Interfacing your current financial tools with IBM® Lotus® Symphony™ Spreadsheets

Ed Ward
IBM Software Group
Software Engineer, Lotus Notes and Domino
Atlanta, GA

February 2009

Abstract: This white paper addresses the key areas that would help users adapt third-party financial applications such as Microsoft® Excel to IBM® Lotus® Symphony™ Spreadsheets. Originating from a request from IBM's Finance divisions, the paper is meant to help any user who wants to accelerate their migration to, and proficiency with, Lotus Symphony Spreadsheets.

Table of Contents

1 Introduction....3
2 Calculating with formulas....3
2.1 Overview of the DataPilot feature....3
2.2 Consolidating data....8
2.3 Applying Solve Equations....10
2.4 Using What-ifs....11
3 Macros and programming....11
3.1 Basics....12
3.2 Syntax....12
3.3 Lotus Symphony Basic IDE....18
3.4 Using macros....20
3.5 Event-driven macros....22
3.6 Debugging a basic program....23
4 Hyperion Essbase spreadsheet plug-in for Lotus Symphony....24
4.1 Software requirements....25
4.2 Installing the plug-in....25
4.3 Uninstalling plug-ins....27
5 Conclusion....28
6 Resources....29
7 About the author....29

1 Introduction

Lotus Symphony Spreadsheets is a spreadsheet application that you can use to calculate, analyze, and manage your data, as well as import and modify Microsoft® Excel spreadsheets. Symphony Spreadsheets also provides statistical and banking functions that you can use to create formulas to perform complex calculations on your data:

- **What-If calculations.** This is an interesting feature that lets you immediately view the results of changes made to one factor of calculations that are composed of several factors. For instance, you can see how changing the time period in a loan calculation affects the interest rates or repayment amounts. Furthermore, you can manage larger tables by using different predefined scenarios.

- **Arranging data.** With a few mouse clicks, you can reorganize your spreadsheet to show or hide certain data ranges, to format ranges according to special conditions, or to quickly calculate subtotals and totals.

- **Dynamic charts.** Symphony Spreadsheets lets you present spreadsheet data in dynamic charts that update automatically when the data changes.

- **Opening and saving Microsoft files.** You can use the Symphony Spreadsheets filters to convert Excel files, or to open and save files in a variety of other formats.

2 Calculating with formulas

All formulas begin with an equals sign. The formulas may contain numbers or text, and possibly other data such as format details. The formulas also contain arithmetic operators, logic operators, or function starts.

TIP: Remember that the basic arithmetic signs (+, -, *, /) can be used in formulas using the "Multiplication and Division before Addition and Subtraction" rule. Instead of writing =SUM(A1:B1), it is better to write =A1+B1. Parentheses can also be used.

The result of the formula =(1+2)*3 produces a different result than =1+2*3. It is also possible to nest functions in formulas and nest functions within functions. The Function Instant Pilot assists you with nested functions.

2.1 Overview of the DataPilot feature

Most spreadsheet users are familiar with the data summarization tool, PivotTable, found in data visualization programs. Lotus Symphony Spreadsheets provides an alternative function with limited features, called the DataPilot, which lets you combine, compare, and analyze large amounts of data.

You can view different summaries of the source data, and you can display the details for areas of interest and create reports. A table created with the DataPilot is an interactive table; data can be arranged, rearranged, or summarized according to different points of view.

As an example, suppose you have a data analysis table containing your company's sales

figures, not only for specific product groups, but also for branches and years (see figure 1). By using the DataPilot to quickly find the data of interest, you are able to filter the data.

Figure 1. Example DataPilot table

Sales					
Article	**Area**	**Sales person**	**1998**	**1999**	**2000**
Video	New York	Fisher	$200,000.00	$197,000.00	$220,000.00
Audio	New York	Fisher	$350,000.00	$235,000.00	$420,000.00
Accessories	New York	Fisher	$50,000.00	$60,000.00	$70,000.00
Video	New York	Fisher	$300,000.00	$180,000.00	$310,000.00
Audio	New York	Fisher	$270,000.00	$200,000.00	$270,000.00
Accessories	New York	Fisher	$25,000.00	$50,000.00	$40,000.00
Video	New York	Fisher	$189,000.00	$320,000.00	$234,000.00
Audio	New York	Fisher	$210,000.00	$240,000.00	$290,000.00
Accessories	New York	Fisher	$100,000.00	$80,000.00	$90,000.00
Video	Miami	Brown	$150,000.00	$160,000.00	$180,000.00
Audio	Miami	Brown	$210,000.00	$250,000.00	$300,000.00
Accessories	Miami	Brown	$10,000.00	$20,000.00	$15,000.00
Video	Kissimmee	Clark	$250,000.00	$300,000.00	$340,000.00
Audio	Kissimmee	Clark	$250,000.00	$290,000.00	$350,000.00
Accessories	Kissimmee	Clark	$100,000.00	$120,000.00	$130,000.00
Video	Washington	Smith	$200,000.00	$220,000.00	$250,000.00
Audio	Washington	Smith	$240,000.00	$260,000.00	$300,000.00
Accessories	Washington	Smith	$80,000.00	$60,000.00	$90,000.00
Filter					
Sum - 1998	**Area**				
Article	Kissimmee	Miami	New York	Washington	**Total Result**
Accessories	$100,000.00	$10,000.00	$175,000.00	$80,000.00	**$365,000.00**
Audio	$250,000.00	$210,000.00	$830,000.00	$240,000.00	**$1,530,000.00**
Video	$250,000.00	$150,000.00	$689,000.00	$200,000.00	**$1,289,000.00**
Total Result	**$600,000.00**	**$370,000.00**	**$1,694,000.00**	**$520,000.00**	**$3,184,000.00**

The DataPilot Area, Page, Column, Row, and Data are used to filter and display the data in a spreadsheet (see figure 2).

Figure 2. Filtering data

Area	Use to
Page	Filter the entire report based on the selected item in the DataPilot table.
Column	Display fields as columns at the top of the DataPilot table.
Row	Display fields as rows on the side of the report.
Data	Display summary numeric data.

2.1.1 Organizing the data using DataPilot

A DataPilot table provides a summary of large amounts of data. You can rearrange the DataPilot table to view different summaries of the data by selecting Manipulate > DataPilot. On the DataPilot menu, click one of the options shown in table 1, depending on what you want to do:

Table 1. Manipulate menu options

Option	Description
Create	Opens a window in which you can select the source for your DataPilot table, and then create your table.
Filter	Opens the Filter window.
DataPilot Table options	1. Ignore empty rows: Ignores empty fields in the data source. 2. Identify categories: Automatically assigns an empty row with the values from the preceding row. 3. Total columns: Calculates and displays the grand total of the column calculation. 4. Total rows: Calculates and displays the grand total of the row calculation.
Refresh	Refreshes the DataPilot table. This option is especially useful when you import a spreadsheet that contains a DataPilot table.
Delete DataPilot Table	Deletes the selected DataPilot table.
Show DataPilot Panel	Displays the DataPilot window.

2.1.2 Creating a DataPilot table

To create a DataPilot table, you must select the data source and enter the location of the table:

A. First, create an empty DataPilot table:

1. Position the cursor within a range of cells containing values, row headings, and column headings.

2. Select Manipulate > DataPilot > Create; the Create DataPilot Table window opens.

B. Select the data source by doing one of the following:

1. Click the Shrink button near the Range field to temporarily hide the window and make it easier to select a range. Select the range in the sheet and then click the Maximize button. The selected cells are the data source.

   NOTE: If you select a cell in a range of cells before you launch the Create DataPilot Table window, the cells around the selected cell is the data source.

2. Put the cursor in the Range field. Type your entries directly in the Range field to define the data source.

3. Click OK. A blank DataPilot table displays. You can move, resize, or float the window to be placed where you want it. Also, the DataPilot window can be docked on the left and right edge, but not on the top or bottom. The window is automatically shown or hidden when the focus is moved in or out of the DataPilot table.

4. If the window is closed, open the window by doing one of the following:

   - Right-click the DataPilot table and select Show DataPilot Table, or
   - Select Manipulate > DataPilot > Show DataPilot Table.

5. Add fields. All the fields from data source are listed in the Field field. The name of the fields are the content in the first cell of each column.

   1. Drag a field button to Page, Column, Row or Data to assign the field to the area.
   2. Drag a field button between Page, Column, Row or Data to move the field from an area to another.
   3. Drag a field button within the area to change the sequence of the fields in this area.
   4. Drag a field button from the area to Field to remove the field from the area.
   5. When dragging a field to the Data area, if the data source type is number, the field option is set to the Sum function by default. If the data source is text or date and so on, the field option is set to the Count function by default.
   6. Right-click the field name and then select the appropriate item: Add to Page, Add to Column, Add to Rows or Add to Data by to place the field in a specific area.

6. Rearrange fields. To do this, click the button on field name in one of the areas, and then select one of options shown in table 2.

Table 2. Rearrange field options

Option	Description
Move up	Moves the field up one position in the area.
Move down	Moves the field down position in the area.
Move to Column	Moves the field to column area.
Move to Row	Moves the field to row area.
Move to Data by	Moves the field to the data area and selects the appropriate formula that is used to calculate the data.

Field Option	1. Displays the Field Option window for page, column, or row area. 2. Displays the Data Field Option window for the data area.

Automatically apply changes.

1. The changes are automatically applied to the DataPilot table by default.
2. You can also clear the Automatically Update check box at the bottom of the panel. Then the changes can be applied by manually clicking Update.

Optional: Remove the fields from an area by doing one of the following:

1. Click the button on one of the fields or right-click one of the fields, and select Remove from Areas.
2. Drag the field from the area to the field list.

2.1.3 Setting the data field option for the data area

You can select the function and the calculation type that you want to apply to the data field by setting the data field option:

1. Create a DataPilot table.
2. Click the button on the field name or right-click the field name in the data area, and then select Field Option. The Data Field Option window displays.
3. Click one of the tabs and select the appropriate option (see table 3); click OK.

Table 3. Data field options

Option	Description
Function	Select the function that you want to apply to the data field.
Display Value	Display the data value as: Select the calculation type that you want to use on the base item in the data area. Base field: Select the field from which a value is used as a base for the calculation. Base item: Select a value from the base field to be used as a base for the calculation.

2.1.4 Setting the field option for page, column, or row areas

The Field Option is for areas Page, Column, and Row. You can calculate the subtotals and sort the value by setting the field option. The current field name is displayed in the title of the window:

1. Create a DataPilot table.
2. On the DataPilot window, click the button on the field name or right-click the field name in one of the areas: Page, Column, or Row. Select Field Option; the Field Option window displays. (**Optional:** Put the cursor in the DataPilot table, right-click Page, Column, or Row, and select Field Option.)
3. Click one of the tabs and select the appropriate subtotals option (see table 4).

Table 4. Subtotals options

Option	Description
Never	Does not calculate subtotals.
Automatically	Automatically calculates subtotals.

Manually	Select this option, and then click the functions to calculate in the list.
Using Function	Click the functions to calculate subtotals. This option is only available when the Manually option is selected.
Show items without data	When the option is selected, it includes the empty columns and rows in the result table.

4. Click one of the tabs and select the appropriate sort option (see table 5). Click OK.

Table 5. Sort options

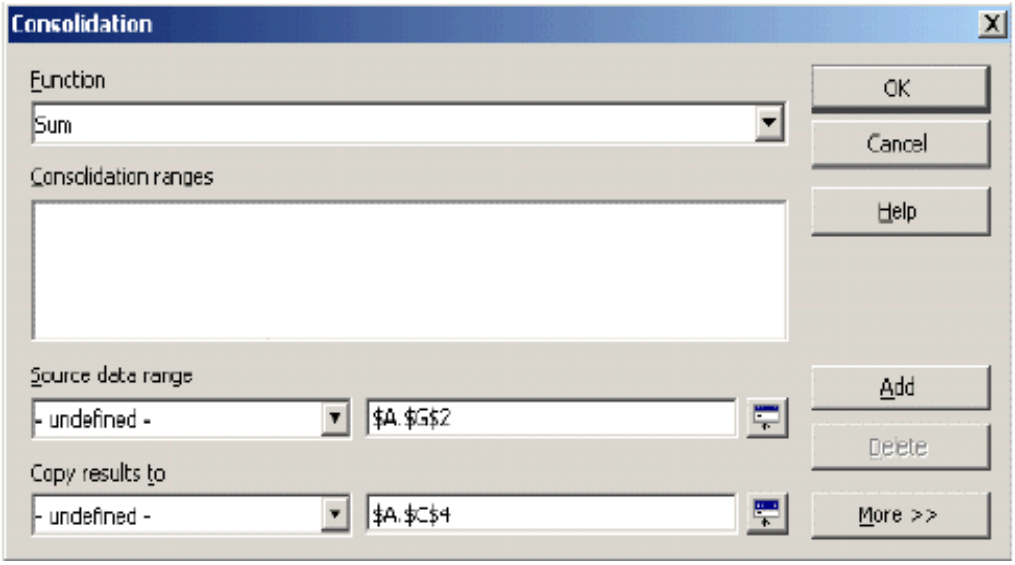
Option	Description
Manually (Select items in the table and drag to organize them)	Sorts items manually by dragging them in the table. Use Ctrl+Click to select a single item to drag.
Ascending	Automatically sorts the values from the lowest value to the highest value.
Descending	Automatically sorts the values from the highest value to the lowest value.
Using Field	Select the field to sort.

2.2 Consolidating data

During consolidation, the contents of the cells from several sheets are combined in one place. To do this:

1. Open the document that contains the cell ranges to be consolidated.
2. Choose Manipulate > Consolidation, to open the Consolidation dialog (see figure 3).

Figure 3. Consolidation dialog box



3. From the “Source data range” field, select a source cell range to consolidate with other areas. If the range is not named, click in the field next to the Source data area. A blinking text cursor appears. Type a reference for the first source data range or select the range with the mouse.

4. Click the Add button to insert the selected range in the Consolidation areas field. Select additional ranges and click Add after each selection.

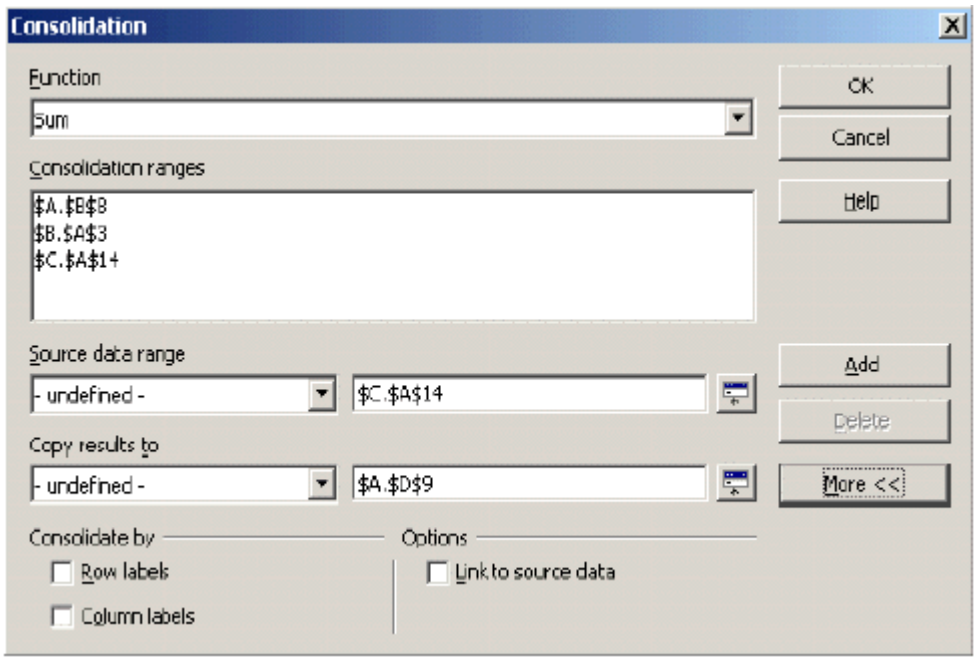
5. Specify where you want to display the result by selecting a target range from the “Copy results to” field.

   If the target range is not named, click in the field next to Copy results to and enter the reference of the target range. Alternatively, you can select the range using the mouse or position the cursor in the top-left cell of the target range.

6. Select a function from the Function drop-down list. The function specifies how the values of the consolidation ranges are linked. The "Sum" function is the default setting. Click OK to consolidate the ranges.

7. If you prefer to retain links to the source ranges instead of copies, or if you want to consolidate ranges in which the order of rows or columns varies, click the More button in the Consolidation dialog (see figure 4).

Figure 4. Consolidation ranges



8. Select “Link to source data” to insert the formulas that generate the results in the target range, rather than the actual results. If you link the data, any values modified in the source range are automatically updated in the target range.

   The corresponding cell references in the target range are inserted in consecutive rows,

which are automatically ordered and then hidden from view. Only the final result, based on the selected function, is displayed.

9. Under “Consolidate by” select either Row labels or Column labels, if the cells of the source data range are not to be consolidated corresponding to the identical position of the cell in the range, but instead according to a matching row label or column label.

   To consolidate by row labels or column labels, the label must be contained in the selected source ranges.

NOTE: The text in the labels must be identical, so that rows or columns can be accurately matched. If the row or column label does not match any that exist in the target range, it will be appended as a new row or column.

The data from the consolidation ranges and target range will be saved when you save the document. If you later open a document in which consolidation has been defined, this data will again be available.

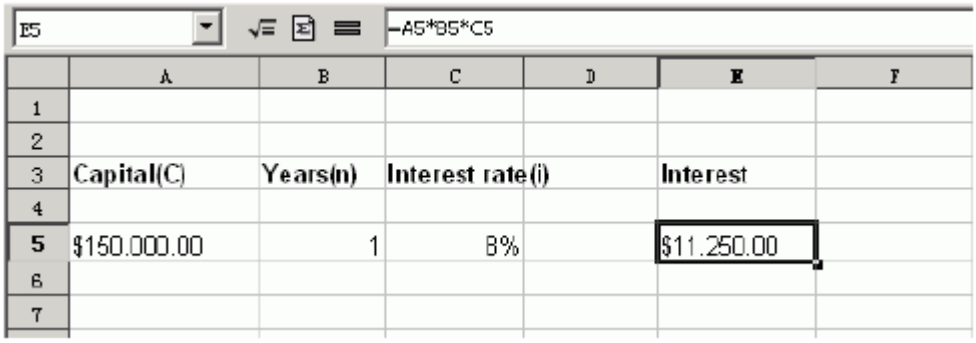
2.3 Applying Solve Equations

With the help of Solve Equations you can calculate a value that, as part of a formula, leads to the result you specify for the formula. You thus define the formula with several fixed values, one variable value, and the result of the formula.

Solve Equations is best illustrated by means of an example:

To calculate annual interest, create a table (see figure 5) with the values for the capital (C), number of years (n), and interest rate (i). The formula is: I = C * n* i / 100

Figure 5. Solve Equations example



Note that you can also enter the formula using the variable names as entered in the headings cells: = 'Capital (C)' * 'Years (n)' * 'Interest rate (i)' / 100

Also, each name must be spelled in the formula exactly as it is seen in the cell, and the name must be enclosed in single quotes.

In this example, the investment capital of $150,000 and an interest rate of 8% is calculated to yield an annual interest income of $11,250. (The cells are formatted after calculation.)

2.3.1 Starting Solve Equations

To do this, assuming we're still using figure 5 as an example:

1. Place the cursor in cell E5 in figure 5 and select Tools > Solve Equations. The Solve Equations dialog appears, and the correct cell is already entered in the field Formula Cell.

2. Place the cursor in the field Variable Cell. In the sheet, click in the cell (A5) that contains the value to be changed.

3. Enter the expected result of the formula in the Target Value text box. In this example, the value is 15,000. Click OK .

4. A dialog box appears informing you that the Solve Equations was successful. The result is indicated for you to use, if you want to. Click Yes to enter the result in cell A5.

2.4 Using What-ifs

What-ifs are important aids for making numbers that are dependent on each other, and their resulting calculations, visible. What-if calculations let you create a list of values from which to select for a given cell or group of cells.

The contents of these cells change when you select different items from the list. If you modify specific basic requirements in the table, you will see the new result. You can give a name to the What-if created in this manner and compare it with another.

2.4.1 Creating your own What-If

To create a What-If, you must select all the cells that provide the data for the What-if. You can create an additional What-if called "High Dollar Rate", as follows:

1. Select the cells that are already highlighted by the frame of the existing scenario. You can also select multiple cells distributed across the table. To select multiple cells, hold down the CTRL key as you click each cell. Select a range, for example, from F18:F23.

2. Choose Tools > What If; the Create What If dialog box appears.

3. Enter "High Dollar Rate" as the name of the scenario and leave the other fields with their default values. Close the dialog with OK. Your new scenario is automatically activated.

4. Change the values in the frames to the values you want to apply to your new scenario. In this case, modify the dollar rate and all other values that you believe could be affected by it (for example, economic growth and sales prices). The effects of your operating figures are then displayed.

3 Macros and programming

This section provides the fundamentals for working with macros in Lotus Symphony Basic.

3.1 Basics

Lotus Symphony Basic code is based on subroutines and functions that are specified between sub...end sub and function...end function sections. Each sub or function can call other subs and functions. If you are careful when writing generic code for a sub or function, you can probably re-use it in other programs.

3.1.1 What is a sub?

Sub is the short form of subroutine, which is used to handle a certain task within a program. Subs are used to split a task into individual procedures. Splitting a program into procedures and sub-procedures enhances readability and reduces the error probability. A sub possibly takes some arguments as parameters, but it does not return any values back to the calling sub or function; for example:

```
DoSomethingWithTheValues(MyFirstValue,MySecondValue)
```

3.1.2 What is a function?

A function is essentially a sub that returns a value. You may use a function at the right side of a variable declaration, or at other places where you normally use values; for example:

```
MySecondValue = myFunction(MyFirstValue)
```

3.1.3 Global and local variables

Global variables are valid for all subs and functions inside a module. They are declared at the beginning of a module before the first sub or function starts. Variables that you declare within a sub or function are valid only inside that sub or function. These variables override both global variables with the same name and local variables with the same name coming from superordinate subs or functions.

3.1.4 Structuring

After separating your program into procedures and functions (subs and functions), you can save these procedures and functions as files for reuse in other projects.

Lotus Symphony Basic supports Modules and Libraries. Subs and functions are always contained in modules, which can be defined to be global or part of a document. Multiple modules can be combined to a library. Also, you can copy or move subs, functions, modules, and libraries from one file to another by using the Macro dialog.

3.2 Syntax

This section describes some of the syntax elements of Lotus Symphony Basic.

3.2.1 Using variables

The following describes the basic use of variables in Lotus Symphony Basic.

3.2.1.1 Naming conventions for variable identifiers

A variable name can consist of a maximum of 255 characters. The first character of a variable name must be a letter A-Z or a-z. Numbers can also be used in a variable name, but punctuation symbols and special characters are not permitted, with exception of the underscore character ("_").

In Lotus Symphony Basic, variable identifiers are not case sensitive. Variable names may contain spaces but must be enclosed in square brackets if they do.

Figure 6 shows some examples of variable identifiers.

Figure 6. Example variable identifiers

MyNumber=5	Correct
MyNumber5=15	Correct
MyNumber_5=20	Correct
My Number=20	Not valid, variable with space must be enclosed in square brackets
[My Number]=12	Correct
Dij`Vu=25	Not valid, special characters are not allowed
5MyNumber=12	Not valid, variable may not begin with a number
Number,Mine=12	Not valid, punctuation marks are not allowed

3.2.1.2 Declaring variables

In Lotus Symphony Basic you don't need to declare variables explicitly; a variable declaration can be performed with the Dim statement. You can declare more than one variable at a time by separating the names with a comma. Also, to define the variable type, use either a type-declaration sign after the name, or the appropriate key word.

Figure 7 shows some examples of variable declarations.

Figure 7. Example variable declarations

DIM a$	Declares the variable "a" as a String
DIM a As String	Declares the variable "a" as a String
DIM a$, b As Integer	Declares one variable as a String and one as an Integer

IMPORTANT: When declaring variables, you must use the type-declaration character each time, even if it was used in the declaration instead of a keyword. Thus the following statements are invalid:

DIM a$	Declares "a" as a String
a="TestString"	Type-declaration missing: "a$="

NOTE: Once you have declared a variable as a certain type, you cannot declare the variable under the same name again as a different type!

Forcing variable declarations. To force declaration of variables, use the following command: OPTION EXPLICIT.

The Option Explicit statement must be the first line in the module, before the first SUB. Generally, only arrays need to be declared explicitly. All other variables are declared

according to the type-declaration character or, if omitted, as the default type Single.

3.2.1.3 Variable types

Lotus Symphony Basic supports four variable classes:

- Numeric variables, which can contain number values. Some variables are used to store large or small numbers, and others are used for floating-point or fractional numbers.
- String variables, which contain character strings.
- Boolean variables, which contain either the TRUE or the FALSE value.
- Object variables, which can store objects of various types, like tables and documents within a document.

Integer variables. Integer variables range from -32768 to 32767. f you assign a floating-point value to an integer variable, the decimal places are rounded to the next integer. Integer variables are rapidly calculated in procedures and are suitable for counter variables in loops. An integer variable only requires two bytes of memory, and "%" is the type-declaration character:

```
Dim Variable%
Dim Variable As Integer
```

Long integer variables. Long integer variables range from -2147483648 to 2147483647. If you assign a floating-point value to a long integer variable, the decimal places are rounded to the next integer. Long integer variables are rapidly calculated in procedures and are suitable for counter variables in loops for large values. A long integer variable requires four bytes of memory, and "&" is the type-declaration character:

```
Dim Variable&
Dim Variable as Long
```

Single variables. Single variables can take positive or negative values ranging from 3.402823 x 10E38 to 1.401298 x 10E-45. Single variables are floating-point variables, in which the decimal precision decreases as the non-decimal part of the number increases.

Single variables are suitable for mathematical calculations of average precision. Calculations require more time than for Integer variables but are faster than calculations with Double variables. A Single variable requires 4 bytes of memory, and the type-declaration character is "!":

```
Dim Variable!
Dim Variable as Single
```

Double variables. Double variables can take positive or negative values ranging from 1.79769313486232 x 10E308 to 4.94065645841247 x 10E-324. Double variables are floating-point variables, in which the decimal precision decreases as the non-decimal part of the number increases.

Double variables are suitable for precise calculations, and these calculations require more time than for Single variables. A Double variable requires 8 bytes of memory, and the type-declaration character is "#":

```
Dim Variable#
Dim Variable As Double
```

Currency variables. Currency variables are internally stored as 64-bit numbers (8 bytes) and displayed as a fixed-decimal number with 15 non-decimal and 4 decimal places. The values range from -922337203685477.5808 to +922337203685477.5807. Currency variables are used to calculate currency values with a high precision, and the type-declaration character is "@":

```
Dim Variable@
Dim Variable As Currency
```

String variables. String variables can hold character strings with up to 65,535 characters. Each character is stored as the corresponding Unicode value. String variables are suitable for word processing within programs and for temporary storage of any non-printable character up to a maximum length of 64 KB. The memory required for storing string variables depends on the number of characters in the variable. The type-declaration character is "$".

```
Dim Variable$
Dim Variable As String
```

Boolean variables. Boolean variables store only one of two values: TRUE or FALSE. Boolean variables are used to store binary values, like the result of a comparison, and are represented internally by a 2-byte integer value. Any value assigned to a Boolean is converted to "False" if the value is not exactly equal to "-1". Boolean variables can only be declared by the key words True or False:

```
Dim Variable As Boolean
```

Date variables. Date variables can contain only dates and time values stored in an internal format. Values assigned to Date variables with Dateserial, Datevalue, Timeserial, or Timevalue are automatically converted to the internal format.

You can convert Date variables to normal numbers by using the Day, Month, Year or the Hour, Minute, Second function. The internal format allows a comparison of date/time values by calculating the difference between two numbers. These variables can only be declared with the key word Date:

```
Dim Variable As Date
```

3.2.1.4 Initial variable values

As soon as the variable has been declared, it is automatically set to the "Null" value. Note the following conventions:

- Numeric variables are automatically assigned the value "0" as soon as they are declared.
- Date variables are assigned the value 0 internally, equivalent to converting the value to "0" with the Day, Month, Year or the Hour, Minute, Second function.
- String variables are assigned an empty-string ("") when they are declared.

3.2.1.5 Arrays

Lotus Symphony Basic knows one- or multi-dimensional arrays, defined by a specified variable type. Arrays are suitable for editing lists and tables in programs. Individual elements of an array can be addressed through a numeric index.

Arrays must be declared with the Dim statement, and there are several ways to define the index range of an array (see figure 8).

Figure 8. Ways of defining index range of array

DIM text$(20)	21 elements numbered from 0 to 20
DIM text$(5,4)	30 elements (a matrix of 6 x 5 elements)
DIM text$(5 to 25)	21 elements numbered from 5 to 25
DIM text$(-15 to 5)	21 elements (including 0), numbered from -15 to 5

The index range can include positive as well as negative numbers. The maximum number of elements that can be addressed through an index is 16368.

3.2.1.6 Constants

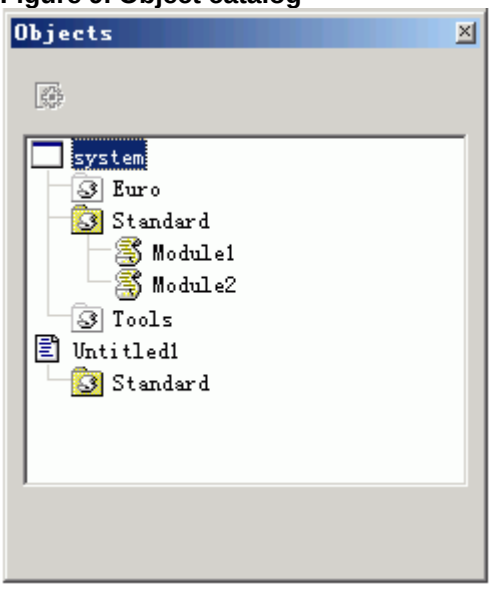
Constants have a fixed value. They are only defined once in the program and cannot be redefined later:

```
CONST ConstName=Expression
```

3.2.2 Using objects

The object catalog provides an overview of all modules and dialogs you've created in Lotus Symphony. Click the Object Catalog icon in the Macro toolbar to display the object catalog dialog box (see figure 9).

Figure 9. Object catalog



The dialog box shows a list of all existing objects in a hierarchical representation. Double-clicking a list entry opens its subordinate objects. To display a certain module in the Editor or to position the cursor in a selected SUB or FUNCTION, select the corresponding entry and click the Show icon.

3.2.3 Using procedures and functions

Here we describe the use of procedures and functions in Lotus Symphony Basic. Procedures (SUBS) and functions (FUNCTIONS) help you maintain a structured overview by separating a program into logical pieces.

One benefit of procedures and functions is that, once you have developed a program code containing task components, you can use this code in other project.

3.2.3.1 Passing variables to procedures (SUB) and functions (FUNCTION)

Variables can be passed to both procedures and functions. The SUB or FUNCTION must be declared to expect parameters:

```
SUB SubName( Parameter1 As Type, Parameter2 As Type,... )

Program code

END SUB
```

The SUB is called using the following syntax:

```
SubName(Value1, Value2,...)
```

The parameters passed to a SUB must fit to those specified in the SUB declaration.

The same process applies to FUNCTIONS. In addition, functions always return a function result. he result of a function is defined by assigning the return value to the function name:

```
FUNCTION FunctionName(Parameter1 As Type, Parameter2 As Type,...) As Type

Program code

FunctionName=Result

End Function
```

The FUNCTION is called using the following syntax:

```
Variable=FunctionName(Parameter1, Parameter2,...)
```

3.2.3.2 Passing variables by value or reference

Parameters can be passed to a SUB or a FUNCTION either by reference or by value; however, unless otherwise specified, a parameter is always passed by reference. That means that a SUB or a FUNCTION gets the parameter and can read and modify its value.

If you want to pass a parameter by value, insert the key word "ByVal" in front of the parameter when you declare a SUB or FUNCTION; for example:

Sub ChangeValue(ByVal Parameter As Integer)

In this case, the original content of the parameter will not be modified by the FUNCTION since it gets only the value and not the parameter itself.

NOTE: When you create a new module, Lotus Symphony Basic automatically inserts a SUB called "Main". This default name has nothing to do with the order or the starting point of a Lotus Symphony Basic project. You can also safely rename this SUB.

3.2.3.3 Scope of variables

A variable defined within a SUB or FUNCTION remains valid only until the procedure is exited. This is known as a "local" variable. However, in many cases, you need a variable to be valid in all procedures, in every module of all libraries, or after a SUB or FUNCTION is exited. This property is controlled in a variable declaration through key words in the Dim statement.

Declaring variables outside a SUB or FUNCTION:

DIM GLOBAL VarName As TYPENAME

The variable is valid as long as the Lotus Symphony session lasts.

DIM PUBLIC VarName As TYPENAME

The variable is valid in all modules.

DIM PRIVATE VarName As TYPENAME

The variable is valid only in this module.

DIM VarName As TYPENAME

The variable is valid only in this module.

Saving variable content after exiting a SUB or FUNCTION:

DIM STATIC VarName As TYPENAME

The variable retains its value until the next time the FUNCTION or SUB is entered. The declaration must exist inside a SUB or a FUNCTION.

3.2.3.4 Specifying the return value type of a FUNCTION

As with variables, to define the type of the function's return value, include a type-declaration character after the function name, or the type indicated by "As" and the corresponding key word at the end of the parameter list; for example:

Function WordCount (WordText as String) as Integer

3.3 Lotus Symphony Basic IDE

This section describes the structure of the Basic Integrated Development Environment (IDE).

3.3.1 Watch window

The Watch window lets you observe the value of variables during the execution of a program. To do this, you define the variable in the Watch text box, and then click Add Watch to add the variable to the list box and display its values.

3.3.2 Watch

Enter the name of the variable whose value is to be monitored:

Remove Watch. Removes the selected variable from the list of watched variables.

Editing the Value of a Watched Variable. Displays the list of watched variables. Click twice with a short pause in between on an entry to edit its value. The new value will be taken as the variable's value for the program.

Call Stack window (Calls). The Call Stack lets you monitor the sequence of procedures and functions during the execution of a program. The procedures are functions that are displayed bottom to top, with the most recent function or procedure call at the top of the list.

3.3.3 Manage Breakpoints

Specify the options for breakpoints:

Breakpoints. Enter the line number for a new breakpoint, then click New.

Active. Activates or deactivates the current breakpoint.

Pass Count. Specifies the number of loops to perform before the breakpoint takes effect.

New. Creates a breakpoint on the line number specified.

Delete. Deletes the selected breakpoint.

3.3.4 Keyboard shortcuts in the Basic IDE

Figure 10 shows the keyboard shortcuts you can use in the Basic IDE.

Figure 10. Basic IDE keyboard shortcuts

Action	Keyboard shortcut
Run code starting from the first line, or from the current breakpoint, if the program stopped there before	F5
Stop	Alt+F5
Stop, then start again from the first line	Shift+F5
Add watch for the variable at the cursor	F7
Single step through each statement, starting at the first line or at that statement where the program execution stopped before	F8
Single step as with F8, but a function call is considered to be only one statement	Shift+F8
Set or remove a breakpoint at the current line or all breakpoints in the current selection	F9
Enable/disable the breakpoint at the current line or all breakpoints in the current selection	Shift+F9

Note that a running macro can be aborted with Shift+CTRL+Q, and also from outside the Basic IDE. If you are inside the Basic IDE and the macro halts at a breakpoint, Shift+CTRL +Q stops the execution of the macro, but you can only recognize this after the next F5, F8, or Shift+F8.

3.4 Using macros

Choose the macro that you want to execute when the selected graphic, frame, or OLE object is selected. Depending on the object that is selected, the function is found either on the Macro tab of the Object dialog or in the Assign Macro dialog.

3.4.1 Convert from Visual Basic for Applications (VBA) to Symphony Basic (SB)

VBA macros are unable to run in Lotus Symphony, so they must first be converted and adapted. Macro convertor is an automated converting tool that converts VBA macros of Microsoft Office into SB macros supported in Lotus Symphony.

To open the macro editor of Lotus Symphony, do the following:

1. Choose Tools > Macros > Macro....
2. Click the Edit button to start the Basic editor.

To convert the whole macro file:

1. Basic editor will open VBA files to be converted from library list box in Macro Toolbar.
2. Click the button Convert from VBA to SB; a dialog about conversion statistics information pops up.

3.4.2 Event

This lists the events that are relevant to the macros currently assigned to the selected object. The table in figure 11 describes the macros and the events that can be linked to objects in your document.

Figure 11. Macros and events

Event	Event trigger	OLE object	Graphics	Frame	AutoText	ImageMap area	Hyperlink
Click object	Object is selected.	x	x	x			
Mouse over object	Mouse moves over the object.	x	x	x		x	x
Trigger Hyperlink	Hyperlink assigned to the object is clicked.	x	x	x			x
Mouse leaves object	Mouse moves off of the object.	x	x	x		x	x
Graphics load successful	Graphics are loaded successfully.		x				
Graphics load terminated	Loading of graphics is stopped by the user (for example, when downloading the page).		x				
Graphics load faulty	Graphics not successfully loaded, for example, if a graphic is not found.		x				
Input of alpha characters	Text is entered from the keyboard.			x			
Input of non-alpha characters	Nonprinting characters are entered from the keyboard, for example, tabs and line breaks.			x			
Resize frame	Frame is resized with the mouse.			x			
Move frame	Frame is moved with the mouse.			x			
Before inserting AutoText	Before a text block is inserted.				x		
After inserting AutoText	After a text block is inserted.				x		

3.4.3 Macros

Choose the macro that you want to execute when the selected event occurs. Frames allow you to link events to a function, so that the function can determine if it processes the event or processes Lotus Symphony Documents.

Category. Lists the open Lotus Symphony documents and applications. Click the name of the location where you want to save the macros.

Macro name. Lists the available macros. Click the macro that you want to assign to the selected object.

Assign. Assigns the selected macro to the specified event. The assigned macro's entries are set after the event.

Remove. Removes the macro that is assigned to the selected item.

Macro selection. Selects the macro that you want to assign.

3.5 Event-driven macros

This section describes how to assign Basic programs to program events.

You can automatically execute a macro when a specified software event occurs by assigning the desired macro to the event. The table in figure 12 provides an overview of program events and the point at which an assigned macro is executed.

Figure 12. Program events and when macro is executed

Event	An assigned macro is executed...
Program Start	... after a IBM® Lotus® Symphony™ application is started.
Program End	...before a Lotus Symphony application is terminated.
Create Document	...after a new document is created with File - New or with the New icon.
Open Document	...after a document is opened with File - Open or with the Open icon.
Save Document As	...before a document is saved under a specified name (with File - Save As , or with File - Save or the Save icon, if a document name has not yet been specified).
Document has been saved as	... after a document was saved under a specified name (with File - Save As , or with File - Save or with the Save icon, if a document name has not yet been specified).
Save Document	...before a document is saved with File - Save or the Save icon, provided that a document name has already been specified.
Document has been saved	...after a document is saved with File - Save or the Save icon, provided that a document name has already been specified.
Close Document	...before a document is closed.
Document is being closed	...after a document is closed. Note that the "Save Document" event may also occur when the document is saved before closing.
Activate Document	...after a document is brought to the foreground.
Deactivate Document	...after another document is brought to the foreground.
Print Document	...after the Print dialog is closed, but before the actual print process begins.
Print Mail Merge	...after the Print dialog is closed, but before the actual print process begins. This event occurs for each copy printed.
Change of the page count	...when the page count changes.
Message received	...if a message is received.

3.5.1 Assigning a macro to an event

To assign a macro to an event, follow these steps:

1. Select Tools > Macros > Document Events Configure.
2. Select whether you want the assignment to be globally valid or just valid in the current document by selecting the Lotus Symphony or Document option.
3. Select the event from the Event list.
4. Select the module containing the macro to be assigned to the selected event from the Macros list. The list contains a hierarchical list of modules, libraries, files, and templates.
5. Select the macro to be assigned from the right list; click Assign.
6. Click OK to close the dialog.

3.5.2 Removing the assignment of a macro to an event

To do this, follow these steps:

1. Select Tools > Macros > Document Events Configure.
2. Select whether you want to remove a global assignment or an assignment that is just valid in the current document by selecting the Lotus Symphony or Document option.
3. Select the event that contains the assignment to be removed from the Event list.
4. Click Remove, and click OK to close the dialog.

3.6 Debugging a basic program

Let's now discuss the procedures to debug a basic program in Lotus Symphony Basic.

3.6.1 Breakpoints and single-step execution

You can check each line in your basic program for errors, using single-step execution. Errors are easily traced because you can immediately see the result of each step. A pointer in the breakpoint column of the Editor indicates the current line, and you can set a breakpoint if you want to force the program to be interrupted at a specific position.

Double-clicking in the breakpoint column at the left of the Editor window toggles a breakpoint at the corresponding line, and when the program reaches a breakpoint, the program execution is interrupted.

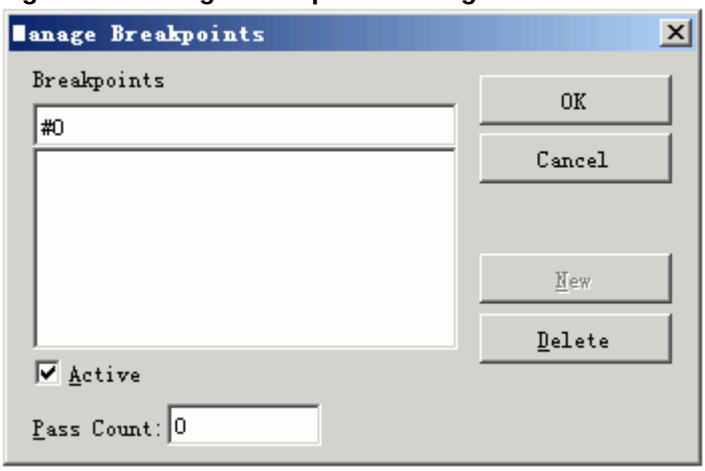
Single-step execution using the Single Step icon causes the program to branch into procedures and functions, while the procedure-step execution using the Procedure Step icon causes the program to skip over procedures and functions as a single step.

3.6.1.1 Properties of a breakpoint

You can access the properties of a breakpoint through its context menu by right-clicking the breakpoint in the breakpoint column. You can activate and deactivate a breakpoint by selecting Active from its context menu. When a breakpoint is deactivated, it does not interrupt the program execution.

Select Properties from the context menu of a breakpoint or select Breakpoints from the context menu of the breakpoint column to call the Manage Breakpoints dialog (see figure 13), in which you can specify other breakpoint options. The list displays all breakpoints with the corresponding line number in the source code.

Figure 13. Manage Breakpoints dialog box



You can activate or deactivate a selected breakpoint by checking or clearing the Active box. The Pass Count specifies the number of times the breakpoint can be passed over before the program is interrupted. If you enter 0 (default setting, as shown in the figure), the program is always interrupted as soon as a breakpoint is encountered.

Click Delete to remove the breakpoint from the program.

3.6.2 Observing the value of variables

You can monitor the values of a variable by adding it to the Watch window. To add a variable to the list of watched variables, type the variable name in the Watch text box and press Enter. The values of variables are displayed only if they are in scope. Variables that are not defined at the current source code location display ("Out of Scope") instead of a value.

You can also include arrays in the Watch window. If you enter the name of an array variable without an index value in the Watch text box, the content of the entire array is displayed.

TIP: If you rest the mouse over a predefined variable in the Editor at runtime, the content of the variable is displayed in a pop-up box.

The Call Stack window provides an overview of the call hierarchy of procedures and functions. Using this window, you can determine which procedures and functions called which other procedures and functions at the current point in the source code.

4 Hyperion Essbase spreadsheet plug-in for Lotus Symphony

This plug-in adds the capability of displaying multi-dimensional data in Lotus Symphony. Users can access and analyze data on the Essbase server by simple mouse-click operations. The plug-in also enables multiple users to access data on the Essbase server simultaneously.

After the plug-in is installed, a special menu is added to Lotus Symphony providing the following functions:

- connect to and disconnect from the Essbase server,
- retrieve data from server into a worksheet,
- drill down and drill up a worksheet to analyze data from multiple viewpoints.

An existing accessible Hyperion Essbase server is required to use the plug-in.

4.1 Software requirements

The required software is as follows:

- Operating System: Microsoft Windows XP
- Language: English
- Lotus Symphony: Lotus Symphony Beta 4 or later version
- Hyperion Java™ client

Hyperion provides a Java client beginning with Hyperion version 7. The client includes three Java JAR files, which must be on your system before you install the Lotus Symphony plug-in:

ess_es_server.jar
ess_japi.jar
log4j-1.2.8.jar

You can install any Hyperion product that contains the Java client JAR files, such as Hyperion Essbase - System 9 Release 9.3.1 Client, available from the Oracle Essbase Downloads Web site.

4.2 Installing the plug-in

You should already have Lotus Symphony installed on your computer. Follow the steps below to install a plug-in into Lotus Symphony:

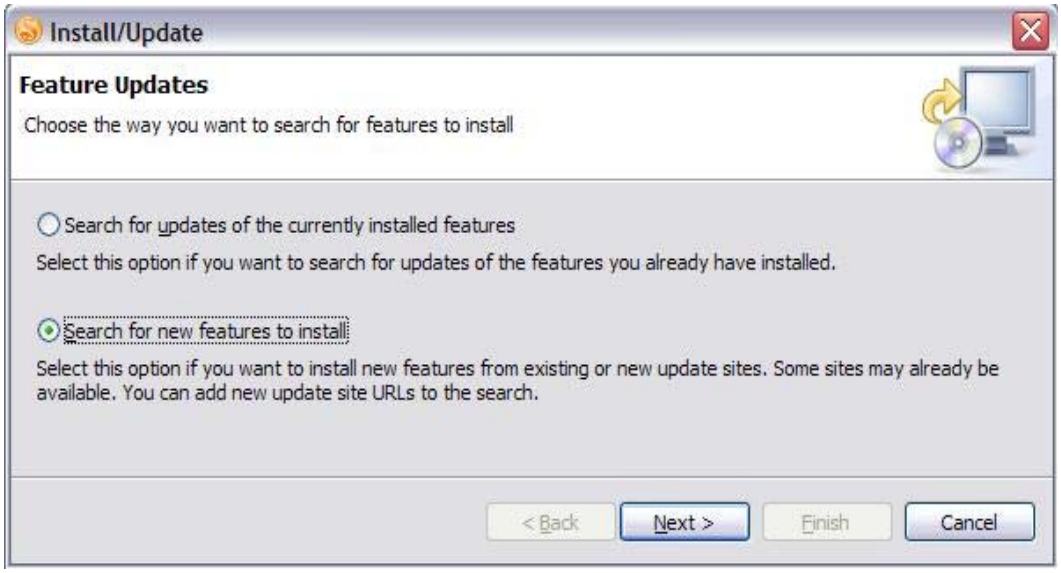
1. Copy the Hyperion Java client JAR files to the Lotus Symphony plug-in installation directory. To locate Hyperion Java client JAR files:

   - After you install the Hyperion client, you can get Java client JAR files from the installation directory. For example, if you installed “Hyperion Essbase - System 9 Release 9.3.1 Client ” to C:\Hyperion\AnalyticServices, the files ess_es_server.jar and ess_japi.jar are in the directory C:\Hyperion\AnalyticServices\JavaAPI\lib, and the file log4j-1.2.8.jar is in the directory C:\Hyperion\AnalyticServices\JavaAPI\external \css.

   - Copy the JAR files to the Lotus Symphony plug-in installation directory. If you install Lotus Symphony at C:\IBM\Lotus\Symphony, then the plug-in installation directory is C:\IBM\Lotus\Symphony\framework\shared\eclipse\plugins. Copy the three JAR files to this directory.

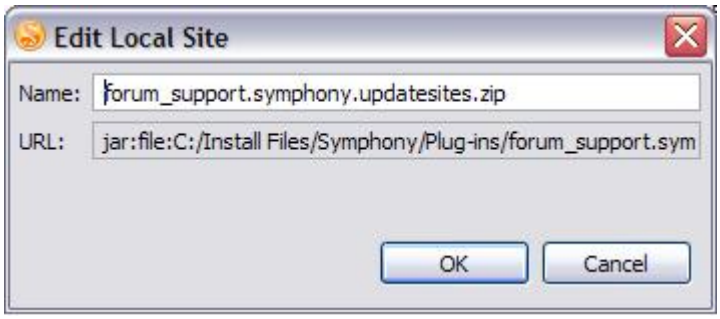
2. Start Lotus Symphony.
3. On the menu, select File > Application > Install; the Install/Update Wizard appears (see figure 14).

Figure 14. Install/Update wizard



4. Select the option “Search for new features to install” and click Next.
5. On the Applications Locations screen, click the button to Add Zip/Jar Location.
6. Locate the plug-in file that you downloaded; for example, **forum_support.symphony.updatesites.zip** is the name of the plug-in .zip file for the Forum Support Plug-in for Lotus Symphony.
7. Select the file and click Open.
8. When prompted with the Edit Local Site dialog, verify the name of the file you selected and click OK (see figure 15).

Figure 15. Edit Local Site dialog



9. In the Application Locations screen you should now see the file Forum Support plug-in listed in the Location List; click Finish.

10. On the Search Results screen, click in the box next to the Forum Support plug-in to select it (see figure 16).

Figure 16. Search Results screen

11. Click Finish to complete the installation process.

When the installation completes you will be prompted to restart Lotus Symphony, which you must do before the plug-in will be available.

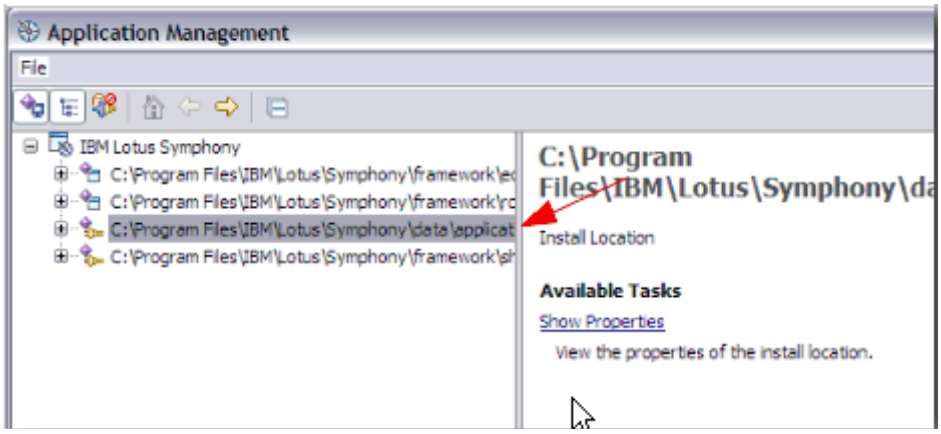
When Lotus Symphony restarts, you'll notice that a button for the plug-in has been added to your toolbar. For other plug-ins, you may see a new menu choice called Add-ins, rather than a toolbar button.

4.3 Uninstalling plug-ins

To uninstall plug-ins, follow these steps:

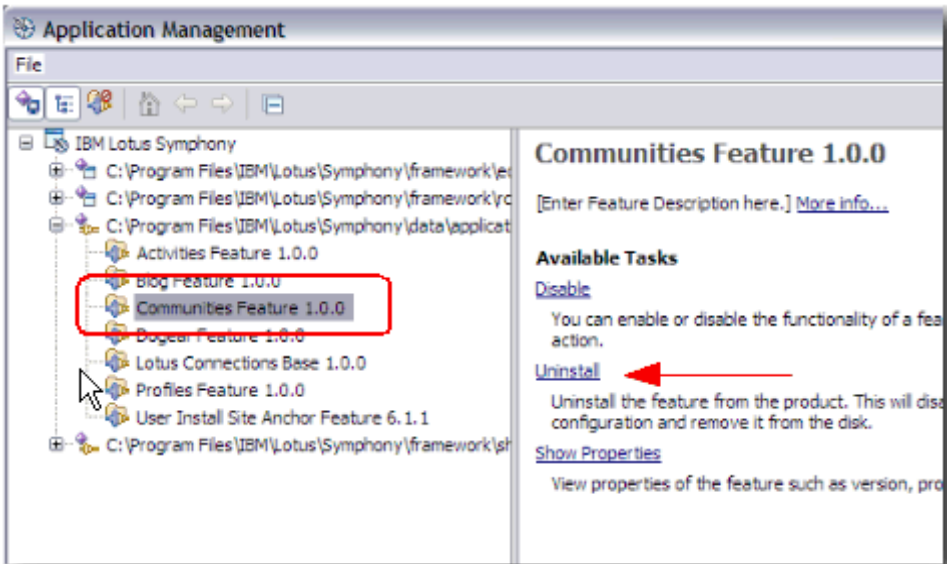
1. Select File > Application > Application Management. A window displays showing the Application Management menu (see figure 17).

Figure 17. Application Management menu



2. From the list on the left-hand side, expand the fourth item, called C:\Program Files\IBM \Lotus\Symphony\framework\shared\eclipse. Click to highlight the plug-in you want to remove (see figure 18).

Figure 18. Uninstalling plug-in



3. Notice that, on the right-hand side of the screen, you are given the option to Disable, Uninstall, or Show Properties of the highlighted plug-in. Click Uninstall. In this example, Communities Feature 1.0.0 is chosen to be uninstalled.

4. A pop-up window will prompt you whether you want to uninstall. Click OK.

5. Click Yes to restart Lotus Symphony. The uninstall process is complete.

NOTE: In some rare cases a plug-in may need to be disabled before being uninstalled. To do this, follow the beginning steps and instead of clicking Uninstall, click Disable. You will then be prompted if you want to disable the plug-in and then asked to restart Lotus Symphony.

5 Conclusion

Hopefully this white paper has enabled you to adapt some of the key financial tools you currently use to Lotus Symphony Spreadsheets. It's not meant to be a comprehensive migration guide; however, it should help you progress down the learning curve to becoming productive and comfortable with Lotus Symphony Spreadsheets.

6 Resources

- IBM Lotus Symphony 1.2 information center:
  http://publib.boulder.ibm.com/infocenter/symphony/v1r1/index.jsp

- Lotus Symphony Discussion forum:
  http://symphony.lotus.com/software/lotus/symphony/ForumnsHome.nsf/home

- Lotus Symphony developerWorks product page:
  http://www.ibm.com/developerworks/lotus/products/symphony/?S_TACT=105AGX13&S_CMP=LP

- Lotus Symphony Help page:
  http://symphony.lotus.com/software/lotus/symphony/help.nsf/home, especially:

  - Moving from Microsoft Excel 2003 to IBM Lotus Symphony Spreadsheets Toolbar Reference Card
  - Moving from Microsoft Excel 2003 to IBM Lotus Symphony Spreadsheets keyboard shortcut comparison table

7 About the author

Ed Ward is a Software Engineer for IBM in Atlanta, GA, and is currently a member of the Calendar and Scheduling Support team for Lotus Notes and Domino. He started his professional career less than 2 years ago--with IBM—during which time he has earned the titles of IBM Advanced Certified System Administrator, Security Professional, and Application Developer for Lotus Notes and Domino 7. Ed holds dual bachelor's degrees in Computer Science and Electrical Engineering. You can reach him at warde@us.ibm.com.