Appendix A Language and Script Limits

This appendix describes LotusScript language limits of several kinds: for example, the legal ranges in data representation, the limits on numerical specifications within statements, and the maximum number of different kinds of elements that can be defined in a script.

Appendix B Platform Differences

The LotusScript language and functionality on the OS/2 platform, the UNIX platform, and the Macintosh platform differ in various ways from the language and functionality described in the rest of this language reference. This appendix describes the differences.

Chapter 1 Script and Statement Construction Rules

This chapter describes the rules for writing the basic elements of a script in the LotusScript language.

Chapter 6 Operators

The LotusScript operators are described in detail in the sections of this chapter. The operators are grouped according to the kinds of operations they perform.

Chapter 7 Statements, Built-In Functions, Subs, Data Types, and Directives

Part 1 Scripting Basics

Part 2 Language Elements

~~Part 3 Appendices~~

Illegal pass by value

You tried to pass an argument by value that may not be passed by value, either by using parentheses around the argument, or by using the ByVal keyword on an argument in a call to an external C function.

You may have inadvertently put parentheses around an argument in a sub or function call. Use parentheses on arguments in sub and function calls only if you are using the Call keyword.

The following arguments cannot be passed by value:

- *Arrays*
- *Lists*
- *Variables of a user-defined data type*
- *Object reference variables*

In addition, only arguments of type String or Variant can be passed by value to the LotusScript Len function. Arguments of other data types cannot be passed by value.

Remove the parentheses or the ByVal keyword.

Cannot subclass: <class name>

You specified a product class as the base class of a derived class. A product class may not be used as the base class of a derived class.

Remove the As BaseClassName clause in the class declaration, or specify a LotusScript class as the base class.

Arguments not legal in declaration of: <sub name>

The following conditions could have caused this error:

- *You specified parameters in a Sub Initialize or Sub Terminate definition. Because the Initialize and Terminate subs are executed automatically on module load and unload, they cannot be passed arguments.*

  Redefine the sub without parameters.

- *You specified parameters in a Sub Delete definition. Because the Delete sub is executed automatically when an object reference is deleted, it cannot be passed arguments.*

  Redefine the sub without parameters.

Illegal OPTION BASE after array declaration

The Option Base statement appeared after an array declaration or after a ReDim statement.

Move the Option Base statement so that it precedes all array declarations and ReDim statements.

Array size exceeds maximum: <array name>

You declared an array whose total size is greater than the maximum allowable size. The maximum allowable array size is 65,536 bytes (64K).

*Reduce the array size to 65,536 bytes or less. The size is calculated as (number of elements) * (size of each element in bytes).*

Illegal BYVAL on arguments to: <subprogram name>

You used the ByVal keyword in a call to a procedure that is not an external C function. The ByVal keyword may only be used when specifying the parameters in the declaration or definition of a sub or function with a Declare, Sub, or Function statement, in specifying the parameters of an external C function with a Declare, and in calling an external C function with a Call statement.

Remove the ByVal keyword, revise the definition of the sub or function, or use parentheses around the argument in the call statement to pass the argument by value.

Illegal Directive

Any of the following could have caused this error:

- *You used an unrecognized directive. For example:*

%Else If ' Illegal

%ElseIf ' Legal

- *You nested a %Rem...%End Rem block inside another %Rem...%End Rem block.*
- *You used an %End Rem without a preceding %Rem.*
- *You used a %Else, %ElseIf, or %End If directive outside a %If...%End If block.*
- *You nested a %If...%End If block inside another %If...%End If block.*

Cannot open included file: <file name>

One of the following conditions could have caused this error:

- *The path or the file name you specified is incorrect.*

Fix the path or the file name, or move the file to the directory specified in the path.

- *The file is not in your working directory or in the directory you specified in the path.*

Move the file to your working directory or to the directory you specified in the path.

- *The file could not be opened.*

Correct the situation that is preventing you from opening the file.

Illegal string length constant for: <name>

You specified a length for a fixed-length string as one of the following:

• *An item that is not a literal or a constant (created with the Const statement)*

Change the length specifier to a literal or a constant.

• *A literal that is not an Integer or Long value, or a constant that does not have an Integer or Long value*

Use an Integer or Long literal, or a constant with an Integer or Long value.

• *A value not in the range 1 to 32767*

Change the length specifier to a number within this range.

Illegal use of NEW on array or list declaration: <name>

You used the keyword New in declaring an array or list. This not allowed. In an array or a list whose type is a class, the elements must be constructed individually.

Remove the New keyword from the declaration of the array or list specified in the error message.

Wrong number of arguments for: <name>

The following conditions could have caused this error:

- *You specified the wrong number of arguments when you called a sub or function.*

Change the number of arguments in the sub or function call to the correct number.

- *You specified the wrong number of arguments when you called a built-in function.*

For information about the function signature for a specific built-in function, consult the Help topic for that function.

Wrong number of arguments to constructor for class: <class name>

You supplied the wrong number of arguments for a class constructor in one of the following statements:

- *A declaration of the form:*

```
Dim X As  New ClassName
```

For example:

```
Class MyClass
   Sub New(A As Integer, B As String)
      '...
   End Sub
End Class
Dim ObjRef As New MyClass(4, "Alex", "Jones")   ' Illegal because
                                                ' MyClass's Sub New takes
                                                ' only two arguments
Dim ObjRef As New MyClass(4, "Alex Jones")      ' Legal
```

- *A Set statement of the form:*

```
Set X = New ClassName
```

- *A declaration of a derived class when the arguments that the derived class's constructor requires are different from the ones that the base class's constructor requires. In this case, constructor arguments for the base class must be specified after the BaseClassName clause in the Sub New declaration, as in the following example:*

```
Class BaseClass
    Sub New(X As Integer)
        '...
    End Sub
End Class
Class DerivedClass As BaseClass
    Sub New(Y As String, X As Integer ), BaseClass(X%, Y)   ' Illegal
    Sub New(Y As String, X As Integer ), BaseClass(X)       ' Legal
        '...
    End Sub
End Class
```

Supply the correct number of arguments to the constructor.

Wrong number of arguments for event handler: <sub name>

In an On Event statement, the number of arguments you included in the Call clause does not match the number required by the product class event.

Check the product documentation for a description of the arguments defined for the event.

Class not specified on BIND into: <name>

You tried to assign a reference to a product object to a variable of type Variant with the Set...Bind statement and you omitted the class name of the object. For example, assuming a product class named ProdADT:

```
Dim P As New ProdADT("MyProdADT")
Dim varV As Variant
Set varV = Bind("MyProdADT")                ' Illegal because product class name
                                            ' is missing
Set varV = Bind ProdADT("MyProdADT")        ' Legal syntax
```

Insert the name of the product class after the Bind keyword.

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.

CLASS or TYPE declaration may not be inside a control block

You tried to include a Class or Type statement inside one or another of the following block statements: Do, For, ForAll, If...Then...Else...EndIf, Select Case, While. This is not allowed. For example:

```
If 1 = 1 Then
    Class MyClass    ' Illegal
        '...
    End Class
End If
```

Move the Class or Type statement to outside the block.

Cannot forward declare CLASS or TYPE

You tried to use the Declare statement to declare a user-defined data type or class before defining it. This is not allowed. Declare may only be used to forward-declare functions, subs, and properties.

Collection item is not an instance

You referred to an item in an indexed collection as though that item were an object, but it isn't. For example, if iColl is a collection of integers, the following statement would be illegal:

iColl(3).value = 4

Conflicting option

You specified conflicting options in an Option Compare statement or statements. You cannot specify any other options if you specify Binary. You cannot specify both Case and NoCase. You cannot specify both Pitch and NoPitch.

Error number must be INTEGER constant: <name>

You used a name as an error number in an On Error statement, but it is not a constant of type Integer. A name used as an error number in an On Error statement must be a constant of type Integer.

Define the name as an integer constant (with the Const statement), or use an integer numeric value. If the name is the name of a LotusScript error constant, use %Include to include the file LSERR.LSS in your module.

Size of data cannot exceed 64K in this scope

The data in the enclosing scope (module or class) exceeds the limit of 64K bytes.

Split the enclosing scope into multiple units, each with less than 65536 bytes of data.

Size of data cannot exceed 32K in this scope

The data in the enclosing scope (sub, function, or property) exceeds the limit of 32K bytes.

Split the enclosing scope into multiple units, each with less than 32768 bytes of data.

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.

Duplicate forward declaration: <name>

You have used a Declare statement twice to declare the same function, sub, or property in this scope.

Remove one or the other of these Declare statements.

Duplicate label: <label name>

You defined the label specified in the error message more than once within the same scope.

Define the label named in the error message only once. Define other labels to replace the other instances of this label.

Duplicate option

You used the Option Base, Option Declare, or Option Public statements more than once in a module. These statements can only appear once each per module.

Remove any repeated instances of the Option Base, Option Declare, or Option Public statements within the module. To override the lower bound setting specified by the Option Base statement, use explicit lower bounds in a Dim or ReDim statement.

Public symbol is declared in another module: <name>

A name declared as Public has already been declared as Public in another loaded module. A name can be declared as Public in only one loaded module at a time. Other loaded modules can only reference that name.

Remove Public from the declaration, or change the Public name so that it does not conflict with the name in the already loaded module.

Duplicate range specifier

You included a letter in a Deftype range that is already included in another Deftype range in the same module. Once a letter has been included in a Deftype range, it may not be included in another Deftype range in the same module. For example:

DefInt A-D

DefInt D-G   ' Illegal: D already belongs to a range.

If Deftype a-z has been specified in a module, no other Deftype range may be specified in that module.

Redefine your Deftype ranges so that no letter is included in more than one range.

Wrong data type for argument <argument name> in event handler <event handler name>

You specified a procedure as the handler in an On Event statement. The declared data type of a parameter in the definition of that procedure does not match the data type of the corresponding parameter specified when the event was registered with LotusScript. Refer to the documentation of the product in which you are running LotusScript for information about the arguments that the event handler requires. Change the declared data type of the parameter in the subprogram definition to match the registered data type of the corresponding parameter.

Event handler must be a FUNCTION

The event handler for an object is a function and the user-defined procedure is a sub.

Wrong return type in event handler <handler_name>

The return type of the event does not match the return type of the function.

Event handler must be a SUB

The event handler for an object is a sub and the user-defined procedure is a function.

Expected expression before end of argument list for: <function name>

You used a comma before the last optional argument in a call to a built-in function, but you did not supply the argument. For example:

```
myVal% = StrCompare("abc", "abc",)          ' Illegal
```

Remove the comma, or specify the last optional argument:

```
myVal% = StrCompare("abc", "abc")           ' Legal
myVal% = StrCompare("abc", "abc", 1)        ' Legal
```

FOR count variable must be a scalar variable: <name>

The count variable of a For statement must be a scalar variable. The variable cannot be an array or list variable, or an element of an array or list. Its type cannot be a user-defined type or a class; and it cannot be a member of a user-defined type or of a class. It cannot be a property, a function, or a constant.

Change the For count variable to a scalar variable.

Argument does not match forward declaration: <argument name>

You declared a function or sub with a Declare statement and then defined it with a Function or Substatement. Either of the following conditions could have caused the error:

- *The data type of the indicated parameter in the procedure definition is different from the corresponding parameter in the procedure declaration. For example:*

```
Declare Sub MySub(X As Integer)
'...
Sub MySub(X As Double)   ' Illegal because X was previously declared
'...                     ' to be of type Integer
End Sub
```

Change the data type of the indicated paramater in the declaration or the definition of the procedure so that they match.

- *The data type of the indicated parameter matches the data type of the corresponding parameter in the procedure declaration, but the parameters represent different kinds of data structure. For example:*

```
Declare Function MyFunction(X() As Integer) As Integer
'...
Function MyFunction(X As Integer) As Integer   ' Illegal because X is a
'...                                           ' scalar variable but
                                               ' X() is an array
End Function
```

Change the parameter specification in the declaration or the definition of the procedure so that the two match.

Storage class or visibility does not match forward declaration: <subprogram name>

You declared a function, sub, or property with a Declare statement and then defined the procedure with a Function, Sub, Property Set, or Property Get statement. The definition differs from the declaration in one or another of the following respects:

- *The declaration contains the keyword Static but the definition doesn't, or vice versa. (The keyword Static specifies that the storage class of the procedure's variables will be static by default.)*
- *The procedure is declared as Public but defined as Private, or vice versa.*

Change the declaration or the corresponding definition of the procedure so that they match.

Number of arguments does not match forward declaration: <subprogram name>

You declared a function or sub with a Declare statement, and then either of the following happened:

- *You defined the function or sub with a Function or Sub statement specifying a different number of parameters than you specified in the Declare statement. For example:*

```
Declare Function MyFunction(X As Integer, Y As Double) As Integer
'...
Function MyFunction(X As Integer) As Integer   ' Illegal because Declare
'...                                            ' specified two parameters
End Function
```

Make the parameters in the declaration and definition match each other.

- *The procedure that you forward declared is a parameterized constructor sub (Sub New) inside a Class statement and you have not defined a Sub New for that class.*

Either remove the Declare statement or define a corresponding Sub New.

Return type does not match forward declaration: <function name>

You have declared a function or property with a Declare statement and then defined it with a Function, Property Set, or Property Get statement. The data type that you specified as the procedure's return value in the Declare statement is different from the data type you specified as the return value in the definition statement. For example:

```
Declare Property Set MyProperty As Integer
'...
Property Set MyProperty As Double   ' Illegal because MyProperty's return
'...                                 ' value was already declared as Integer
End Property
```

Change the data type of the return value in the declaration or in the corresponding definition so that they match.

Too many labels specified in ON...GOTO statement

More than 255 labels were specified in an On...GoTo statement.

Reduce the number of labels to fewer than 256.

Not a product class instance: <name>

Where a reference to a product object was expected in an On Event statement, you used the name of something else. In an On Event statement, the name specified in the From clause must be a product object reference variable, the name of a function or property that returns a product object reference, or a Variant that holds a product object reference.

If a product object was intended, make the reference be to the intended product object. Otherwise, remove the statement.

Illegal data type for argument: <argument name>

You used a fixed-length string as a parameter in the declaration of a sub or function.

Fixed-length strings are not legal as parameters in subs or functions.

Change the parameter's data type to String or Variant.

Illegal array bound for: <array name>

The following conditions could have caused this error:

• *One of the array bounds specified does not evaluate to an integer constant. The range of an integer constant is -32768 to 32767 (inclusive).*

Specify the bound so that it evaluates to between -32768 and 32767.

• *In one of the specified array dimensions, the lower bound is greater than the upper bound. The lower bound must be less than or equal to the upper bound.*

Respecify the lower bound or the upper bound.

Illegal reference to array or list: <array or list name>

You used the name of an array or list in an illegal context. Illegal contexts include the following, where X is the name of an array or list:

- *As the target of an assignment or Set statement, as in X = Y, Set X = Y, Set X = New Y, Set X = Bind Y*
- *As the target of a Delete statement, as in Delete X*
- *As though it were an object reference variable or a variable of a user-defined data type and you were referring to one of its members, as in X.Y*

Remove the illegal use of the array or list.

Illegal value for OPTION BASE

The following conditions could have caused this error:

- *The element following the Option Base statement is not an Integer constant.*
- *The value of the constant is not 0 or 1.*

Change the element following the Option Base statement to an Integer constant whose value is 0 or 1.

Not a product class: <name>

You used a user-defined class name in the following statement:

Set X = Bind ClassName (ObjectName)

The class name used in a Set...Bind... statement must be a product class name, not a user-defined class name.

Change the class name following the Bind keyword to a product class name, or remove the statement.

Illegal call to: <sub name>

You tried to call a class's Sub New or Sub Delete. A class's Sub New is called automatically when an object (class instance) is constructed. It may not be called directly. A class's Sub Delete is called automatically when an object is deleted. It may not be called directly.

Illegal external argument: <argument name>

You declared a C function and specified the data type of one of its parameters as a fixed-length string or as a list. You cannot specify a C function parameter as a fixed-length string or a list.

For a fixed-length string, declare the parameter as type String, Variant, or Any.

For a list, declare the parameter as type Any.

Declaration of external subprogram is not legal inside a class

You tried to use a Declare statement inside a class definition to declare an external C function. This is not allowed.

Move the declaration of the external function to the module level.

Statement is 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 and forward declarations of subprograms, 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.

Illegal constant expression for: <CONST name>

One of the following occurred in a Const statement:

• You used a value of a data type that does not match the data type suffix character of the constant. If the constant and the value are both numeric, the value may be too large for the data type of the constant.

Change the constant's data type suffix character, or change the value so that it is legal for the constant's data type.

• You tried to define a constant with a nonconstant value. The value assigned by a Const statement must be a constant value; that is, one of the following:

•A literal

•A constant previously defined by a Const statement

•A built-in function whose arguments are constant expressions

•An expression whose operands are either literals; constants previously defined by Const statements; or one of a number of built-in functions whose arguments are constant expressions

Change the assigned value to a constant value.

Illegal constructor clause on: <sub name>

You specified a constructor clause on a sub that is not a class constructor sub (Sub New). For example:

```
Class BaseClass
    Sub New (X As long)
    End Sub
End class
Class DerivedClass As BaseClass
    Sub Old (X As Long, Y As Long), BaseClass(X) ' Illegal: Old is not a
                                                 ' constructor sub.
    End Sub
End Class
```

A class constructor sub must be a part of the definition of a class, and must be named New.

If the sub is not intended to be a class constructor, remove the constructor clause (that is, the comma, the name of the class, and the argument list). Otherwise, rename the sub to New.

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.

Member is not a subprogram: <member name>

You used "dotdot" notation to refer to a member variable of a base class. This notation is legal only for referring to a member function, sub, or property of a base class. It is not legal for referring to member variables of a base class.

Refer to the variable by its name only.

Class is not a parent of this class: <class name>

Either of the following conditions could have caused this error:

- *A class specified in a Sub New declaration is not the class from which this one is derived.*
- *A class specified using "dotdot" notation is not the class from which this one is derived.*

For example:

```
 Class BaseClassOne
    Sub New   (X As Integer )
    End Sub
End Class
Class   BaseClassTwo
    Sub   PrintIt
          '...
    End Sub
```

```
End Class

Class    DerivedClass   As   BaseClassOne

    Sub New (Y As Integer ), BaseClassTwo (x%)

' Illegal because BaseClassTwo is not the base

' class from which DerivedClass is derived.

' The appropriate base class is BaseClassOne.

    End Sub

    Sub   PrintIt

          '...

    End Sub

    Sub   CallPrintIt

          Call   BaseClassTwo..PrintIt

' Illegal because BaseClassTwo is not the base

' class from which DerivedClass is derived.

' The appropriate base class is BaseClassOne.

    End Sub

End Class
```

Correct the reference to the base class.

Illegal on declarations in this scope: <keyword>

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 is 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 is not allowed here.
End Class
```

Remove the Dim keyword.

Illegal DEFtype statement after declaration

A Deftype statement is located in the wrong part of the module. Deftype statements must appear before all declarations (both explicit and implicit) in the module.

Move the Deftype statement so that it precedes the first declaration in the module.

DELETE not valid on: <name>

You used the Delete statement on one of the following:

- *A variable that is not an object reference variable*
- *A variable of type Variant that does not contain an object reference*
- *The return value of a function*
- *A property*

Assign the object to an object reference variable and apply Delete to the variable instead.

Illegal use of ERASE

You used the Erase statement incorrectly. You can only erase an array, a list, a list element, or a Variant that holds an array, a list, or a list element.

Remove the invalid Erase statement or change the reference in the statement to an array, list, list element, or Variant.

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.

Illegal use of escape character in identifier: <name>

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.

EVALUATE argument must be a string constant

The name you specified in an Evaluate function or statement is not a quoted literal or a string constant, though one was required.

Supply a quoted literal or string constant.

Not an event name: <name>

Where an event name was expected in an On Event statement, you specified a name that is not an event name. Event names are registered with product classes.

Change the name to a product event name, or remove the statement.

Illegal EXIT <EXIT type>

You used an Exit statement of a particular type outside a block statement of that type. The six types of Exit statement, and the block statements where each can appear, are as follows:

- *Exit Do can appear only within a Do statement*
- *Exit For can appear only within a For statement*
- *Exit ForAll can appear only within a ForAll statement*
- *Exit Function can appear only within a Function statement*
- *Exit Sub can appear only within a Sub statement*
- *Exit Property can appear only within a Property Get statement or a Property Set statement*

If the Exit statement is unintended, remove it.

If the Exit statement has the right type but is misplaced, relocate it to within the intended block of that type.

If the Exit statement is in the intended place within a block but has the wrong type, change its type to the type of that block.

Illegal function return type for: <function name>

You either used a Declare or Function statement to declare or define a function and specified its return type as a fixed-length string or a user-defined data type, or else you used a Declare statement to declare an external C function and specified its return type as Variant, Currency, fixed-length String, or a user-defined data type.

Specify a data type other than the ones listed above for the function's return value.

Name was forward declared as something else: <name>

You named a function, sub, or property in a Declare statement and then used that name in the definition of a different kind of procedure. For example:

```
Declare Sub MyProcedure
Property Set MyProcedure    ' Illegal because you previously declared
                            ' MyProcedure as a sub
    '...
End Property
```

Change the declaration or its corresponding definition so that both are either functions, subs, or properties.

Event handler must be a LotusScript SUB or FUNCTION: <handler name>

The handler name specified in an On Event statement is not the name of a LotusScript sub or function. An event handler may not be an external (C) function or a product object method.

Illegal character after %INCLUDE directive

A %Include directive is followed on the same line by a character that is not the comment character ('). The file name of the file to be included must be the last item on the line, except for an optional comment, beginning with the comment character.

Remove everything following the file name on the line, or insert a comment character following the file name.

Illegal second parenthesized expression

You tried to refer to an element in a nested array, list, or collection. For example:

```
Dim anArray(1 To 3) As Variant
Dim anotherArray(1 To 3) As Integer
anotherArray(1) = 1
anotherArray(2) = 2
anotherArray(3) = 3
anArray(1) = anotherArray
Print anArray(1)(1)                    ' Illegal
```

To refer to an element in a nested array, list, or collection, assign the inner array, list, or collection to a variable of type Variant:

```
Dim varV As Variant
varV = anArray(1)
Print varV(1)                          ' Legal
```

Not an instance name: <name>

A name is followed by a dot, but the name is not an object reference variable, a Variant variable containing a reference to an object, or a variable of a user-defined data type.

Use "dot" notation only with variables of one of these three kinds.

Replace the name with the name of a valid variable.

Illegal reference to FORALL alias variable: <name>

You referred to a name that was previously used as the reference variable in a ForAll reference variable. You referred to that variable outside of the ForAll loop. ForAll reference variables may not be referred to outside of a ForAll loop.

Remove the reference to the variable.

Illegal use of array or list element as FORALL target

You used an array or list element as the target of a ForAll statement.

To iterate over an array or list, use the array or list name only. For example:

```
Dim Y List As String
' ...
ForAll X In Y(1)          ' Illegal. Target is an array or list element.
ForAll X In Y             ' OK. Target is an entire array or list.
```

Not an array, list, collection or variant: <name>

The target of a ForAll statement is not an array, list, or collection or a Variant that holds a reference to an array, list, or collection.

Change the target to one of these, or remove the ForAll statement.

LIB name must be a string constant

The name that you specified in the Lib clause of a Declare statement is not a quoted literal or a string constant though that is what is required. Change the name to a quoted literal or string constant.

~~Reference must contain exactly one subscript: <name>~~

~~A reference to a list or collection contains either no subscript or more than one subscript. A list or collection reference must contain exactly one subscript.~~

~~Specify exactly one subscript in the reference.~~

Variable required: <name>

In one of the following statements, you used a name that is not the name of a variable, a property, or a ForAll reference variable:

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

  Let name = ...

  name = ...

- *A Set statement in any of the following forms:*

  Set name = New...

  Set name = ObjectReferenceVariable

  Set name = Bind (ProductObjectName)

- *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*

In each of these statements, the name must be the name of a variable, a property, or a ForAll reference variable.

Replace the name with a valid name, or remove the invalid statement.

Error in EVALUATE macro

The macro named in an Evaluate function or statement is not a valid macro in the product that you are using.

Correct the macro or remove the Evaluate function or statement.

Not a member: <name>

You referred to a nonexistent member of a class or user-defined data type. For example:

```
Type myType
    A As Integer
End Type
Dim X As myType
X.nonVar% = 10     ' Illegal because nonVar% is not defined in myType
```

Define the member within the class or data type definition, or remove the reference.

Label is illegal outside of a subprogram

You defined a label at the module level. Labels may not be defined at the module level. Executable statements at the module level are executed as the module is compiled, and then discarded. Therefore, control cannot be transferred to a labeled statement at the module level.

Remove the label, or the entire labeled statement. Revise the script to remove any attempted transfer of control to the labeled statement.

Statement is illegal outside of a subprogram

You used a statement that is not legal at the module level. These statements include:

- *End statement*
- *Execute statement*
- *GoSub statement*
- *GoTo statement*
- *If...GoTo statement*
- *On Error statement*
- *On...GoTo statement*
- *On...GoSub statement*
- *Resume statement*
- *Return statement*
- *SendKeys statement*
- *Yield statement*

Revise the script to remove any of these statements at the module level.

Illegal use of NEW or DELETE

You used the name New or Delete to name a function, property, or variable within a class definition. Within a class, the names New and Delete are reserved for subs; they may not be functions, properties, or variables.

Rename the function, property, or variable. To specify a sub to be executed on the construction or deletion of an object, include a Sub New or Sub Delete in the class definition.

Illegal name for class or type: <name>

You used the word Object as the name of a user-defined class or data type. Object is a LotusScript reserved word.

Change the name of the user-defined class or data type.

Illegal ON ERROR statement

In an On_Error...GoTo statement, the element that follows the GoTo keyword is neither a label nor an integer constant equal to zero, which is what is required.

Change the element following the GoTo keyword to a label or to an integer constant equal to zero.

Illegal OPTION DECLARE after implicit declaration

You used an implicit declaration before the Option Declare statement.

Move the Option Declare statement so that it appears before all variable declarations.

Illegal OPTION PUBLIC after declaration

The Option Public statement was used after an explicit declaration of a variable, constant, procedure, user-defined data type, or class.

Move the Option Public statement so that it precedes all explicit declarations.

Illegal product constant: <name>

You specified a product constant name that was not recognized by the product.

Check the documentation for the product. Use a correct product constant name (check the spelling), or remove the reference to the product constant.

Illegal PRIVATE declaration of: <name>

In defining a class, you declared the Sub New or Sub Delete as Private. New and Delete subs may not be declared as Private.

Remove Private from the declaration of the New or Delete sub.

Not a PUBLIC member: <name>

You referred to a Private member of a class outside of the class's scope. Only Public class members can be referred to outside of their defining class's scope. (By default, member variables are Private, whereas member functions, subs, and properties are Public unless explicitly declared as Private.)

Remove the Private keyword (if any) from the declaration of the class member, and substitute the keyword Public in its place.

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.

Illegal property type for: <property name>

In declaring or defining a property with a Declare statement or a Property Get or Property Set statement, you specified its data type as either a fixed-length string or a user-defined data type. Properties cannot be fixed-length strings or user-defined data types.

Declare the property as a different data type.

Illegal PUBLIC declaration of: <name>

You declared the Initialize or Terminate sub as Public. The Initialize and Terminate subs may not be declared as Public.

Remove Public from the declaration of the sub.

Illegal range specifier

You used a Deftype 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 and recompile.

Illegal REDIM on: <name>

You used the ReDim statement on a name that is not the name of a dynamic array. For example:

```
Dim anArray(1 To 2) As Integer
ReDim anArray(1 To 3)              ' Illegal because anArray was previously
                                   ' declared as a fixed array.
```

Either replace the name in the ReDim statement with the name of a dynamic array, or remove the statement.

Illegal parenthesized reference: <name>

You referred to a name followed by parentheses, but the reference is not to an array, list, or a collection, or a Variant containing a reference to one of these, or to a function.

- *If the reference is intended to be to one of the above, check the spelling and correct it if necessary.*
- *If the reference is not intended to be to one of the above, remove the parentheses from the reference.*

Illegal RESUME statement

In a Resume statement, you used a numeric to specify the statement at which execution is to continue. If you specify a numeric element in a Resume statement, it must evaluate to zero.

Remove the element or change it to an integer constant or literal with a value of zero. Resume and Resume 0 have the same meaning.

Illegal reference to: <name>

You used a name as though it contained or referred to a value, but it doesn't. For example:

```
Sub MySub
    Print "Hello"
End Sub
stringVar$ = MySub        ' Illegal because MySub does not return a value
```

Remove this use of the name, or replace it with a name that has a value (for example, a function name instead of a sub name).

Illegal scope for PUBLIC or PRIVATE on: <name>

You used the Public or Private keyword in a declaration within a sub, function, or property. The Public and Private keywords are not legal in declarations in subs, functions, or properties. Public and Private only have meaning in declarations in module scope or within the definition of a user-defined class.

Remove the Public or Private keyword from the declaration.

SET may only be used on class instance assignments

You used a Set statement to try to assign something other than a object reference to a variable. For example:

```
Class MyClass
    Public X As Integer
End Class
Dim MyObjRef As New MyClass
Set MyObjRef.X = 5              ' Illegal
Let MyObjRef.X = 5              ' Legal
MyObjRef.X = 5                  ' Legal
```

Remove the Set keyword or replace it with the Let keyword.

Illegal single-line IF

A physical end-of-line (with no line-continuation character) appeared before the end of the Then or Else clause in an If...Then...Else statement. For example:

```
If  X = Y Then Do : X = X + 1
Loop                    ' Illegal. Loop must appear on same line as Do.
```

A single-line If...Then...Else s tatement must be completely contained on one line, including any continuation lines designated by line-continuation characters.

Do one of the following:

- *Write the Then clause and the Else clause on the same line as the If.*
- *Use a line-continuation character.*
- *Use an If...Then...Else...End If block statement in place of the single-line If...Then...Else statement.*

Illegal STATIC on: <name>

You used the Static keyword in the declaration of one of the following:

- *An external C function*
- *A class member (a variable, property, function, or sub)*
- *A variable declared inside a class method or property*
- *A variable declared at module level*

Remove the keyword Static from the declaration.

Illegal use of UNICODE or LMBCS keyword

In a Declare statement, you included the Unicode or LMBCS keyword with an object reference argument. This is not allowed. For example:

```
Class MyClass
    '...
End Class
Dim X As New MyClass
Declare Function MyFunc Lib "C:\USER.DLL" (X As LMBCS MyClass) As Long
' Illegal
```

Instead of passing an object reference, pass a variable of a user-defined data type.

Not a sub or function name: <name>

In a statement where the name of a function or sub is expected, you specified a name that is not recognized as a sub or function name. The statement is one of the following:

- *A Call statement*
- *A call without the Call keyword (for example, a statement consisting of a name)*

If the sub or function has not been defined before being called from within a procedure, use the Declare statement to forward declare it. You must define a sub or function before calling it at module level.

Illegal type suffix on name: <name>

You appended a data type suffix character to one of the following:

- *The name of a user-defined data type*
- *The name of a class*
- *A sub name*
- *A label*
- *A product event name*

Suffix characters are not valid on these names.

Remove suffix characters from any names on which they are invalid.

Illegal TO in reference to: <name>

One of the following conditions could have caused this error:

• *You specified a range (bound1 To bound2) as a subscript in an array element reference.*

Remove the range; specify a single subscript.

• *You specified a range (bound1 To bound2) as an argument in a call to a procedure.*

Replace the range by a valid argument.

Use ranges in array declarations or ReDim statements only.

Declaration not valid in TYPE scope: <name>

You declared one of the following as a member of a user-defined data type:

- *An object reference variable*
- *A list variable*
- *A dynamic array variable*

Object reference variables, list variables, and dynamic array variables are not valid members of a user-defined data type.

Remove the invalid member declaration.

Statement is 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.

UNICODE and LMBCS strings must be declared BYVAL

In a Declare statement, you included the Unicode or LMBCS keyword with a string argument but did not include the ByVal keyword, which is required for passing string arguments. For example:

```
Declare Function MyFunc Lib "c:\USER.DLL" (X As LMBCS String) As Long
' Illegal
```

Include the ByVal keyword in the Declare statement:

```
Declare Function MyFunc Lib "c:\USER.DLL" (ByVal X As LMBCS String) As Long
```

Illegal USE or UseLSX statement after declaration

You used a Use or UseLSX statement after an implicit declaration.

Move the Use or UseLSX statement so that it precedes all implicit declarations.

USE or USELSX name must be a string constant

The name that you specified in a Use or UseLSX statement is not a quoted literal or a string constant though that is what is required. For example:

```
Use LSCONST.LSS              ' Illegal

Use "LSCONST.LSS"            ' Legal

Const myFile$ = "LSCONST.LSS"

Use myFile$                  ' Legal
```

Change the name to a quoted literal or string constant.

Empty parentheses not legal on: <name>

You included empty parentheses in referring to a variable of type Variant or an undefined function or sub (which LotusScript interprets as a reference to an implicitly declared variable of type Variant). For example:

```
Dim anArray(1 To 3) As Integer
Dim varV As Variant
varV() = anArray()          ' Illegal
varV = anArray()            ' Legal
varV = anArray              ' Legal
Dim X As Integer
X% = varV()                 ' Illegal
X% = varV                   ' Legal
```

Remove the parentheses from the Variant variable.

Error number must be INTEGER

You used a numeric constant as an error number in an On Error statement, but it is not an integer. The value of a constant used as an error number in an On Error statement must be an integer.

Change the numeric constant to an integer.

.. 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 M ySub"
    End Sub
End Class

Class DerivedClass As BaseClass
    Sub   MySub
        Print " In DerivedClass's MySub "
    End Sub
—
    Sub   MyOtherSub
        Call MySub                          ' Print "In DerivedClass's MySub "
        Call BaseClass..MySub               ' Print "In BaseClass's MySub "
    End Sub
End Class
```

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

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.

Illegal BYVAL

One of the following conditions could have caused this error:

• *You specified the ByVal keyword on a subscript in referring to an array element.*

Remove the ByVal keyword.

• *You specified the ByVal keyword in an array bounds expression in a ReDim statement.*

Remove the ByVal keyword.

Illegal use of property: <property name>

You tried to use the named property as one of the following:

- *The target in a Get or Put statement*
- *The target in an Input # or Line Input # statement*
- *The target in an LSet, RSet, or Mid statement*

You must use a variable, not a property, for any of these purposes.

This error also occurs when the property appears with a subscript as the target of an assignment statement. For example:

```
Dim privateArray(1 To 2) As String
Property Set MyProperty As Variant
    privateArray(1) = MyProperty(1)
    privateArray(2) = MyProperty(2)
End Property
Property Get MyProperty As Variant
    MyProperty = privateArray
End property
MyProperty(1) = "Fred"          ' Produces error
```

To assign values to MyProperty, assign it a whole array:

```
Dim anArray(1 To 2) As String
anArray$(1) = "Fred"
MyProperty = anArray
```

Too many items specified in input/output statement

More than 255 items were specified in one of the following:

- *A Print statement*
- *A Write statement*
- *An Input statement*

Reduce the number of items to fewer than 256.

ISELEMENT argument is not a list or variant: <name>

The first argument that you passed to the IsElement function is not the name of a list or the name of a variable of type Variant holding a list.

Change the argument to a list or a Variant holding a list, or remove the call to the IsElement function.

Missing list subscript for ISELEMENT argument: <list name>

You called the IsElement function and did not include the list tag, which is required. For example:

```
Dim myList List As Double
myList("Alex") = 12345
myList("Martin") = 23456
If IsElement(myList) = TRUE Then Print "Yes."                ' Illegal
If IsElement(myList("Mary")) = TRUE Then Print "Yes."        ' Legal
```

Specify a list tag when you call IsElement.

FORALL alias variable is not of same data type: <name>

You reused a ForAll reference variable, but the array, list, or collection being iterated over is of a different data type than the collection previously iterated over using the same variable. For example:

```
Dim X(10) As Integer
Dim Y(10) As Long

ForAll I In X
    ' ...
 End ForAll

ForAll I In Y       ' Error. I is an Integer above;
                    ' it can't be Long here.
End ForAll
```

Use a different variable: either an existing ForAll reference variable of the correct type, or a new variable.

FOR count variable already in use: <name>

You used the count variable of an outer For loop as the count variable of an inner For loop. The count variable of an outer For loop may not be reused as the count variable of an inner For loop. For example:

```
For X% = 1 To 10
    For X% = 1 To 5          ' Illegal. X% is already in use.
    ' ...
    Next
Next
```

Change the count variable in one of the For loops so that they are different from each other.

FORALL alias variable already in use: <variable name>

You used the reference variable of an outer ForAlL loop as the reference variable of an inner ForAll loop. The reference variable of an outer ForAll loop may not be reused as the reference variable of an inner ForAl l loop.

Rename the reference variable of the inner ForAll loop.

FORALL alias variable was previously declared: <name>

You used a previously declared variable as a ForAll reference variable. Previously declared variables may not be used as ForAll reference variables. A ForAll reference variable may only be used in a ForAll statement.

Rename the ForAll reference variable.

Illegal type suffix on FORALL alias variable: <name>

The ForAll reference variable's declaration or a reference to that variable contains a data type suffix character. Data type suffix characters are not allowed in either the declaration of, or references to, a ForAll reference variable.

Remove the suffix character from the variable's declaration or reference.

LISTTAG argument is not a FORALL alias variable

You used an invalid argument when you called the ListTag function. The ListTag function may only be passed the ForAll reference variable of the ForAll statement:

```
Dim Y List As String
ForAll X In Y
    Print ListTag(ABC) ' Illegal
    Print ListTag(X)    ' Legal
End ForAll
```

Replace the invalid argument in the ListTag function call with the ForAll reference variable where ListTag appears.

Maximum allowable code size exceeded

The module you are compiling contains more than 64K bytes of executable code.

Split the module into multiple modules and recompile.

Maximum allowable data size exceeded

The module you are compiling contains more than 64K bytes of data.

Split the module into multiple modules and recompile, or reduce the amount of data in the module.

Maximum number of errors reached

The maximum of twenty compilation errors has been reached, causing compilation to stop.

Fix the reported errors and recompile the program.

Too many nested INCLUDEs

You have more than 16 levels of nested %Include directives. This may be due to circular %Include references.

Reduce the number of nested %Include directives to 16 or fewer. Remove any circular %Include references.

Maximum allowable symbol table size exceeded

The module you are compiling contains more than 64K bytes of symbols (names).

Split the module into multiple modules and recompile, or reduce the number of names in the module.

Token is too long

The maximum length of a LotusScript token (a sequence of characters with a unique meaning) may not exceed the maximum allowable length of a string constant (16K characters) plus its delimiters.

Reduce the length of the token.

Name does not match FOR count variable: <name>

The variable name that immediately follows the Next keyword in a For...Next block does not match the corresponding For count variable.

Match the name with its corresponding For count variable, or remove the name that follows Next: the name is optional.

Undefined label: <label name>

The sub, function, or property just compiled contains a reference to a label that was never defined. The line number of the error message identifies the End Sub, End Function, or End Property statement that marks the end of the offending procedure. Labels must be defined within the same scope in which they are referenced.

Define the label in the sub, function, or property that refers to it.

Type mismatch on: <name>

The following conditions could have caused this error.

- *You tried to pass an argument to a sub or function by reference, but the data types of the argument and the corresponding parameter do not match.*

  Pass the argument by value or pass an argument of the correct data type.

- *You tried to pass an array, a list, or an object reference to a function or sub, but the corresponding parameter is not defined as one of these or as a Variant.*

  Pass an argument of the correct kind.

- *You tried to pass a scalar value to a function or sub, but the corresponding parameter is defined as an array, a list, or an object reference variable.*

  Pass an argument of the correct kind.

- *You tried to assign an instance of a user-defined data type to a Variant. For example:*

```
Type myType
    A As Integer
End Type
Dim typeInst As myType
Dim varV As Variant
varV = typeInst                ' Illegal
```

  This is not allowed. Remove the assignment statement.

- *You used a Set statement to try to assign a value other than an object reference to an object reference variable (or a Variant holding an object reference). For example:*

Class MyClass

'...

End Class

Dim X As New MyClass

Dim N As Integer

N% = 5

Set X = N% ' Illegal

This is not allowed. Remove the assignment statement.

- *You used a Set statement to try to assign an object reference to something other than an object reference variable or a Variant. For example:*

Class MyClass

'...

End Class

Dim X As New MyClass

Dim N As Integer

Set N% = X ' Illegal

This is not allowed. Remove the assignment statement.

- *You used a Set statement to try to assign an object reference variable of one class to an object reference variable of another class. You can only do this when the variables designate instances of the same class or when the target variable designates a base class and the variable whose value is being assigned designates a derived class from that base. For example:*

~~*Class MyClass*~~

~~*   '...*~~

~~*End Class*~~

~~*Class BaseClass*~~

~~*   '...*~~

~~*End Class*~~

~~*Class DerivedClass As BaseClass*~~

~~*   '...*~~

~~*End Class*~~

~~*Dim A As New MyClass*~~

~~*Dim B As New BaseClass*~~

~~*Dim D As New DerivedClass*~~

~~*Set B = A   ' Illegal*~~

~~*Set D = B   ' Illegal*~~

~~*Set B = D   ' Legal*~~

~~*Remove or revise the assignment.*~~

- ~~*You used a Set or Set...New statement to try to create an object (class instance) and assign a reference to it to a variable that is not an object reference variable or a Variant.*~~

~~*Class MyClass*~~

~~*   '...*~~

~~*End Class*~~

~~*Dim X As New MyClass*~~

~~*Dim N As Integer*~~

~~*Set N% = New MyClass   ' Illegal*~~

Remove or revise the assignment.

- *You used a Set or Set...Bind statement in which the target variable is not an object reference variable or a Variant holding an object reference.*
- *You used a With statement whose target is not an object reference variable or a Variant containing an object reference. The With statement can only be used to operate on objects.*
- *A ReDim statement contains a data type that does not match the data type in the declaration of the array, or the data type in a previous ReDim statement whose target was that array.*

Change the data type in the ReDim statement so that it matches the data type of the declaration or previous ReDim statement whose target was that array, or remove the data type from the ReDim statement—once you specify a data type for a dynamic array, it is not necessary to specify the data type again in subsequent ReDim statements.

- *You used a variable declared as a non-numeric data type as the count variable in a For statement.*

Replace the count variable with a variable of the appropriate numeric type.

Missing argument for: <function name>

The following conditions could have caused this error:

• *You did not include a required argument when you called a function. For example:*

```
Function MyFunction(A As Integer, B As Integer) As Integer
    '...
End Function
anInt% = MyFunction%(5)        ' Illegal because MyFunction takes two arguments
```

Supply the missing argument in the function call.

• *A comma was not followed by an argument. For example:*

```
Function MyFunction(A As Integer, B As Integer) As Integer
    '...
End Function
anInt% = MyFunction(,3)        ' Illegal
```

Remove the comma, or specify the argument.

Missing array bound for: <array name>

You used a ReDim statement to define the dimensions of a dynamic array but included an extra comma (,) in the bounds list. For example:

```
Dim anArray()
ReDim anArray(,1,2)      ' Illegal comma at beginning of bounds list
ReDim anArray(1,2,)      ' Illegal comma at end of bounds list
ReDim anArray(1,,2)      ' Illegal comma immediately after another comma
```

Remove the misplaced comma.

~~*Missing argument to constructor for: <class name>*~~

~~*You used a Dim or Set statement to create a new instance of a user-defined class or product class and omitted one or more of the arguments that the class's constructor sub (Sub New) requires. For example:*~~

```
Class MyClass
    Sub New(aString As String, anInt As Integer)
        '...
    End Sub
End Class
Dim X As New MyClass("ABC")        ' Illegal. MyClass's Sub New expects two
                                   ' arguments: a string and then an integer.
```

~~*or:*~~

```
Class MyClass
    Sub New(aString As String, anInt As Integer)
        '...
    End Sub
End Class
Dim X As MyClass
Set X = New MyClass("ABC")         ' Illegal. MyClass's Sub New expects two
                                   ' arguments: a string and then an integer.
```

~~*Include all of the required arguments in the appropriate order in the Dim or Set statement.*~~

Missing collection index for: <name>

You included empty parentheses in a reference to a collection. This is not allowed. You can either remove the empty parentheses or insert the appropriate subscript. Removing the parentheses makes the reference be to the entire collection, while including the subscript makes the reference be to a single element in the collection.

Missing array subscript or collection index for: <name>

Either of two conditions could have caused this error:

- *You included empty parentheses in a reference to the return value of a function or property. This is not allowed. Assuming that the function or property returns a Variant containing an array, list, or reference to a collection, you can either remove the empty parentheses or insert the appropriate subscript or subscripts. Removing the parentheses makes the reference be to the entire array, list or collection, while including the subscript or subscripts makes the reference be to a single element in the array, list, or collection. For example:*

```
Dim anArray(5) As Variant
Function MyFunction(someArray()) As Variant
    ' ...
    MyFunction = someArray
End Function
varV = MyFunction(anArray)()           ' Illegal.
varV = MyFunction(anArray)             ' Legal. Returns the contents
                                       ' of the array.
varV = MyFunction(anArray)(1)          ' Legal. Returns the first element
                                       ' of the array.
```

- *You included empty parentheses in a reference to a class member function that returns an array, list, or collection.*

You can either remove the empty parentheses or insert the appropriate subscript or subscripts.

Illegal executable code at the module level

An executable statement appears at the module level. The product in which you are running LotusScript does not allow executable statements at the module level.

Move the executable statement into a procedure. If you want the statement to be executed when the module is loaded, move the statement into the Initialize sub. If you want the statement to be executed when the module is unloaded, move the statement into the Terminate sub.

Statement is illegal in a subprogram

You used one of the following statements within a LotusScript procedure:

- *Class*
- *Declare*
- *Function*
- *One of the Deftype statements*
- *Option Base, Option Compare, Option Declare, or Option Public*
- *Property Get*
- *Property Set*
- *Sub*
- *Type*
- *Use*
- *UseLSX*

You can only use these statements at the module level.

Move the statement to the module level.

Name too long: <name>

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.

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
```

Illegal construction of type instance: <instance name>

You used the keyword New in the declaration of a variable of a user-defined data type or in a statement assigning a value to a variable of a user-defined data type. The keyword New is not allowed in referring to variables of a user-defined type. For example:

```
Type MyType
    A As Integer
End Type
Dim X As New MyType      ' Illegal
```

or:

```
Set X = New MyType       ' Illegal
```

You use the keyword New to declare or assign a value to an object reference variable, that is, an instance of a class.

Remove New from the declaration or assignment statement.

Illegal specification of array bounds for: <array name>

You included array bounds in specifying a parameter in the declaration of a sub or function. A parameter that is an array should contain empty parentheses only.

Specify the parameter with empty parentheses. For example:

```
Function    Comper (X(5,2) As Integer) As Single      ' Illegal
Function Comper ( X ( ) As Integer) As Single         ' Corrected form
```

Variable not declared: <name>

You referred to an undeclared variable while the Option Declare statement was in effect.

Implicit declarations are illegal when Option Declare is in effect.

Declare the variable, or remove the Option Declare statement.

Parent SUB NEW has arguments, SUB NEW is required for: <class name>

You defined a derived class that has no Sub New. If the corresponding base class's Sub New requires arguments, the derived class must have a Sub New that provides those arguments. For example:

```
Class BaseClass
    Sub New (X As Integer)
    End Sub
End Class
Class DerivedClass As BaseClass
End Class
Dim ObjRefVar As New DerivedClass      ' Illegal because BaseClass's
                                       ' Sub New needs to be passed an
                                       ' integer.
```

Define a Sub New for the derived class whose signature includes the arguments required by the base class's Sub New . For example:

```
Class BaseClass
    Sub New (X As Integer)
    End Sub
End Class
Class DerivedClass As BaseClass
    Sub New (X As Integer)
    End Sub
End Class
Dim ObjRefVar As New DerivedClass(5)      ' Legal
```

SUB NEW arguments do not match parent's SUB NEW arguments

The parameters in the derived class's Sub New differ in number or type from the parameters in the base class's Sub New. For example:

```
Class Baseclass
    Sub New (X As Long)
    End Sub
End Class
Class Derivedclass As Baseclass
    Sub New (X As Long, Y As Long) ' Illegal, because Y is not a parameter
                                   ' in Baseclass's Sub New.
    End Sub
End Class
```

Do one of the following:

- *In the derived class's Sub New declaration, specify which arguments to pass to the base class's Sub New, for example as follows:*

```
Class Derivedclass As Baseclass
    Sub New (X As Long, Y As Long), Baseclass (X)
    End Sub
End Class
```

- *Redefine the derived class's Sub New so that its parameters match those of the base class's Sub New. For example:*

```
Class Derivedclass As Baseclass

    Sub New (X As Long)

    End Sub

End Class
```

Product class does not have a New method: <class name>

You tried to assign a product object reference to a variable and used the keyword New but the product class does not have a New method. Use a Set...Bind statement instead.

PUBLIC is not allowed in this module

An Option Public statement, or a declaration of a name as Public, appears in the current module. The product in which you are running LotusScript does not allow Public declarations anywhere within this module.

Move the Option Public statement or the Public declaration to a module where Public declarations are allowed. Alternatively, remove the Option Public statement or remove the keyword Public from the declaration.

Must be a sub: <procedure_name>

A module-level procedure named "Initialize" or "Terminate" must be a sub. Initialize and Terminate are special subs at the module level. Initialize is executed when the module is loaded and Terminate when it is unloaded.

~~*Illegal pass by value: <argument name>*~~

~~*One of the following happened:*~~

~~• *In declaring or defining a function or sub, you used the ByVal keyword in specifying a parameter that is an array, list, object reference, or user-defined data type. Arrays, lists, instances of user-defined data types, and object references cannot be passed by value, so ByVal is not allowed in the specification of one of these as a parameter. For example:*~~

```
Type MyType
    A As Integer
End Type
Declare Function MyFunction(ByVal X As MyType) ' Illegal
```

~~*Remove ByVal from the declaration.*~~

~~• *You tried to pass an array, list, object reference, or instance of a user-defined data type by value in a call to a LotusScript procedure. For example:*~~

```
Type MyType
    A As Integer
End Type
Sub MySub(X As MyType)
' ...
End Sub
Dim Z As MyType
MySub(Z)                ' Illegal: this tries to pass by value.
MySub Z                 ' Legal: this passes by reference.
```

~~*or*~~

~~*Dim anArray(1 to 3) As String*~~

~~*Sub MySub2(Z As Variant)*~~

~~*' ...*~~

~~*End Sub*~~

~~*MySub2(anArray()) ' Illegal: this tries to pass by value.*~~

~~*MySub2 anArray() ' Legal: this passes by reference.*~~

~~*Pass the argument by reference. Remove the parentheses around the argument in the calling statement.*~~

Illegal numeric constant

You tried to define a numeric constant, assigning it a value that doesn't match the specified or default data type. For example:

```
Const ANINT = 1.2%        ' Illegal because 1.2 is not an Integer
```

Fix the numeric constant.

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.*

Numeric overflow

In defining a constant with the Const statement, you specified a numeric value that is too large for the specified or default data type:

- *The value is too large for the data type specified by the value's suffix character.*
- *If no suffix character is specified, the value is too large for a Double.*

For example:

```
Const X = 100000%            ' Illegal because the value is too large for
' the data type Integer
Const Y = 100000!            ' Legal
```

Change the suffix character to match the magnitude of the value, or specify a smaller value.

Parser 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 offending statement into multiple, less complex statements.

Member declared in a parent class

You tried to declare a member variable in a derived class using the same name as a member variable, sub, function, or property of the base class. This is not allowed.

The name space for variables also includes functions, subs, and properties. This means that if a name is used as a method name in a base class, it may not be used as a variable name in a derived class. For example:

```
Class BaseClass
    X As Integer
    Sub Y
    '...
    End Sub
End Class
Class DerivedClass As BaseClass
    X As Integer                    ' Illegal
    Y As Integer                    ' Illegal
    '...
End Class
```

Declare the variable using a different name.

Method was declared as something else in a parent: <method name>

You used a Declare statement or a Function, Sub, Property Set, or Property Get statement to declare or define a procedure within the definition of a base class. In subsequently defining a derived class, you declared or defined a function or sub with the same name as the base class's procedure, but the procedure types are different. For example:

```
Class BaseClass
    Function MyProcedure As Integer
    '...
    End Function
End Class
Class DerivedClass As BaseClass
    Sub MyProcedure          ' Illegal because MyProcedure is a different
    '...                     ' kind of procedure in BaseClass
    End Sub
End Class
```

Change the base class procedure or the corresponding derived class procedure so that both are either subs, functions, or properties.

~~*Property was declared as something else in a parent: <property name>*~~

~~*You used a Declare statement or a Function or Sub statement to declare or define a function or sub within the definition of a base class. In defining a derived class, you used Declare or a Property Get or Property Set statement to declare or define a property with the same name as the base class's function or sub. For example:*~~

```
Class Baseclass
    Sub MyProcedure
    '...
    End Sub
End Class
Class DerivedClass As Baseclass
    Property Set MyProcedure          ' Illegal because MyProcedure is a sub rather
    '...                              ' than a property in Baseclass.
    End Property
End Class
```

~~*Change the base class's procedure or the corresponding procedure in the derived class so that both are either subs, functions, or properties*~~

Name previously declared: <name>

A name that has already been declared in the current scope is being declared again in the same name space. Names that reside in the same name space may only be declared once in a scope. Each module, sub, function, property, class, and user-defined data type has a particular scope. LotusScript has three separate name spaces:

- *Variable, Const, Sub, Function, and Property names*
- *Type and Class names*
- *Labels*

For example, a module-scope variable may have the same name as a class defined in that module, because variable names and class names are in different namespaces and therefore don't conflict. However, a module-scope variable may not have the same name as a function defined in that module.

The name space where a name resides doesn't depend on whether the name is declared Public , Private, or external (declared by the external Declare statement). All of these share the same name space.

Remove the duplicate declaration.

Name previously referenced in this scope

You declared a variable in an outer scope. You then referred to this variable in an inner scope and then declared it in that scope. For example:

```
Dim X As Integer
Sub MySub
    X% = 5
    Dim X As Integer    ' Illegal because the preceding assignment
                        ' statement referred to the X declared in
                        ' outer scope
End Sub
```

Move the declaration of the variable in the inner scope so that it precedes the assignment statement, or remove the declaration of the variable in the inner scope. Moving the declaration of the variable in the inner scope creates a local variable that shadows the one in the outer scope, while removing the declaration lets you refer to the variable in the outer scope from within the inner scope.

Derived class may not be PUBLIC when parent is PRIVATE: <class name>

You defined a Public class whose base class is Private. The base class from which a Public class is derived cannot be Private.

Change the definition of the base class to Public, or change the derived class to Private.

Member of PUBLIC class or type is instance of a PRIVATE class or type: <member name>

Within the definition of a Public class or user-defined data type, you declared as Public a member variable that refers to a Private class or user-defined data type, or you included a Public method that returns an instance of a Private class or user-defined type. For example, in the following code, the definition of the variable B produces this error condition:

```
Private Type MyType
    A As Integer
End Type
Public Class MyClass
    Public B As MyType    ' Illegal because MyType is defined as Private
End Class
```

Change the Public class or user-defined data type to Private, or the Private class or user-defined data type to Public.

PROPERTY GET and SET arguments do not match: <property_name>

The corresponding parameters to Get and Set for a property are not of the same type.

Property type does not match parent property: <property name>

You used a Declare statement or a Property Get or Property Set statement to declare or define a property within the definition of a base class. In defining a derived class, you used Declare or Property Set or Property Get to declare or define a property with the same name as the one in the base class, but with a different data type. For example:

```
Class BaseClass
    Property Get MyProperty As Integer
    '...
    End Property
End Class
Class DerivedClass As BaseClass
    Property Get MyProperty As Double
    '...
    End Proper      ty          ' Illegal because MyProperty's return type
                                ' was defined as Integer in BaseClass
End Class
```

Change the data type of the derived class's property or the corresponding property in the base class so that they match.

Number of arguments do not match for PROPERTY GET and SET <property_name>

The number of parameters to Get and Set for a property are not the same.

Property signature does not match parent property: <property name>

You used a Declare statement or a Property statement to declare or define a property within the definition of a base class. In subsequently defining a derived class, you declared or defined a property of the same name as the base class's property but with a different signature.

One of the following does not match:

- *The data type*
- *The number of parameters*
- *The data type of one of the parameters*
- *The data structure of one of the parameters*

Change the signature of the base class property or of the corresponding property in the derived class so that they match.

PROPERTY GET and SET must have same data type

You declared a property's data type in a Property Get statement differently from the property's data type in the corresponding Property Set statement. The property must have a single declared data type.

Change the data type in one statement to the data type in the other.

PROPERTY GET and SET must have same storage class and visibility

One of the following occurred:

- *You declared a property's variables to be Static by default in either the Property Get statement or the Property Set statement, but not in both. The declarations must agree: either both or neither must specify Static.*

Change either statement to agree with the other.

- *You declared a property's scope in a Property Get statement differently from the property's scope in the corresponding Property Set statement. The property must have a single scope: either Public or Private.*

Make them both Public or Private.

Illegal PUBLIC instance of PRIVATE class or type: <instance name>

You declared a Public instance of a Private user-defined data type or class, or a Public function or property that returns an instance of a Private class.

Make the class or type Public, or make the instance Private.

Bounds must be specified in REDIM of: <array name>

You used the ReDim statement but did not specify the bounds of the array. A ReDim statement must specify bounds.

Specify the bounds within the ReDim statement.

Property is read-only: <property name>

You attempted to apply a Property Set statement to a property of a product object but the product has defined that property as read-only. This means that you can retrieve but cannot modify the property's current value.

Remove the Property Set statement.

Cannot assign into collection item

You tried to assign a value to a collection item. You can retrieve items in a collection but you cannot assign values to them.

Remove the assignment statement.

Method signature does not match parent method: <method name>

You used a Declare statement or a Function or Sub statement to declare or define a procedure within the definition of a base class. In subsequently defining a derived class, you declared or defined a procedure of the same kind (a function or sub) with the same name as the base class's procedure but with a different signature.

One of the following does not match:

- *The return type*
- *The number of parameters*
- *The data type of one of the parameters*
- *The data structure of one of the parameters*

Change the signature of the base class procedure or of the corresponding procedure in the derived class so that they match.

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.

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.

~~*Compiler statement stack overflow at: <token name>*~~

~~*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 offending statement into multiple, less complex statements.*~~

INCLUDE filename must be a string constant

Following the keyword %Include, you specified something other than a quoted literal.

For example:

Dim myFile As String

myFile$ = "C:\myroot\myfile.lss"

%Include myFile$ ' Illegal because %Include takes a

' quoted literal

%Include "C:\myroot\myfile.lss" ' Correct syntax

Use a quoted literal.

Procedure declaration may not be inside a control block

You tried to include a Function, Property Get, Property Set, or Sub statement inside one or another of the following block statements: Do, For, ForAll, If...Then...Else...EndIf, Select, While. This is not allowed. For example:

```
If 1 = 1 Then
    Sub MySub        ' Illegal
        '...
    End Sub
End If
```

Move the Function, Property Get, Property Set, or Sub statement to outside the block.

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.

Type suffix does not match data type: <name>

You referred to a variable, constant, function, or property with a data type suffix character that does not match its declared data type. If a variable is declared as a Variant , references to that variable may not contain any suffix character.

Change the suffix character to match the declared data type, or remove the suffix character.

Declaration may not contain type suffix and data type: <name>

You specified a declaration that contains both a data type suffix character and an As dataType clause . A declaration may not contain both, even if they match. For example:

Dim myInt% As Integer ' Illegal

Remove either the suffix character or the As dataType clause from the declaration.

Illegal character after directive

Your script contains a %If directive in which the keyword %Else or the block terminator %End If is followed on the same line by a space or Tab and then one or more characters other than the comment character ('). For example:

%If WIN16

%ElseIf WIN32

%End If Win16. ' Illegal

%End If ' Win16. (This is legal.)

Insert the comment character if a comment is intended, or remove the superfluous characters.

Too many arguments for: <subprogram name>

You specified more than the limit of 31 parameters in the declaration of a sub or function. The maximum number of parameters that may be specified for a sub or function is 31.

Reduce the number of declared parameters to 31 or fewer.

Maximum array dimensions (8) exceeded: <array name>

You either declared an array with more than eight dimensions or you used more than eight subscripts in referring to an array. An array can have a maximum of eight dimensions.

If the problem is that the declaration of the array specifies more than eight dimensions, reduce the number of dimensions in the declaration to at most eight. If the problem is that you used more than eight subscripts in a statement referring to an array, reduce the number of subscripts in that statement.

File contains too many source lines

The source file contains too many lines.

Split the source file into two or more files.

~~Too many nested WITHs~~

~~You tried to nest a series of With statements to more than 16 levels. This is not allowed.~~

TYPE declaration has no members

You have a Type declaration with no members. A Type declaration must contain at least one variableName As dataType statement.

Add at least one member to the Type declaration, or remove the Type declaration.

Type suffix character required on: <name>

A variable that was implicitly declared with a data type suffix character was used without the suffix character. When a variable is implicitly declared with a suffix character, all subsequent references must contain the suffix character. A reference without the suffix character is treated as an implicit declaration of an already declared variable. This is illegal (a variable can't be declared twice).

Append the suffix character to the variable name when you refer to it.

~~*TYPE may not have instance of itself as a member: <instance name>*~~

~~*You declared an instance of the user-defined data type being defined as a member of itself. The definition of a user-defined data type may include an instance of another user-defined data type as a member, but not an instance of itself. For example:*~~

```
Type MyFirstType
    X As Integer
End Type
Type MySecondType
    Y As MyFirstType          ' This is legal
    Z As MySecondType         ' This is illegal
End Type
```

~~*Remove the invalid member declaration.*~~

PROPERTY GET not defined for: <property name>

You tried to retrieve the value of a property for which you did not define a Property Get procedure. For example:

```
Dim myInt As Integer
Dim myOtherInt As Integer
Property Set MyProp As Integer
    myInt% = MyProp%
End Property
MyOtherInt% = MyProp% ' Illegal because there is no
                      ' Property Get MyProp defined.
```

Define a Property Get procedure for the property whose value you want to retrieve.

Class or type name not found: <name>

You used a name that does not refer to an existing class or user-defined data type where one of these was required. You used the name in one of the following contexts:

- *A variable declaration, as in:*

Dim X As ClassName

Dim X As User-definedTypeName

- *A derived class declaration, as in:*

Class NewClassName As ClassName

Class ClassName As ClassName is also illegal even if ClassName exists because a class may not be derived from itself.

- *A Set statement, as in:*

Set X = New ClassName

- *A base class reference in a derived class method, as in:*

Call ClassName..MethodName

- *A Bind statement (product classes only), as in:*

Set X = Bind ClassName (objectName)

Declare the class or user-defined data type before you refer to it.

PROPERTY SET not defined for: <property name>

You tried to assign a value to a property, but did not define a Property Set procedure for the property. For example:

```
Dim myInt As Integer
Property Get MyProp As Integer
    MyProp% = myInt%
End Property
MyProp% = 3          ' Illegal because there is no
                     ' Property Set MyProp defined
```

Define a Property Set procedure for the property to which you want to assign a value.

Numeric underflow

In defining a constant with the Const statement, you specified a numeric value that is too small for the specified or default data type:

- *The value is too small for the data type specified by the value's suffix character.*
- *If no suffix character is specified, the value is too small for a Double.*

For example:

```
Const X = .1E-300!          ' Illegal because the value is too small for
                            ' the data type Single
Const X = .1E-300#          ' Legal
```

Change the suffix character to match the magnitude of the value, or specify a larger value.

Unknown statement

The compiler could not parse the statement on the line specified in the error message. If a statement was intended, check the legal syntax for the statement. If a comment was intended, designate the line as a comment line. Otherwise, remove the incorrect text.

~~*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 statement with the appropriate keyword.*~~

Unterminated %IF, %ELSEIF, or %ELSE directive

Your script contains a %If directive to which there is no corresponding %End If. For example:

%If WIN16

%ElseIf WIN32

' End of script. Error message appears here because there is no %End If.

Insert a %End If in the appropriate place in the script.

Unterminated %REM block

You used a %Rem keyword with no corresponding %End Rem. Beginning with the unpaired %Rem, all lines of the script were read as comments.

Insert the corresponding %End Rem.

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.

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 items.

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.

Unterminated multiline string

You omitted the vertical bar (|) that marks the end of a multiline string; or you omitted the close brace (}) that marks the end of a multiline string; or you used a brace as one delimiter and the "|" character as the other. For example:

Print |Hi,

Martin.

' ... ' Illegal because there is no matching vertical bar

Print |Hi,

Martin.} ' Illegal because the delimiters don't match.

Check for any unpaired or improperly paired multiline string delimiters and enclose the string appropriately.

LEN argument must be a variable or string expression

You called the Len function and specified as its argument something other than a string expression or the name of a variable. For example:

```
Print Len(123)          ' Illegal because 123 is a numeric constant
Print Len("123")        ' Legal. Returns the number of characters in
                        ' the string "123" (3).
Dim X As Integer
Print Len(X%)           ' Legal. Returns the number of bytes allocated
                        ' to store an integer value in memory (2).
```

Make the argument a string expression or the name of a variable.

Wrong number of array subscripts for: <array name>

The number of subscripts in an array reference does not match the number of defined dimensions for the array.

Change the number of subscripts to match the number of defined dimensions for the array.

LotusScript Compiler Error Messages

To see Help for a specific LotusScript message, select the message from the alphabetical listing.

.. not valid outside of class scope

A

Argument does not match forward declaration: <argument name>

Arguments not legal in declaration of: <sub name>

Array size exceeds maximum: <array_name>

B

Bounds must be specified in REDIM of: <array name>

C

Cannot assign a value to: <name>

Cannot assign into a collection item

Cannot forward declare CLASS or TYPE

Cannot open included file: <file name>

Cannot subclass: <class name>

CASE ELSE must be the last CASE in a SELECT statement

Class is not a parent of this class: <class name>

Class is not specified on BIND into: <name>

CLASS or TYPE declaration may not be inside a control block

Class or Type name not found: <name>

Collection item is not an instance

Compiler statement stack overflow at: <token name>

Conflicting option

D

Declaration may not contain type suffix and data type: <name>

Declaration not valid in TYPE scope: <name>

Declaration of external subprogram is not legal inside a class

DELETE not valid on: <name>

Derived class may not be PUBLIC when parent is PRIVATE: <class name>

DIM required on declarations in this scope

Duplicate forward declaration: <name>

Duplicate label: <label name>

Duplicate option

Duplicate range specifier

E

Empty parentheses not legal on: <name>

Error in EVALUATE macro

Error number must be INTEGER

Error number must be INTEGER constant: <name>

EVALUATE argument must be a string constant

Event handler must be a FUNCTION

Event handler must be a SUB

Event handler must be a LotusScript SUB or FUNCTION: <handler name>

Expected expression before end of argument list for: <function name>

F

File contains too many source lines

FOR count variable already in use: <name>

FOR count variable must be a scalar variable: <name>

FORALL alias variable already in use: <name>

FORALL alias variable is not of same data type: <name>

FORALL alias variable was previously declared: <name>

G

GET and SET PROPERTY must have same data type

H

I

Illegal array bound for: <array name>

Illegal BYVAL

Illegal BYVAL on arguments to: <subprogram name>

Illegal call to: <sub name>

Illegal character after continuation character

Illegal character after directive

Illegal character after %INCLUDE directive

Illegal constant expression for: <CONST name>

Illegal construction of type instance: <instance name>

Illegal constructor clause on: <sub name>

Illegal data type for argument: <argument name>

Illegal data type for external argument: <argument name>

Illegal declaration in this scope: <name>

Illegal DEFtype statement after declaration

Illegal directive

Illegal executable code at the module level

Illegal EXIT <EXIT type>

Illegal external argument: <argument name>

Illegal function return type for: <function name>

Illegal name for class or type: <name>

Illegal numeric constant

Illegal ON ERROR statement

Illegal OPTION BASE after array declaration

Illegal OPTION EXPLICIT after implicit declaration

Illegal OPTION PUBLIC after declaration

Illegal parenthesized reference: <name>

Illegal pass by value

Illegal pass by value: <argument_name>

Illegal PRIVATE declaration of: <name>

Illegal product constant: <name>

Illegal property type for: <property name>

Illegal PUBLIC declaration of: <name>

Illegal PUBLIC instance of PRIVATE class or type: <instance name>

Illegal range specifier

Illegal REDIM on: <name>

Illegal reference to: <name>

Illegal reference to array or list: <array or list name>

Illegal reference to FORALL alias symbol: <name>

Illegal RESUME statement

Illegal scope for PUBLIC or PRIVATE on: <name>

Illegal second parenthesized expression

Illegal single-line IF

Illegal specification of array bounds for: <array name>

Illegal STATIC on: <name>

Illegal string length constant for: <name>

Illegal type suffix on name: <name>

Illegal TO in reference to: <name>

Illegal type suffix on FORALL alias variable: <name>

Illegal type suffix on keyword: <keyword>

Illegal type suffix on name: <name>

Illegal use of array or list element as FORALL target

Illegal use of ERASE

Illegal use of escape character

Illegal use of escape character in identifier: <name>

Illegal use of NEW on array or list declaration: <name>

Illegal use of NEW or DELETE

Illegal use of parentheses

Illegal use of property: <property name>

Illegal use of UNICODE or LMBCS keyword

Illegal USE or USELSX statement after declaration

Illegal value for OPTION BASE

INCLUDE filename must be a string constant

ISELEMENT argument is not a list or Variant

J

K

L

Label is Illegal outside of a subprogram

LEN argument must be a variable or string expression

LIB name must be a string constant

LISTTAG argument is not a FORALL alias variable

M

Maximum allowable code size exceeded

Maximum allowable data size exceeded

Maximum allowable symbol table size exceeded

Maximum array dimensions (8) exceeded: <array name>

Maximum number of errors reached

ME not valid outside of class scope

Member declared in a parent class

Member is not a subprogram: <member name>

Member of a PUBLIC class is instance of a PRIVATE class: <member name>

Method signature does not match parent method: <method name>

Method was declared as something else in a parent: <method name>

Missing argument for: <function name

Missing argument to constructor for: <class name>

Missing array bound for: <array name>

Missing array subscript or collection index for: <name>

Missing collection index for: <name>

Missing list subscript for ISELEMENT argument: <list name>

Must be a sub: <name>

N

Name does not match FOR count variable: <name>

Name not found: <name>

Name previously declared: <name>

Name previously referenced in this scope

Name too long: <name>

Name was forward declared as something else: <name>

Named product class instance not valid here

Not a member: <name>

Not a product class: <name>

Not a product class instance: <name>

Not a PUBLIC member: <name>

Not a sub or function name: <name>

Not an array, list, collection, or Variant: <name>

Not an event name: <name>

Not an instance name:<name>

Number of arguments does not match for PROPERTY GET and SET: <property name>

Number of arguments does not match forward declaration: <subprogram name>

Numeric overflow

Numeric underflow

O

Out of memory

P

Parent SUB NEW has arguments, SUB NEW is required for: <class name>

Parser stack overflow at: <token name>

Procedure declaration may not be inside a control block

Product class does not have a NEW method: <class name>

PROPERTY GET and SET arguments do not match: <property name>

PROPERTY GET and SET must have same data type

PROPERTY GET and SET must have same storage class and visibility

PROPERTY GET not defined for: <property name>

Property is read-only: <property name>

PROPERTY SET not defined for: <property name>

PROPERTY signature does not match parent propertyr: <property name>

Property type does not match parent property: <property name>

Property was declared as something else in a parent: <property name>

PUBLIC is not allowed in this module

PUBLIC symbol is declared in another module

Q

R

Reference must contain exactly one subscript: <name>

Return type does not match forward declaration: <function name>

S

SET may only be used on class instance assignments

SET required on class instance assignment

Size of data cannot exceed 32K in this scope

Size of data cannot exceed 64K in this scope

Statement is illegal in CLASS block: <keyword>

Statement is illegal in TYPE block: <keyword>

Statement is illegal in a subprogram

Statement is illegal outside of a subprogram

Storage class or visibility does not match forward declaration: <subprogram name>

SUB NEW arguments do not match parent's SUB NEW arguments

T

Token is too long

Too many arguments for: <subprogram name>

Too many items specified in input/output statement

Too many labels specified in ON...GOTO statement

Too many nested INCLUDEs

Too many nested WITHs

TYPE declaration has no members

TYPE may not have instance of itself as a member: <instance name>

Type mismatch on: <name>

Type suffix character required on: <name>

Type suffix does not match data type: <name>

U

Undefined label: <label name>

Unexpected/Expected

UNICODE and LMBCS strings must be declared BYVAL

Unknown statement

Unterminated <keyword> block

Unterminated multiline string

Unterminated %REM block

Unterminated square bracket reference

Unterminated string constant

Unterminated %IF, %ELSEIF, or %ELSE directive

Unterminated %REM block

Unterminated <token name> block

USE or USELSX name must be a string constant

V

Variable not declared: <name>

Variable required: <name>

W

Wrong data type for argument <argument name> in event handler

Wrong number of arguments for: <subprogram name>

~~*Wrong number of arguments for event handler: <sub name>*~~

~~*Wrong number of arguments to constructor for class: <class name>*~~

~~*Wrong number of array subscripts for: <array name>*~~

~~*Wrong return type in event handler <handler name>*~~

~~*X*~~

~~*Y*~~

~~*Z*~~

LotusScript Run-time Error Messages

To see Help for a specific LotusScript message, select the message from the alphabetical listing.

A

ADT error: Control procedure missing

Array size exceeds maximum limit

Attempt to access an uninitialized dynamic array

Automation-Object argument count

Automation-Object argument type mismatch

Automation-Object cannot create

Automation-Object error

Automation-Object file name error

Automation-Object member not found

B

Bad argument to external function

Bad attribute

Bad DLL calling convention

Bad file mode

Bad file name

Bad file name or number

Bad record length

Bad record number

C

Can't set attribute for file

Cannot assign into collection item

Cannot destroy active instance

Cannot find external name <name>

Cannot find module <module name>

Cannot rename with different drive

Collection item not found

Compiler error

Conflicting modes supplied

D

Data too big for record

Device I/O error

Device unavailable

Disallowed

Disk full

Disk not ready

Division by zero

Duplicate PUBLIC name in USE module: <module name>

E

Error accessing product object

Error accessing product object method

Error accessing product object property

Error creating product object

Error in constant expression evaluation

Error in EVALUATE macro

Error in loading DLL

Error loading USE or USELSX module

Event does not exist

Event handler argument count mismatch

Event handler argument type mismatch

Event handler not attached

Event handler procedure type mismatch

Event handler return type mismatch

Expression out of range

External function not found

F

File already exists

File already open

File not found

File not open

File not readable

File not writable

FOR loop not initialized

~~*ForAll container invalid or modified*~~

~~*G*~~

~~*H*~~

~~*I*~~

~~*Illegal circular USE: <module name>*~~

~~*Illegal DELETE*~~

~~*Illegal file number*~~

~~*Illegal function call*~~

~~*Illegal LIKE pattern*~~

~~*Illegal operation for file mode*~~

~~*Illegal REDIM*~~

~~*Illegal REDIM of fixed array*~~

~~*Illegal use of function*~~

~~*Illegal use of member*~~

~~*Illegal use of property*~~

~~*Illegal use of read-only property*~~

~~*Illegal use of sub*~~

~~*Input past end of file*~~

~~*Instance member does not exist*~~

~~*Internal error*~~

~~*Invalid ^ operator operands*~~

~~*Invalid collection item*~~

~~*Invalid module file*~~

~~*Invalid pattern string*~~

Invalid use of NULL

J

K

L

List item does not exist

List reference must contain exactly one subscript

LISTTAG argument not a list element

M

Missing argument

Module already loaded

Module in use

N

Named product object does not exist

Name used as a method is not a method

No RESUME

Not a collection object

Not a product object

Not a PUBLIC member

O

Object variable not SET

Opcode <opcode name> not implemented

Operation not supported on this platform

Out of memory

Out of stack space

Out of string space

Out of system stack space

Overflow

P

Path not found

Path/File access error

Permission denied

PROPERTY GET not defined

PROPERTY SET not defined

Q

R

RESUME without error

RETURN without GOSUB

S

SET required on class instance assignment

String too large

Sub or function not defined

Subscript out of range

T

Too many calls into module

Too many files

Type mismatch

Type mismatch on external name <name>

Type suffix does not match actual data type

U

Unable to open file

Underflow

Unknown class instance

Unsupported argument type to external function

Unsupported return type for external function

User-defined error

V

Variable is read-only

Variant does not contain a container

Variant does not contain an object

W

Wrong number of arguments for a method

Wrong number of arguments for property

Wrong number of array subscripts

Wrong number of collection indices

X

Y

Z

ADT error: Control procedure missing

The Lotus product from which you invoked LotusScript is missing a procedure needed to manage product objects.

Record the error message number and contact Lotus Customer Support.

Named product object does not exist

You tried to use a product object that does not exist.

Refer to an existing product object; or, in the LotusScript product from which you invoked LotusScript, define the object you are trying to use.

Array size exceeds maximum limit

The total storage space in memory of the dynamic array exceeds the allowable maximum of 64K. For example:

```
ReDim MyArr(-20000 To 20000) As Integer
' This declares an array with 40,001 elements of 2 bytes each.
' The declared array size is greater than 64K.
```

Use the ReDim statement to decrease the array size.

Bad attribute

You supplied an illegal file attribute number using the FileAttr function.

Supply a legal attribute number (either 1 or 2) that specifies the type of information you want.

Bad DLL calling convention

You are using a C-callout function to call a DLL entry point with a different calling convention than the one used to implement the DLL entry point.

Define the correct argument-passing protocol in the C-callout function to implement the DLL entry point.

Bad file mode

You used an Open statement to try open a file in a mode that is incompatible with the file's access type. For example, opening a file for Output that has Read access causes this error.

If you intended to open this file, change either the file's access type, or change the For clause specification in the Open statement.

Bad file name

You specified a file using an invalid DOS file name.

Specify a correct file name using DOS file naming rules.

Bad file name or number

You tried to access a file that does not exist, or you specified a file number that is currently not assigned to a file. For example, using Print # to print to a file that has not first been opened generates this error.

- *Check the spelling of the file name and correct it if it is wrong.*
- *Open the file first with the specified file number.*
- *Check the file number and correct it if it is wrong.*

Illegal file number

You specified a file number outside of the range 1 to 255 in an Open statement.

Specify a file number between 1 and 255.

Bad record length

You tried to give a record length for a file that is incompatible with that specified in the Open statement for the file, or you specified a negative record length.

For record-oriented I/O with random files, use the Len = reclen clause of the Open statement to define the record length. When opening the file for reading, reclen should be the same as when the file was opened for writing.

Bad record number

You tried to read from a file using a record number that is either invalid (negative) or out-of-bounds (larger than the number of records in the file).

If you are using a Get statement, make sure that the record numbers are within the bounds of the file. Numbering of records begins at 1.

Cannot destroy active instance

You attempted to delete an instance of a class that is still in use in your program.

Record the error message number and contact Lotus Customer Support.

Cannot rename with different drive

You used a Name statement to try to rename a file to a different drive than the one where it is currently stored. You cannot change the drive on which a file is stored when you rename the file.

Rename the file without changing the drive where the file is currently stored.

Bad argument to external function

In a Declare statement, you declared an external function using an invalid argument.

Replace the invalid argument with a valid one.

External function not found

The dynamically linked library (DLL) named in a Declare statement for an external function was found, but the declared function was not found in the DLL.

If the function name was misspelled in the Declare statement, correct it.

If the function name was correct but the wrong DLL was specified, specify the correct DLL.

Unsupported argument type to external function

In a Declare statement, you declared an external function using an unsupported type for an argument.

Replace the argument type with a valid one.

Unsupported return type for external function

In a Declare statement, you declared an external function using an unsupported type for the return value.

Replace the return type with a valid one.

Illegal circular USE: <module name>

The module currently being compiled contains a Use statement whose target module contains a Use statement whose target module refers to the current module (possibly by transitivity) in another Use statement. Use statements cannot be circular: if module A uses module B, then B, or any module that B uses, may not use A.

Reconfigure the set of Use statements to remove the circularity.

Wrong number of collection indices

You used more than a single subscript in referring to a member of a collection. For example, assuming a collection class IntegerCollection:

```
Dim IntCol As New IntegerCollection("astring",10)
'...
Dim varV As Variant
Set varV = IntCol
Print varV(1,1)              ' Illegal.
Print varV(1)                ' Legal.
```

Use one, and only one, subscript when referring to a collection member.

Collection item not found

You tried to refer to a nonexistent member of a collection. For example, assuming a collection class IntegerCollection:

```
Dim varV As Variant
Dim IntCol As New IntegerCollection("astring",10)
Print IntCol(3)                ' Illegal because the collection doesn't have any
   ' members.
```

Add members to the collection before trying to refer to them; specify an index that identifies a member; or remove the reference.

Cannot assign into collection item

An attempt is made to write to a member when accessing a collection object through a variant.

Invalid Collection item

You attempted to access a member of a collection, but the product was unable to comply with your request correctly.

Record the error message and contact Lotus Customer Support.

Compiler error

The function signature of an external C-callout function has been corrupted.

Record the error message number and contact Lotus Customer Support.

Conflicting modes supplied

You used an Open statement to try to open a file in a mode that is incompatible with its access type. For example, opening a file for Output that has Read access causes this error.

If you intended to open this file, change either the file's access type, or change the For clause specification in the Open statement.

Cannot set attribute for file

You tried to supply a legal file attribute using the FileAttr function, but could not do so because the file is write-protected or is being used by another program.

Verify whether or not you can access the file, or close the file in the other program.

Data too big for record

You tried to write data into a record that is too small for the amount of data you are writing.

Write less data into the record, or create another file with a larger record size to hold the data.

Device I/O error

The following conditions could have caused this error:

- *You tried to write to a read-only disk.*

Change the disk's read-only access to read-write access.

- *You tried to open a file on a protected diskette.*

Make sure there is a diskette in the drive and that it has read-write access.

Device unavailable

You specified an invalid drive.

Specify a drive that exists on your system.

Operation is disallowed in this session

The product from which you are running LotusScript has disabled the function, statement, or directive that you attempted to use.

Remove the function call, statement, or directive.

~~Disk full~~

~~You tried to save a file on a disk that did not have enough room for the file.~~

~~Save the file on another disk.~~

~~*Disk not ready*~~

~~*The disk drive door is not closed.*~~

~~*Close the disk drive door.*~~

Division by zero

In a mathematical operation, there was an attempt to divide by zero. It is impossible to divide by zero.

Check the appropriate operand for a zero value before using it as a divisor.

Duplicate PUBLIC name in USE module: <module name>

You declared as Public a name that is also declared as Public in another loaded module, a module that was loaded in executing a Use statement.

Determine the duplicate Public name and change its declaration in one module or the other.

Error creating product object

You tried to create an instance of a product class but the product encountered an error condition (such as Out of Memory) and was unable to create the object.

Record the error message and contact Lotus Customer Support.

Error accessing product object

You tried to delete an instance of a product class but the product encountered an error condition when you tried to do so.

Record the error message and contact Lotus Customer Support.

Error accessing product object method

You tried to refer to a method (member sub or function) of an instance of a product class but the product encountered an error condition when you tried to do so.

Record the error message and contact Lotus Customer Support.

Error accessing product object property

You tried to refer to a property of an instance of a product class but the product encountered an error condition when you tried to do so.

Record the error message and contact Lotus Customer Support.

Error in EVALUATE macro

When you tried to execute an Evaluate function or statement, the product containing the macro to which the function or statement refers encountered an error condition.

Record the error message and contact Lotus Customer Support.

Error in loading DLL

The dynamically linked library (DLL) specified in a Declare statement could not be found.

If the Declare statement does not specify the path of the DLL, LotusScript seeks the DLL as follows, in order:

- *In the working directory*
- *In the directories on the search path specified by the DOS environment variable PATH*

If the Declare statement specified the path of the DLL, correct the DLL name or the path.

If the Declare statement did not specify the path of the DLL, do one of the following:

- *Specify the path of the DLL in the Declare statement.*
- *Move the DLL to the working directory; or change the working directory to the directory that contains the DLL.*
- *Add the location of the DLL to the DOS environment variable PATH.*

Event handler argument count mismatch

The event you specified in an On Event statement requires a different number of parameters than are found in the specified event handler's signature. For example, assume that the Moved event defined for the product class Walden requires three parameters (an object reference and two integers):

```
Sub GoodSub(Source As Walden, X As Integer, Y As Integer)
'...
End Sub
Sub BadArgNum(Source As Walden, X As Integer)
'...
End Sub
Dim objRefVar As New Walden("ABC")
On Event Moved From objRefVar Call GoodSub              ' Legal.
On Event Moved From objRefVar Call BadArgNum            ' Illegal: BadArgNum
    ' has only 2 parameters,
    ' but Moved requires 3.
```

Change the event handler's signature to make it have the required number of parameters.

Event handler argument type mismatch

The event you specified in an On Event statement requires that one or more of the event handler's parameters be different in data type from what appears in the event handler's signature. For example, assume that the Moved event defined for the product class Walden requires three parameters (an object reference and two integers):

```
Sub GoodSub(Source As Walden, X As Integer, Y As Integer)
'...
End Sub
Sub BadArgType(Source As Walden, X As Integer, Y As String)
'...
End Sub
Dim objRefVar As New Walden("ABC")
On Event Moved From objRefVar Call GoodSub          ' Legal.
On Event Moved From objRefVar Call BadArgType       ' Illegal: BadArgType's
                                                    ' third parameter should
                                                    ' be an Integer.
```

Change the event handler's signature so that its parameters are of the required data types.

Event handler procedure type mismatch

The event handler for an object is a sub and the user-defined procedure is a function, or vice-versa, when attaching an event handler to an object through a variant.

Event handler return type mismatch

The return type of the event does not match the return type of the function when attaching an event function to an object through a variant.

Event handler not attached

You tried to remove an event handler from a product object with an On Event statement, but the handler is not bound to the object.

Remove the On Event statement or specify the correct handler.

Expression out of range

You used a numeric expression whose value at run time is out of the legal range, in one of these contexts:

- *As the numeric expression in an On...GoTo or On...GoSub statement.*

  The value of the expression must be between 0 and 255 inclusive.

- *As the designated error number in an Err or Error statement.*

  The error number must be positive or 0.

- *As the designated error number in an Error function call.*

  The error number must be positive.

Respecify the expression in the statement or in the function call, to ensure that its value falls within the legal range.

File already exists

You tried to create a file with the same name as a file that already exists on disk.

Specify a different file name.

File already open

You used the Open statement on a file that is already open.

Use Close to close the open file, or remove the Open statement that attempts to open it.

File not found

You referred to a file that cannot be found.

If you do not specify the path of the file, LotusScript seeks the file as follows, in order:

- *In the working directory*
- *In the directories on the search path specified by the DOS environment variable PATH*

If the file specification included the path, correct the file name or the path.

If the file specification did not include the path, then do one of the following:

- *Specify the path.*
- *Move the file to the working directory; or change the working directory to the directory that contains the file.*
- *Add the location of the file to the DOS environment variable PATH.*

File not open

In a statement or function that requires an open file, you specified a file that is not open.

Use the Open statement to open the file.

File not readable

You used an Open statement to try to open a file that cannot be read at this time. It may currently be locked by another program, or it could be corrupted and therefore cannot be opened.

If the file is currently locked by another program, access the other program and close the file there.

File not writable

You tried to write to a file that is marked read-only on disk.

- *Open the Windows File Manager, choose File Properties to remove the read-only attribute from the file, and then return to LotusScript to save the file.*
- *Use the DOS Attrib command to remove the read-only attribute from the file.*
- *Save the file under a different file name.*

ForAll container invalid or modified

You tried to assign a value to the target in a ForAll block. For example:

```
Dim anArray(3) As Integer
Dim varV As Variant
varV = anArray
ForAll X In varV
    '...
    varV = 4          ' Illegal.
End ForAll
```

Remove the assignment statement.

FOR loop not initialized

One of the following conditions could have caused this error:

• *You used the GoTo statement to transfer control to a For statement.*

You cannot use the GoTo statement to transfer control to a For statement (though you can use it to exit a For loop).

• *The count variable that you specified in a For statement does not have a valid initial value.*

Make sure the count variable of the For statement is properly initialized. For example, in the statement For I = X, I and X must be of the same data type.

Illegal DELETE

You tried to use the Delete statement to delete a member of an object rather than the object itself. The Delete statement requires a plain object name. For example:

```
Class MyClass
   Public X As Integer
End Class
Dim varV As Variant
Set varV = New MyClass
Delete varV.X                    ' Illegal.
Delete varV                      ' Legal.
```

Remove the Delete statement or change its argument to an unqualified object name.

Illegal function call

The following conditions could have caused this error:

- *You tried to pass a negative subscript to an array.*
- *You tried to pass an invalid argument to the ACos function, the ASin function, or the ATn2 function.*
- *You tried to pass an invalid argument to the Asc function. The empty string ("") is an invalid argument.*
- *You used an invalid string expression with the Val function.*
- *You tried to pass an invalid argument to the Chr function.*
- *You tried to use the Date function to set an invalid system date.*
- *You tried to use the Time function to set an invalid system time.*
- *You tried to pass an invalid argument to the DateNumber function or the TimeNumber function.*
- *You tried to pass NULL, EMPTY, the empty string (""), or a number less than 1 or greater than 255 to the Environ function.*
- *You tried to use a negative record number in the Get statement or the Put statement.*
- *You tried to pass an invalid start position or an invalid count to the Instr function or the InstrB function.*
- *You tried to pass an array to the Len function or the LenB function.*
- *You tried to pass a negative value to the Log function or the Sqr function.*
- *You tried to pass a negative value or a value greater than 65K as an argument to a function.*
- *You tried to pass an invalid window style argument (it must be an integer from 1 to 9 inclusive) to the Shell function.*

- *You tried to pass an invalid comparison argument (it must be 0 or 1) to the StrCompare function.*
- *You tried to activate a program using the ActivateApp statement, but the program was not found.*
- *The string specified in the SendKeys statement contained an unmatched parenthesis, an illegal key name, or an illegal repeat count; or the string was too long to be processed.*

Illegal LIKE pattern

The pattern specified for a Like operation is illegal for one of the following reasons:

- *You specified a range of characters in square brackets, but the second character is earlier in the collating sequence than the first character. For example:*

"a" Like "[e-a]"

Reverse the order of the characters in the illegal range specification.

- *You specified an open square bracket without a close square bracket. For example:*

"a" Like "[abc"

Supply the close square bracket. If you want to specify an open square bracket as a character to match, enclose it in square brackets:

"[" Like "[[]"

Illegal operation for file mode

You tried to perform an operation on a file that is illegal for the file's mode. For example, using the Get statement on a sequential file generates this message.

Use the FileAttr function to determine the file's mode before performing this operation on the file.

Illegal REDIM

You used a ReDim statement in a context in which it is inappropriate:

• *In referring, with the Preserve keyword, to a variable of type variant that doesn't already contain an array. For example:*

```
Dim varV As Variant
varV = 5
ReDim Preserve varV(1 To 3)        ' Illegal
```

Remove the keyword Preserve if you want varV to hold an array, or remove the Redim statement.

• *You referred to a member variable of a class as though it were an array, though it isn't. For example:*

```
Class AClass
   Public X As Integer
End Class
Dim varV As Variant
Set varV = New AClass
ReDim varV.X(1 To 3)                ' Illegal, because X isn't an array.
```

Declare X as a dynamic array or remove the ReDim statement.

• *You referred to a member variable or property of a class as though it held or returned a dynamic array rather than a fixed array. For example:*

```
Class AClass
    Public X(1 To 2) As Integer
End Class
Dim varV As Variant
Set varV = New AClass
ReDim varV.X(1 To 3)                ' Illegal, because X is a fixed array.
Define X as a dynamic array or remove the ReDim statement.
```

Illegal use of function

You defined a function as a member of a class and specified its return type as something other than Variant or object reference. You then referred to that function as though its return type were an object reference or a Variant holding an array, list, or object reference. For example:

```
Class MyClass
  Function MyFunction(X As Integer) As Integer
  '...
  End Function
End Class
Dim varV As Variant
Set varV = New MyClass
Print varV.MyFunction.F(1)          ' Illegal.
Print varV.MyFunction.Something     ' Illegal.
```

Remove the reference or change the function's return type to Variant.

Illegal use of MEMBER

An argument list is specified when accessing an object member variable through a variant.

Illegal use of property

You defined a property as a member of a class and then referred to that property in an inappropriate way. For example:

```
Class MyClass
   Property Set MyProp As Integer
   '...
   End Property
   Property Get MyProp As Integer
   '...
   End Property
End Class
Dim varV As Variant
Set varV = New MyClass
varV.MyProp                ' Illegal: a reference to a property must occur
   ' in a statement that assigns or retrieves the
   ' property's value.
X% = varV.MyProp(1)        ' Illegal: integer variables can't be subscripted.
```

Remove the reference or correct its syntax.

Illegal use of sub

You defined a sub as a member of a class and then referred to that sub as though it were a member function, property, or variable. For example:

```
Class MyClass
    Sub MySub
    '...
    End Sub
End Class
Dim varV As Variant
Set varV = New MyClass
X = varV.MySub                    ' Illegal: a sub doesn't have a return value.
varV.MySub = 5                    ' Illegal: you can't assign a sub a value.
```

Remove the reference or redefine the sub as the appropriate type of class member.

Input past end of file

One of the following conditions could have caused this error:

- *You tried to read past the end of the file.*

Use the EOF function to check for the end-of-file.

- *An Input statement tried to read in more values than are present in the last record in the file.*

Adjust the number of values being read so that the number read from the last record is less than or equal to the number of values in the last record.

Internal error

An internal error occurred.

Record the error message and contact Lotus Customer Support.

List reference must contain exactly one subscript

You declared a list variable as a class member. When you subsequently referred to that list, you either omitted a subscript or included more than one subscript. A reference to a list must include one, and only one, subscript. For example:

```
Class MyClass
   Public myList List As Integer
End Class
Dim varV As Variant
Set varV = New MyClass
Print varV.myList(1,1)              ' Illegal: too many subscripts.
Print varV.myList()                 ' Illegal: missing subscript.
```

Supply one, and only one, subscript.

Invalid pattern string

You used an invalid pattern string with the Like operator.

Record the error message number and contact Lotus Customer Support.

Invalid ^ operator operands

In an expression whose operator is the exponentiation operator (^), the pair of operands is invalid.

The expression X ^ Y (the base X raised to the exponent Y) cannot be evaluated when

- *X is 0, and Y is negative or 0: for example, 0 ^ -2*
- *X is negative, and Y is not an integer: for example, -1 ^ 2.2*

Respecify the expression, or the computations leading up to it, to ensure that the operands will have legal values when the ^ operator is applied to them.

Invalid use of NULL

You tried to convert a NULL value to another value type. NULL cannot be converted to another value type.

For example, the function call CInt (NULL) is an invalid use of NULL. This function call attempts to convert NULL to an integer explicitly.

Implicit conversion of NULL is also invalid, as in the following sequence of statements:

S = NULL

For I = 1 To 5 Step S

Next

In a For statement, the step value must be numeric. LotusScript attempts to convert the value in S to a number when executing the For statement above. This is an invalid use of NULL.

Use the IsNull function to determine if a value is NULL.

Type mismatch on external name <name>

The currently executing module contains a Use statement whose target module contains a Public name to which the currently executing module refers. The data type of the name in the target module has been changed since the currently executing module was compiled.

Restore the name's original data type in the target module, or change the name's data type in the currently executing module to the new data type.

Cannot find module <module name>

You tried to access a Public name in a module that is not loaded. At compile time, that module was accessed indirectly: it was made available by a Use statement in another loaded module, not the current module.

Insert a Use statement in the current module to make the other module available before accessing the Public name.

Cannot find external name <name>

The currently executing module contains a Use statement whose target module contains a Public name to which the currently executing module refers. That name has been changed in the target module since the currently executing module was compiled. Restore the original name in the target module, or change the name in the currently executing module to the new name.

Error loading USE or USELSX module

The target that you specified in a Use or UseLSX statement cannot be found or is invalid (possibly because of version skew).

Supply thename of an existing file of the appropriate format or remove the Use or UseLSX statement.

List item does not exist

You used a list tag that does not exist in a list.

Before accessing a list element with that tag, use the IsElement function to test if the element exists in the list.

Missing argument

You called a member sub or function of a product class and omitted one or more of the arguments that it expected. For example, assume a product class Walden that has a member sub Move that has two integer parameters:

```
Dim varV As Variant
Set varV = New Walden("ABC")
varV.Move 5                    ' Illegal: Walden's Move method has two
    ' parameters, not one.
```

Supply the required number of arguments in the call, or remove the calling statement.

Module in use

You tried to unload the currently running module.

Remove any attempt to unload the currently running module.

Too many calls into module

You have exceeded the allowable maximum number (65K) of nested calls to functions or subs within a single module.

Record the error message number and contact Lotus Customer Support.

Invalid module file

You tried to use a module that is incompatible with this release of LotusScript.

Verify that the script source language is compatible with this release of LotusScript, and recompile the module.

Module already loaded

In the Use statement, you named a module that is already present.

If the named module is the one you want to load, remove the Use statement. Otherwise, change the name in the Use statement.

Event does not exist

The event that you specified in an On Event statement is not defined for the specified product object. For example, suppose that ProdADT is a product object for which NotAnEvent is not a defined event:

```
Sub MySub(Source As ProdADT)
'...
End Sub
Set prodObjRef = New ProdADT("astring")
On Event NotAnEvent From prodObjRef Call MySub   ' Illegal.
```

Remove the On Event statement or specify an event defined for the object.

Instance member does not exist

You referred to a nonexistent member of a class. For example:

```
Class MyClass
'...
End Class
Dim varV As Variant
Set varV = New MyClass
Print varV.Something                ' Illegal because Something is not defined
    ' as a member of MyClass.
```

Define the member within the class, or remove the reference.

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 a user-defined object, 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 varV As Variant
Dim otherVarv As Variant
Dim X As New MyClass
Set varV = X
otherVarV = varV                    ' Illegal.
otherVarV = New varV                ' Illegal.
Set otherVarV = varV                ' Legal.
Set otherVarV = New varV ' Legal.
```

Include the Set keyword in the assignment statement or remove the statement.

Operation not supported on this platform

You tried to use a LotusScript function, statement, or directive that your operating system does not support. For example, the CreateObject statement is not supported under OS/2 or UNIX.

Remove the unsupported function call, statement, or directive.

No RESUME

You are using an On Error statement in a procedure, but have not included a Resume statement.

Insert one or more Resume statements at the appropriate points in the script.

Not a collection object

You referred to a product object as though it were a collection, but it isn't a collection.

For example, assuming the product class ProdADT, which is not a collection class:

```
Dim varV As Variant
Set varV = New ProdADT("abc")
ForAll X In varV                    ' Illegal.
'...
End ForAll
```

Remove the reference or replace its target with the name of a collection.

Variant does not contain a container

You referred to a variable of type Variant as though it held an array, list, or collectionbut it does not hold one of these. For example:

Dim varV As Variant

varV(1) = 5 ' Illegal.

Remove the reference or insert a statement before it that assigns a list, fixed array, or reference to a collection to the Variant.

LISTTAG argument not a list element

Within a ForAll loop that iterates over the elements of an array, you supplied the reference variable as the argument to the ListTag function. You can apply ListTag only to the ForAll reference variable for a list, not an array. For example:

```
Dim anArray(10)
ForAll X In anArray
    Print ListTag(X)                ' Illegal. ListTag can only refer to a list,
                                    ' not to an array.
End ForAll
```

Remove the incorrectly used ListTag function.

Name used as a method is not a method

You referred to something as though it were a member function or sub of a class when no such function or sub has been defined for that class. For example:

```
Class MyClass
'...
End Class
Dim varV As Variant
Set varV = New MyClass
Print varV.Something("ABC")        ' Illegal: Something is not defined
                                   ' as a sub or function in MyClass.
```

Remove the reference or define the sub or function as a member of the class.

Variant does not contain an object

You referred to a variable of type Variant as though it contained an object reference, but no such reference has been assigned to it. For example:

```
Dim varV As Variant
varV.Something          ' Illegal.
```

Remove the reference or insert a statement before it that assigns an object reference to the Variant.

Not a product object

Where a reference to a product object was expected in an On Event statement or an Evaluate function or statement, you used a reference to a user-defined object. This is not allowed. For example:

```
Class MyClass
'...
End Class
Sub MySub
'...
End Sub
Dim varV As Variant
Set varV = New MyClass
On Event Click From varV Call MySub          ' Illegal.
On Event Click From varV Remove MySub        ' Illegal.
X = Evaluate("mymacro",varV)                 ' Illegal.
```

Remove the function or statement, or change it so that it refers to a product object.

Not a PUBLIC member

You referred to a variable, property, function, or sub that was defined as a Private member of a class. Private members are not visible outside of the class to which they belong. For example:

```
Class MyClass
   X As Integer                              ' X is Private by default.
   Private Function Z As Integer
   '...
   End Function
End Class
Dim varV As Variant
Set varV = New MyClass
varV.X% = 10                                 ' Illegal: X is Private.
anInt% = varV.Z%                             ' Illegal: Z is Private.
```

Remove the reference or, if possible, change the definition of the class member from Private to Public.

Object variable not set

You tried to access an instance of a LotusScript class or product class, but either of the following was true:

- *The object reference variable you used does not hold a reference to any object. (Its value is NOTHING.)*

  Use the Set statement to assign the variable a reference to an object.

- *The object reference variable has been deleted.*

  Remove the statement that refers to the deleted variable.

Automation-Object argument count

You called a method of an OLE Automation object and included too few or two many arguments. The number of arguments must be the same as the number of parameters defined for the method.

Check the documentation for the OLE Automation object to ascertain the method's parameters.

Automation-Object argument type mismatch

You called a method of an OLE Automation object and included one or more arguments whose data type differs from the corresponding parameters in the method's definition.

The data type of each argument must be the same as the data type of the corresponding parameter.

Check the documentation for the OLE Automation object to ascertain the data type of each of the method's parameters.

Automation-Object cannot create

You called CreateObject or GetObject but LotusScript could not interpret the argument or arguments in the call.

Make sure that the arguments designate a valid application and class and, if appropriate, a valid path.

Automation-Object error

An error occurred when you tried to refer to an OLE Automation object.

Check the syntax of the statement that caused the error, and check the documentation for the OLE Automation object to which you tried to refer.

Automation-Object file name error

The path that you specified in a call to GetObject is invalid.

Specify a valid path or remove the GetObject statement.

Automation-Object member not found

You referred to an undefined member of an OLE Automation object, or you attempted to assign a value to an OLE Automation object property that is read-only.

Check the documentation for the OLE Automation object to ascertain its members and their status.

Unable to open file

The following conditions could have caused this error:

• *You tried to open a file that was not found.*

Verify that you specified the correct file name.

• *You tried to open a file that is currently locked by another active program on your system.*

Close the file in the other program.

• *You tried to open a file that has been corrupted.*

Recreate the file.

Opcode <opcode name> not implemented

A required operation code has not been implemented.

Record the error message number and contact Lotus Customer Support.

Out of memory

There is not enough system memory to perform an operation.

To free memory on your computer, end one or more other programs that are currently in memory, other than the Lotus product running LotusScript.

Out of stack space

One of the following conditions could have caused this error:

- *You wrote a recursive function that never reaches its base case, and therefore never terminates itself.*

  Rewrite the function so that it reaches its base case.

- *You declared too many local variables in a procedure.*

  Remove a sufficient number of variable declarations in the procedure to free up stack space by rewriting the it as several smaller procedures. If you are using fixed arrays, declare them as dynamic.

Out of string space

There is too little available memory for string storage, either at compile time or at run time.

The maximum memory available for the set of all strings varies during compilation and during execution of your program. It cannot be larger than 64K at any time.

If your program includes many strings, or very long strings, either eliminate some strings, or restructure your program to limit the set of strings that must be kept in memory at any one time.

If your program includes a great many names, you may need to restructure it similarly. LotusScript creates and stores a string for each name. The string's length is the number of characters in the name. For example, if your program includes a definition of a type with several thousand members, string storage space may be exhausted.

Out of system stack space

You entered an expression that LotusScript is unable to evaluate because the expression contains too many elements. For example, an expression consisting of hundreds of values separated by arithmetic operators would cause this error because the result of each individual arithmetic operation has to be saved on the stack until they can all be combined to calculate the value of the expression as a whole, and there isn't enough room on the stack to save them all.

Break the expression up into smaller pieces handled by multiple statements.

Overflow

The result of a numeric operation, value conversion, or assignment is outside the range of allowable values for the result data type.

Do one or both of the following:

- *Change the numeric data type of one or more values being used in the operation, conversion, or assignment.*
- *Change the destination data type to accommodate the result. For example:*

```
Dim N As Long

I% = 30000                  ' Declare I implicitly as an Integer .
J% = 10000                  ' J is also an Integer .
Print (I% + J%)             ' Overflow from numeric operation. The number
                            ' 40000 cannot be represented as an Integer.
Print CInt(40000&)          ' Overflow from attempted conversion of
                            ' a Long value to an Integer value.
Invar% = 40000              ' Overflow from attempted assignment of
                            ' a large value to an Integer variable.
N = 40000                   ' No error. N was declared a Long.
MN = 40000                  ' No error. MN is implicitly declared a Long
                            ' by the assignment of a large value to it.
```

Path/file access error

One of the following conditions could have caused this error:

- *You tried to access a file or directory that is protected.*

Use the DOS Attrib command to change the attributes of the file or directory.

- *You tried to access a file that is currently locked by another program.*

Close the file in the other program.

~~*Path not found*~~

~~*You specified a path that cannot be found.*~~

~~*Check the path to make sure you specified it correctly and that it exists, and then respecify it.*~~

Permission denied

One of the following conditions could have caused this error:

- *You tried to access a file that is currently locked by another program.*

Close the file in the other program.

- *You tried to write to a file that has been write-protected with the Attrib command in DOS, or with the File Properties command in Windows.*

Open the Windows File Manager, choose File Properties to remove the read-only attribute from the file, then return to LotusScript and try again to write to the file.

PROPERTY GET not defined

A get operation is attempted through a variant on an object property that does not define Property Get.

PROPERTY SET not defined

A set operation is attempted through a variant on an object property that does not define Property Set.

Illegal use of read-only property

You tried to assign a value to a property of a product object, but the product has defined that property to be read-only. This means that you can retrieve but cannot assign that property's value.

Remove the assignment statement.

~~*Variable is read-only*~~

~~*A set operation is attempted on a product variable that is read-only.*~~

Illegal REDIM of fixed array

You used the ReDim statement to resize an existing fixed array. You can only use ReDim to declare a dynamic array, or to resize an existing dynamic array.

Remove the ReDim statement.

RESUME without error

You tried to execute a Resume statement outside of an error-handling routine. You cannot resume execution if an error has not occurred.

Insert the Resume statement within an error-handling routine.

RETURN without GOSUB

You executed a Return statement without having first transferred control in the procedure to a labeled statement using a GoSub statement or an On...GoSub statement.

Use a GoSub or On GoSub statement to transfer control to a labeled statement before executing a Return statement.

String too large

A string is generated at run-time that exceeds the size limit of 32,000 characters.

Sub or function not defined

You declared a sub or function with a Declare statement, and then tried to call it before defining it with a Sub or Function statement.

Define the function or sub.

Subscript out of range

When accessing an array, either the number of subscripts does not match the given array's defined dimensions; or the size of one or more subscripts does not match the given array's bounds.

- *If the subscripts are correct, redefine the array dimensions or bounds using the ReDim statement.*
- *If the array dimensions and bounds are correct, the subscripts must be respecified. For example:*

```
Dim TiArr(-1 To 1)
Print TiArr(0, 0)           'Error. Array is defined as having one
                            'dimension, and is being accessed as a
                            'two-dimensional array.
Print TiArr(-2)             'Error. The specified subscript falls outside of
                            'the array's defined bounds.
```

Type suffix does not match actual data type

You referred to a variable, constant, function, or property with a data type suffix character that does not match its declared data type. For example:

```
Class MyClass
   Public X As Integer
End Class
Dim varV As Variant
Set varV = New MyClass
Print varV.X$ ' Illegal because X was declared as an Integer.
```

Change the suffix character to match the declared data type, or remove the suffix character.

Too many files

You have too many files open in LotusScript.

Close some open files before attempting this file operation.

Type mismatch

One of the following conditions could have caused this error:

- *You attempted an operation on operands with conflicting data types.*
- *You assigned a value to a variable that has a different data type, and LotusScript cannot convert it automatically.*
- *You are passing a value as an argument that has a different declared data type, and LotusScript cannot convert it automatically.*
- *You used a string as the initial value, or as the To or Step value, in a For statement.*

Use the correct data type.

~~*Underflow*~~

~~*An internal error occurred.*~~

~~*Record the error message and contact Lotus Customer Support.*~~

Attempt to access an uninitialized dynamic array

Either of the following situations could have produced this error:

- *You tried to assign an uninitialized dynamic array to a Variant:*

```
Dim anArray() As Integer
Dim varV As Variant
varV = anArray                    ' Illegal.
```

Use the ReDim statement to assign bounds to the array before assigning the array to the Variant.

- *You tried to pass an uninitialized dynamic array to the LBound or UBound function:*

```
Dim anArray() As Integer
LB% = LBound(anArray)             ' Illegal.
```

Use the ReDim statement to assign bounds to the array before calling the LBound or UBound function.

Unknown class instance

An product object is returned for a class not registered with LotusScript.

User-defined error

A user-defined error occurred. You used the Error statement to create an error and assigned it a number that is not a LotusScript error number. You did not specify an error message in the Error statement, so LotusScript displays the default error message "User-defined error."

To display a more meaningful error message, provide the message string as the second argument to the Error s tatement.

Wrong number of arguments for method

You called a function or sub that is a member of a user-defined class and passed it either too few or too many arguments. For example:

```
Class MyClass
    Function MyFunction(A As Integer) As Integer
    '...
    End Function
End Class
Dim varV As Variant
Set varV = New MyClass
Dim X As Integer
X% = varV.MyFunction                    ' Illegal: too few arguments.
X% = varV.MyFunction(5,10)              ' Illegal: too many arguments.
```

Supply the correct number and type of arguments.

Wrong number of arguments for PROPERTY

The number of parameters do not match when accessing an object property through a variant.

Wrong number of array subscripts

.An array access through a variant has the wrong number of subscripts.

Error in constant expression evaluation

An error occurred in evaluating a constant expression. The error is explained in one of the following messages:

Division by zero

Illegal function call

Illegal Like pattern

Invalid ^ operator operands

Invalid use of NULL

Out of string space

Overflow

Abs function in LotusScript

Returns the absolute value of a number.

Syntax

Abs (numExpr)

Elements

numExpr

Any numeric expression.

Return value

Abs returns the absolute value of numExpr.

The data type of the return value is the same as the data type of numExpr, unless numExpr is a Variant. In that case, the following rules apply:

- *If numExpr contains a string that LotusScript can convert to a number, the data type is Double.*
- *If numExpr contains a value that LotusScript cannot convert to a number, the function returns an error.*
- *If numExpr contains a NULL, the return value is NULL.*

Usage

The absolute value of a number is its unsigned magnitude; for example, 3 and -3 both have an absolute value of 3.

{button ,AL(`LSAZ_ABS_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_SGN_FUNCTION',0)} See related topics

Examples: Abs function

```
Print Abs(12)                          ' Prints 12
Print Abs(-12)                         ' Prints 12
Print Abs(13 - 25)                     ' Prints 12
Print TypeName(Abs(-12))               ' Prints INTEGER
Dim someV As Variant
someV = "123"
Print Abs(someV)                       ' Prints 123
someV = NULL
Print Abs(someV)                       ' Prints #NULL#
```

ACos function in LotusScript

Returns the arccosine, in radians, of a number between -1 and 1, inclusive.

Syntax

ACos (numExpr)

Elements

numExpr

A numeric expression with a value between -1 and 1, inclusive.

Return value

ACos returns the arcosine, in radians, of the value of numExpr.

The range of the return value is zero to PI, inclusive.

The data type of the return value is Double.

If the value of numExpr is not in the range -1 to 1, inclusive, the function returns an error.

Usage

The arcosine of a number is the angle, in radians, whose cosine is equal to the value of that number.

{button ,AL(`LSAZ_ACOS_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_CONSTANTS;LSAZ_ASIN_FUNCTION;LSAZ_ATN_FUNCTION;LSAZ_ATN2_FUNCTION;LSAZ_COS_FUNCTION;LSAZ_SIN_FUNCTION;LSAZ_TAN_FUNCTION',0)} See related topics

Examples: ACos function

```
Dim rad As Double

Dim degrees As Double

' Assign the value PI/2, the angle whose cosine is 0.

rad# = ACos(0)

' Assign the value 90, the same angle in degrees.

degrees# = rad# * (180 / PI)

Print rad#; degrees#                    ' Prints 1.5707963267949   90
```

ActivateApp statement in LotusScript

Makes a program window the active window.

Syntax

ActivateApp windowName

AppActivate is acceptable in place of ActivateApp.

Elements

windowName

A string expression designating the program window to activate.

Usage

windowName is not case-sensitive. It must exactly match the leftmost characters of the program title that appears in the program window title bar. For example, if the program title of a running program window is "Lotus Notes - Workspace", then a windowName value of "Lotus Notes" will activate that window. If more than one program title matches windowName, LotusScript will choose one of the program windows.

ActivateApp can activate a minimized window, but cannot restore or maximize it. Use SendKeys to restore or maximize a window. Use Shell to start a program.

{button ,AL(`LSAZ_ACTIVATEAPP_STATEMENT_EX',1)} See example

{button ,AL(`LSAZ_SENDKEYS_STATEMENT;LSAZ_SHELL_FUNCTION;LSAZ_MACINTOSH_PLATFORM_DIFFERENCES;LSAZ_UNIX_PLATFORM_DIFFERENCES',0)} See related topics

Examples: ActivateApp statement

' Activate the Lotus Notes program window

' (assuming that Lotus Notes is already running).

' This would match a windows with the title "Lotus Notes - Workspace".

ActivateApp "Lotus Notes"

Asc function in LotusScript

Returns the platform-specific numeric character code for the first character in a string.

Syntax

Asc (stringExpr)

Elements

stringExpr

Any string expression.

Return value

Asc returns the numeric character code of the first character in stringExpr. The code represents the character in the character set of the platform on which you are running LotusScript.

The data type of the return value is Long.

If the value of stringExpr is NULL or the empty string (""), the function returns an error.

{button ,AL(`LSAZ_ASC_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_CHR_FUNCTION;LSAZ_UNI_FUNCTION',0)} See related topics

Examples: Asc function

Dim bigA As Long

Dim littleA As Long

bigA& = Asc("A")

littleA& = Asc("a")

Print bigA&; littleA& ' Prints 65 97

ASin function in LotusScript

Returns the arcsine, in radians, of a number between -1 and 1, inclusive.

Syntax

ASin (numExpr)

Elements

numExpr

A numeric expression with a value between -1 and 1, inclusive.

Return value

ASin returns the angle, in radians, whose sine is equal to the value of numExpr.

The range of the return value is -PI/2 to PI/2, inclusive.

The data type of the return value is Double.

If the value of numExpr is not in the range -1 to 1, inclusive, the function returns an error.

{button ,AL(`LSAZ_ASIN_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_CONSTANTS;LSAZ_ACOS_FUNCTION;LSAZ_ATN_FUNCTION;LSAZ_ATN2_FUNCTION;LSAZ_COS_FUNCTION;LSAZ_SIN_FUNCTION;LSAZ_TAN_FUNCTION',0)} See related topics

Examples: ASin function

```
Dim rad As Double

Dim degrees As Double

' Assign the value PI/2, the angle whose sine is 1.

rad# = ASin(1)

' Assign the value 90, the same angle in degrees.

degrees# = rad# * (180 / PI)

Print rad#, degrees#                    ' Prints   1.5707963267949   90
```

ATn2 function in LotusScript

Returns the polar coordinate angle, in radians, of a point in the Cartesian plane.

Syntax

ATn2 (numExprX , numExprY)

Elements

numExprX, numExprY

Any numeric expressions. At least one of the two must be non-zero. numExprX and numExprY designate the coordinates of a point in the Cartesian plane.

Return value

ATn2 returns the angular portion of the polar coordinate representation of the point (numExprX, numExprY) in the Cartesian plane.

The range of the return value is -PI to PI, inclusive.

If numExprX is 0, then ATn2 returns one of the following values:

- *-PI/2, if numExprY is negative*
- *PI/2, if numExprY is positive*

If numExprX is positive, then ATn2(numExprX, numExprY) returns the same value as ATn(numExprY / numExprX).

{button ,AL(`LSAZ_ATN2_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_CONSTANTS;LSAZ_ACOS_FUNCTION;LSAZ_ASIN_FUNCTION;LSAZ_ATN_FUNCTION;LSAZ_COS_FUNCTION;LSAZ_SIN_FUNCTION;LSAZ_TAN_FUNCTION',0)} See related topics

Examples: ATn2 function

```
Dim quad1 As Double, quad2 As Double, _
    quad3 As Double, quad4 As Double
' Assign the arctangents of four points in the plane.
quad1# = ATn2(1, 1)
quad2# = ATn2(-1, 1)
quad3# = ATn2(-1, -1)
quad4# = ATn2(1, -1)
' Print the value each angle in degrees.
Print quad1# * (180 / PI)                    ' Prints 45
Print quad2# * (180 / PI)                    ' Prints 135
Print quad3# * (180 / PI)                    ' Prints -135
Print quad4# * (180 / PI)                    ' Prints -45
```

ATn function in LotusScript

Returns the arctangent, in radians, of a number.

Syntax

ATn (numExpr)

Elements

numExpr

Any numeric expression.

Return value

ATn returns the angle, in radians, whose tangent is equal to the value of numExpr.

The range of the return value is -PI/2 (-90 degrees) to PI/2 (90 degrees), exclusive.

The data type of the return value is Double.

{button ,AL(`LSAZ_ATN_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_CONSTANTS;LSAZ_ACOS_FUNCTION;LSAZ_ASIN_FUNCTION;LSAZ_ATN2_FUNCTION;LSAZ_COS_FUNCTION;LSAZ_SIN_FUNCTION;LSAZ_TAN_FUNCTION',0)} See related topics

Examples: ATn   function

Dim rad As Double

Dim degrees As Double

' Assign the value PI/4, the angle whose tangent is 1.

rad# = ATn(1)

' Assign the value 45, the same angle in degrees.

*degrees# = rad# * (180 / PI)*

Print rad#; degrees#   ' Prints .785398163397449   45

Beep statement in LotusScript

Generates a tone on the computer.

Syntax

Beep

Usage

The tone that LotusScript produces depends on the sound-generating hardware in your computer.

{button ,AL(`LSAZ_BEEP_STATEMENT_EX',1)} See example

{button ,AL(`LSAZ_MESSAGEBOX_FUNCTION_AND_STATEMENT',0)} See related topics

```
Examples: Beep statement

' While a user-specified interval (in seconds) is elapsing, beep and

' count the beeps. Then tell the user the number of beeps.

Dim howLong As Single, howManyBeeps As Integer

Function HowManyTimes (howLong As Single) As Integer

    Dim start As Single, finish As Single, counter As Integer

    start! = Timer

    finish! = start! + howLong!

    While Timer < finish!

        Beep

        counter% = counter% + 1

    Wend

    HowManyTimes% = counter%

End Function

howLong! = CSng(InputBox _

    ("For your own sake, enter a small number."))

howManyBeeps% = HowManyTimes(howLong!)

MessageBox "Number of beeps:" & Str(howManyBeeps%)
```

Bin function in LotusScript

Returns the binary representation of a number as a string.

Syntax

Bin[$] (numExpr)

Elements

numExpr

Any numeric expression. If numExpr evaluates to a number with a fractional part, LotusScript rounds it to the nearest integer before deriving its binary representation.

Return value

Bin returns a Variant of DataType 8 (String), and Bin$ returns a String.

Return values will only include the characters 0 and 1. The maximum length of the return value is 32 characters.

Usage

If the data type of numExpr is not Integer or Long, then LotusScript attempts to convert it to a Long. If it cannot be converted, a type mismatch error occurs.

{button ,AL(`LSAZ_BIN_FUNCTION_EX',1)} See example

{button ,AL(`LSAZ_DATA_TYPE_CONVERSION;LSAZ_LITERAL_NUMBER_CONSTRUCTION_RULES;LSAZ_HEX_FUNCTION;LSAZ_OCT_FUNCTION',0)} See related topics

Examples: Bin function

Print Bin$(3) ' Prints "11"

' Converts Double argument to Long.

Print Bin$(3.0) ' Prints "11"

' Rounds Double argument, then converts to Long.

Print Bin$(3.3) ' Prints "11"

' Computes product 2.79, rounds to 3.0, then converts to Long.

*Print Bin$(3.1 * .9) ' Prints "11"*

Bracket notation in LotusScript

For applications built in some Lotus products, such as 1-2-3®, you can use names in brackets rather than object reference variables to identify Lotus product objects. To determine whether your Lotus product supports this notation, see the product documentation.

Syntax

[prodObjName]

Elements

prodObjName

The name understood by the product to identify an object (an instance of a product class).

Usage

In some cases, Lotus products assign names to objects, and in other cases you can use the product user interface to name the objects you create. In a spreadsheet, for example, A1 identifies a particular cell, and you could use the user interface to name a chart SalesTracking.

Bracket notation lets you use these names without declaring an object variable and binding it to the object. For example, the product might allow you to use:

[A1].Value = 247000

instead of:

Dim myCell As Cell

Set myCell = Bind Cell("A1")

myCell.Value = 247000

In some cases, the product uses bracket notation when it records transcripts of user

actions. This makes the transcripts easier to read and modify. For more information, see the product documentation.

The LotusScript compiler does not attempt to determine the class of objects that are identified with bracket notation, so any class syntax errors you make (such as the incorrect use of properties and other methods), will generate run-time errors, not compile-time errors.

You can also use empty brackets to identify the currently selected product object. Empty brackets are equivalent to leading dot notation. For example, if the current selection is a range named SalesQuotas, then

[].Print

and

.Print

are equivalent to

[SalesQuotas].Print

All three statements print the contents of the SalesQuotas range.

To include square brackets as text within a string, double the brackets. For example, if the current selection is a range named SalesQuotas[East], use the following syntax:

[SalesQuotas[[East]]].Print

{button ,AL(`LSAZ_BRACKET_NOTATION_EX',1)} See example

{button ,AL(`LSAZ_DIM_STATEMENT;LSAZ_SET_STATEMENT;LSAZ_DOT_NOTATION',0)} See related topics

Examples: Bracket notation

' Use the Chart class Print method to print the chart SalesTracking.

[SalesTracking].Print

Call statement in LotusScript

Calls a LotusScript sub or function.

Syntax 1

Call subOrFunction [([argList])]

Syntax 2

subOrFunction [argList]

Syntax 3