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

[The Script Editor](#)

[The Script Debugger](#)

[The Breakpoints, Browser, Output, and Variables Panels](#)

Related topics

[Showing and Hiding Panes](#)

[Switching Between Panes](#)

[Switching Between Windows](#)

[Closing the IDE Window](#)

[Closing the Script Debugger](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Using the Keyboard in the Script Editor and Script Debugger](#)

[Using the Keyboard in the Breakpoints, Browser, Output, and Variables Panels](#)

[Using the Mouse](#)

[Working with Outline Lists](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Working in the LotusScript IDE Window](#)

[Using the Keyboard and Mouse](#)

Related topics

[Setting Script Properties](#)

[Working with \(Globals\) and Objects](#)

[Creating Scripts](#)

[Managing Text](#)

[Compiling and Testing Scripts](#)

Related topics

[Creating Scripts](#)

[Managing Text](#)

[Formatting Scripts](#)

Related topics

[Selecting an Object to Work With in the Script Editor](#)

[Entering \(Globals\)](#)

[Creating Scripts](#)

Related topics

[Working with \(Globals\) and Objects](#)

[Creating Scripts](#)

[Creating Procedures](#)

Related topics

[Working with \(Globals\) and Objects](#)

[Creating Scripts](#)

[Creating Procedures](#)

Related topics

[Creating \(Declarations\)](#)

[Entering \(Options\)](#)

[Creating Procedures](#)

[Working with Directives](#)

[Formatting Scripts](#)

[Using the Browser Panel](#)

Related topics

[Creating Scripts](#)

[Managing Text](#)

Related topics

[Creating Scripts](#)

[Managing Text](#)

Related topics

[Entering Type Statements](#)

[Entering Dim Statements](#)

[Entering Class Statements](#)

[Creating Scripts](#)

Related topics

[Creating \(Declarations\)](#)

[Entering \(Options\)](#)

[Using the Variables Panel](#)

Related topics

[Creating \(Declarations\)](#)

[Entering \(Options\)](#)

[Using the Variables Panel](#)

Related topics

[Creating \(Declarations\)](#)

[Entering \(Options\)](#)

[Using the Variables Panel](#)

Related topics

[Entering Option Statements](#)

[Entering Deftype Statements](#)

[Entering Use and UseLSX Statements](#)

[Creating Scripts](#)

Related topics

[Entering \(Options\)](#)

[Creating Scripts](#)

Related topics

[Entering \(Options\)](#)

[Creating Scripts](#)

Related topics

[Entering \(Options\)](#)

[Creating Scripts](#)

Related topics

[Renaming Functions and Subs](#)

[Working with Initialize and Terminate Subs](#)

[Creating Scripts](#)

[Working with \(Globals\) and Objects](#)

Related topics

[Creating Procedures](#)

[Creating Scripts](#)

Related topics

[Creating Procedures](#)

[Creating Scripts](#)

Related topics

[Creating Scripts](#)

[Compiling and Testing Scripts](#)

Related topics

[Selecting Text](#)

[Cutting, Copying, and Pasting Text](#)

[Entering and Deleting Text](#)

[Formatting Scripts](#)

Related topics

[Managing Text](#)

[Formatting Scripts](#)

Related topics

[Managing Text](#)

[Formatting Scripts](#)

Related topics

[Managing Text](#)

[Formatting Scripts](#)

Related topics

[Testing a Sub](#)

[Checking a Single Line of Script](#)

[Checking Scripts for an Object](#)

[Checking Scripts for a Document](#)

[Finding and Fixing Compile-Time Errors](#)

[Using the Output Panel](#)

Related topics

[Compiling and Testing Scripts](#)

[Creating Procedures](#)

[Creating Scripts](#)

[Working with \(Globals\) and Objects](#)

[Getting Help for Error Messages](#)

Related topics

[Compiling and Testing Scripts](#)

[Creating Procedures](#)

[Creating Scripts](#)

[Working with \(Globals\) and Objects](#)

[Getting Help for Error Messages](#)

Related topics

[Compiling and Testing Scripts](#)

[Creating Procedures](#)

[Creating Scripts](#)

[Working with \(Globals\) and Objects](#)

[Getting Help for Error Messages](#)

Related topics

[Compiling and Testing Scripts](#)

[Creating Procedures](#)

[Creating Scripts](#)

[Working with \(Globals\) and Objects](#)

[Getting Help for Error Messages](#)

Related topics

[Compiling and Testing Scripts](#)

[Getting Help for Error Messages](#)

Related topics

[Using Breakpoints in Debugging](#)

[Executing Scripts in the Debugger](#)

[Switching to Another Script in the Debugger](#)

[Fixing Errors Reported in the Script Debugger](#)

[Closing the Script Debugger](#)

Related topics

[Setting and Clearing Breakpoints](#)

[Enabling and Disabling Breakpoints](#)

[Using the Breakpoints Panel](#)

Related topics

[Using Breakpoints in Debugging](#)

[Using the Breakpoints Panel](#)

Related topics

[Using Breakpoints in Debugging](#)

[Using the Breakpoints Panel](#)

Related topics

[Stepping Through a Script](#)

[Continuing and Stopping Script Execution](#)

[Using the Output Panel](#)

[Using the Variables Panel](#)

Related topics

[Executing Scripts in the Debugger](#)

[Using Breakpoints in Debugging](#)

[Fixing Errors Reported in the Script Debugger](#)

[Switching to Another Script in the Debugger](#)

[Closing the Script Debugger](#)

Related topics

[Executing Scripts in the Debugger](#)

[Using Breakpoints in Debugging](#)

[Fixing Errors Reported in the Script Debugger](#)

[Switching to Another Script in the Debugger](#)

[Closing the Script Debugger](#)

Related topics

[Selecting a Script from the Calls Drop-down List](#)

[Selecting a Script from the Script List](#)

Related topics

[Switching to Another Script in the Debugger](#)

[Executing Scripts in the Debugger](#)

Related topics

[Switching to Another Script in the Debugger](#)

[Executing Scripts in the Debugger](#)

Related topics

[Getting Help for Error Messages](#)

[Executing Scripts in the Debugger](#)

[Switching to Another Script in the Debugger](#)

Related topics

[Executing Scripts in the Debugger](#)

[Using the Variables Panel](#)

Related topics

[Continuing and Stopping Script Execution](#)

[Closing the IDE Window](#)

Related topics

[Using the Breakpoints Panel](#)

[Using the Browser Panel](#)

[Using the Output Panel](#)

[Using the Variables Panel](#)

Related topics

[Viewing a List of Breakpoints](#)

[Managing Breakpoints](#)

[Using Breakpoints in Debugging](#)

Related topics

[Using the Breakpoints Panel](#)

[Using Breakpoints in Debugging](#)

Related topics

[Using the Breakpoints Panel](#)

[Using Breakpoints in Debugging](#)

Related topics

[Viewing LotusScript Keywords](#)

[Viewing Information About Application Components](#)

[Viewing Registered Type Libraries for OLE Classes](#)

[Pasting from the Browser into a Script](#)

Related topics

[Using the Browser Panel](#)

[Creating Scripts](#)

Related topics

[Using the Browser Panel](#)

[Creating Scripts](#)

Related topics

[Using the Browser Panel](#)

[Creating Scripts](#)

Related topics

[Using the Browser Panel](#)

[Creating Scripts](#)

Related topics

[Compiling and Testing Scripts](#)

[Executing Scripts in the Debugger](#)

Related topics

[Viewing Variables for the Current Script](#)

[Viewing Globals for the Current Script](#)

[Changing Variable Values](#)

Related topics

[Using the Variables Panel](#)

[Creating \(Declarations\)](#)

Related topics

[Using the Variables Panel](#)

[Creating \(Declarations\)](#)

Related topics

[Using the Variables Panel](#)

[Creating \(Declarations\)](#)

Related topics

[Getting Help on the IDE](#)

[Getting Context-Sensitive Help](#)

[Opening LotusScript Help](#)

[Getting Help for Error Messages](#)

Related topics

[Online Help in the IDE](#)

[Getting Help on the IDE](#)

[Opening LotusScript Help](#)

[Getting Help for Error Messages](#)

Related topics

[Online Help in the IDE](#)

[Getting Context-Sensitive Help](#)

Related topics

[Online Help in the IDE](#)

[Getting Context-Sensitive Help](#)

[Opening LotusScript Help](#)

[IDE Error Messages](#)

Related topics

[Online Help in the IDE](#)

[Getting Context-Sensitive Help](#)

[Opening LotusScript Help](#)

[Getting Help for Error Messages](#)

Related topics

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

Related topics

[IDE Error Messages](#)

[Finding and Fixing Compile-Time Errors](#)

[Fixing Errors Reported in the Script Debugger](#)

[Opening LotusScript Help](#)

