MSC-LIMS™ *Insights*

The source for news and tips of interest to users of MSC-LIMS,
an affordable laboratory information management system for small labs.

Issue No. 31 February 2019

Welcome

Welcome to **MSC-LIMS *Insights***.

This newsletter will help current MSC-LIMS users get the most out of their software, and will complement the product literature and demo that prospective users can find on our web site at www.msc-lims.com.

Join our mailing list for more information. Sign up at www.msc-lims.com/lims/maillist.html.

This newsletter is for and about MSC-LIMS users. We welcome your comments, and your suggestions for topics you would like to see addressed in upcoming issues. Please send your thoughts to newsletter@msc-lims.com.

MSC-LIMS 5.0 for SQL Server Released

We are pleased to announce that MSC-LIMS version 5.0 has recently been released. Migrating from its previous Microsoft Access back end database, version 5.0's foundation is now a Microsoft SQL Server back end database.

Labs of all sizes will benefit from the improved reliability, scalability, and security offered by the SQL Server database. Information technology departments will welcome MSC-LIMS' migration to SQL Server since many already employ and maintain this widely used database platform.

Users will appreciate the convenience of SQL Server's integrated Windows authentication, eliminating a separate LIMS login when starting the system. A user's Windows workstation login credentials now provide secured access to MSC-LIMS.

Finally, all example Excel templates used to import and export LIMS data have been upgraded from Excel 97-2003 xlt format to the newer macro-enabled xltm format.

Prices start at $750 per workstation per year for an MSC-LIMS 5.0 Annual Subscription License. See the current price list for details.

If you have been looking for an affordable LIMS with a SQL Server back end database, consider MSC-LIMS version 5.0.

In this Issue

From the Developer

Have you ever received Excel's "Application-defined or object-defined" error when using an Excel template with MSC-LIMS? I have seen this error countless times and I have received many technical support emails and phone calls from users asking what causes the problem.

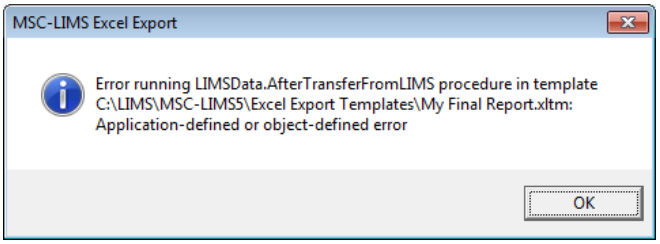
Unfortunately, this message appears to be an Excel catchall for many different errors. If Excel provided a more descriptive error message we might all learn the source of the error more easily. In this issue's "How to Debug an Excel Template Macro" you can follow the same procedures we use to troubleshoot a template's macro and uncover the source of the error.

Finally, if you have been waiting for a SQL Server version of MSC-LIMS, version 5.0 for SQL Server is here! Read the announcement above for more information.

Rick Collard is the founder of Mountain States Consulting, LLC and the principal developer of the MSC-LIMS software. You can reach Rick by email at rcollard@msc-lims.com.

How to Debug an Excel Template Macro

MSC-LIMS' integrated Excel interface is a powerful tool. Exporting LIMS data to your own Excel export template is a dynamic method to control the presentation of your data. However, when something goes wrong, troubleshooting Excel's often cryptic error messages can be a challenge. For example, Excel's all too common "Application-defined or object-defined" error shown below provides little or no insight into the cause of the problem.



This article provides a simple and systematic method you can use to troubleshoot an Excel template's macro to locate the source of the error. Excel export template macros are typically more complex than import templates so we will focus on export templates here.

An Excel macro is a named Visual Basic for Applications (VBA) procedure comprised of any number of VBA statements. There are many Excel VBA statements available including those to execute any task you perform manually such as copying cells, deleting columns, and entering formulas. When you export a LIMS report to an Excel template, MSC-LIMS performs the following procedure:

1. Start Excel if it is not already running.
2. Direct Excel to create a new workbook from the selected template.
3. Run the new workbook's BeforeTransferFromLIMS macro.
4. Copy the LIMS report's underlying data to the new workbook's LIMSData worksheet.
5. Run the new workbook's AfterTransferFromLIMS macro.

An export template's BeforeTransferFromLIMS macro rarely performs any tasks. However, a template's AfterTransferFromLIMS macro normally includes many VBA statements to produce a final report from the exported LIMS data. Therefore, it is this macro that is the culprit in most errors.

Debugging a macro refers to the process of locating and removing the *bug* (i.e. error) within the macro. When a macro runs normally it runs at the speed at which the machine can execute its statements. At this speed it is nearly impossible for us to watch the macro's progress as it executes each statement. When debugging the macro, we control when each VBA

(Continued on next page)

(Continued from previous page)

statement is executed so we can watch the result of each statement and the macro's progress, and hopefully observe the specific VBA statement causing the error.

First, to debug a template's AfterTransferFromLIMS macro, we must prevent MSC-LIMS from automatically executing the macro so that we can start the macro and control the execution of its statements. How you do this depends on your MSC-LIMS version.

With MSC-LIMS 5.x, after selecting the template simply hold down the Shift key while clicking the OK button in the Excel Data Transfer popup to prevent running the template's macros. With MSC-LIMS 4.x we must rename the existing AfterTransferFromLIMS macro and add an empty macro. To do so, first make a copy of your template, then open the copy, use Alt+F11 to open the VBA editor, then double-click LIMSData in the project explorer's list of Excel objects to view its macros. In the code pane on the right, find and copy the `Public Sub AfterTransferFromLIMS()` statement then place an 'x' just before the 'A' to rename the macro xAfterTransferFromLIMS. Now insert a new line just above this statement and paste in the original copied line. Hit Enter and the editor will automatically insert an `End Sub` statement to complete our new empty AfterTransferFromLIMS macro. The screen excerpt below shows the empty and renamed macros.

```
Public Sub AfterTransferFromLIMS()

End Sub
```

```
Public Sub xAfterTransferFromLIMS()

    ' NOTE: Your template must set references
    ' to the Microsoft Access 10.0 Object Library and
    ' to the Microsoft DAO 3.6 Object Library using
    ' Tools | References in the VBA editor.

    Dim objAccess As Access.Application
    Dim db As Database, rs As Recordset, fld As Field
    Dim rng As Range
    Dim i As Integer, nCustomerID As Long
    Dim nRows As Long, nDataRows As Long
```

Second, most of the example export templates included with MSC-LIMS use VBA statements at the beginning of the AfterTransferFromLIMS macro to disable screen updating and prevent user intervention. This improves performance since Excel does not update the screen while the VBA statements are executed. However, while debugging we must see screen updates and interact with Excel so these macro statements must be disabled. If your template has a Settings worksheet with a 'Prevent Screen Updating' option, set it to 'No' now to ensure screen updating is enabled.

	A	B	C
1	*LIMSAccessVersion:*	14	
2	*RemoveInfrastructure:*	Yes	
3	*PreventScreenUpdating:*	No	
4	Exclude "Internal Data" Analyses:	Yes	
5			

If your template does not have a 'Prevent Screen Updating' option on a Settings worksheet, locate the macro code shown below and insert a single-quote at the start of each of the three lines that begin with a period. This will turn the entire lines green indicating they are comments and will no longer execute. Exit the VBA editor then close and save your template.

```
' Prevent screen updating.
With Application
    '.Cursor = xlWait              ' show hourglass cursor
    '.Interactive = False          ' no interruption
    '.ScreenUpdating = False       ' turn it off
End With
```

Third, to debug your template begin by exporting the LIMS report to the edited template with the empty AfterTransferFromLIMS macro or use the Shift bypass key with MSC-LIMS 5.x. In the new Excel workbook created from the template the report will be incomplete since the macro has not yet run. Verify that the LIMS has copied data to the LIMSData worksheet. Use Alt+F11 to open the VBA editor and select the LIMSData sheet to display its macros.

Fourth, before running the macro, set a *breakpoint* in either the xAfterTransferFromLIMS macro for version 4.x or the AfterTransferFromLIMS macro for version 5.x. A breakpoint marks a VBA statement where the VBA editor will halt execution while running the macro. To set a breakpoint click within the grey left margin adjacent to the statement. The editor will place a red dot in the margin and set the entire statement red to identify the breakpoint. Note that a single VBA statement may span multiple lines. For your first debugging session it is best to set the breakpoint near the top of the macro. The excerpt below shows a good place to start with code that should exist in most export templates.

(Continued on next page)

(Continued from previous page)

```
    ' Create named ranges for each column transferred from the LIMS
    ' then save the SampleID and CustomerID for the first report sample.
    With Worksheets("LIMSData")
        .Select
        .UsedRange.Select
        Selection.CreateNames Top:=True, Left:=False, Bottom:=False, Right:=False
        .Range("A1").Select
        sSampleID = .Range("SampleID").Cells(1, 1)
        nCustomerID = .Range("CustomerID").Cells(1, 1)
    End With
```

Finally, leave the VBA editor open, return to Excel, use Alt+F8, select either the xAfterTransferFromLIMS macro for version 4.x or the AfterTransferFromLIMS macro for version 5.x, then click Run. Excel will run the macro and the VBA editor will halt execution at the first breakpoint, highlighting the next statement to be executed in yellow.

```
    ' Create named ranges for each column transferred from the LIMS
    ' then save the SampleID and CustomerID for the first report sample.
    With Worksheets("LIMSData")
        .Select
        .UsedRange.Select
        Selection.CreateNames Top:=True, Left:=False, Bottom:=False, Right:=False
        .Range("A1").Select
        sSampleID = .Range("SampleID").Cells(1, 1)
        nCustomerID = .Range("CustomerID").Cells(1, 1)
    End With
```

Now debugging begins in earnest. Use Shift+F8 to execute the yellow highlighted statement. Continue using Shift+F8 to execute statements. Note that you can switch to Excel to see the results of the statements and view the contents of any worksheet. Using Shift+F8 may be tedious in a long macro but often single-stepping through the macro is necessary to find the statement causing the error.

When you encounter VBA statements that form a loop you may find single-stepping with Shift+F8 too time-consuming. In this case, set another breakpoint after the loop as shown below then use F5 allow the macro to run up to the next breakpoint.

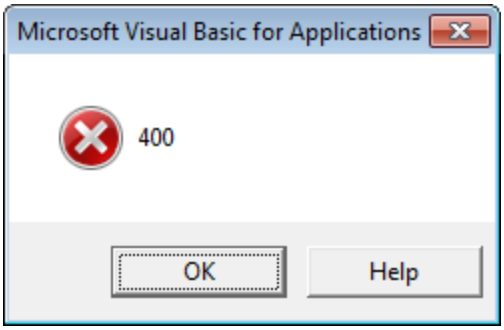
```
        For i = 0 To rs.Fields.Count - 1
            Set fld = rs.Fields(i)
            rng.Offset(0, i).Value = fld.Name    ' show field name
            rng.Offset(0, i).Font.Bold = True    ' make it bold
        Next i

        ' Get the sample's analysis data and create named ranges.
        With Worksheets("AnalysisData")
            .Range("A2").CopyFromRecordset rs
            .Select
```

Continue executing the macro using a combination of single-stepping with Shift+F8, setting breakpoints, and running to the next breakpoint with F5 until you locate the statement causing the error. In our example, the next statement to execute highlighted in the excerpt below caused our error.

```
        ' For each analysis exported, insert a new row below the OneAnalysisResults
        ' then copy the OneAnalysisResults named range vertically for each analysis.
        With Worksheets(1)
            .Select
            Set rng = .Range("OneAnalysisResults")
            rng.Offset(1, 0).Resize(nDataRows, 1).EntireRow.Insert
            nRows = .Range("OneAnalysisResults").Rows.Count
```

The VBA editor displayed the error message below when executing the highlighted statement. Although the error message is not informative and results in the dreaded "Application-defined or object-defined" error, the simplicity of the statement identifies the problem. Here, the OneAnalysisResults named range was inadvertently deleted with its original cells while editing the template. Restoring the named range solved the problem.



Note that when an error message is displayed the offending statement is no longer highlighted so it is important to make a mental note of the current statement when using Shift+F8. It is common to lose track of the current statement when rapidly single-stepping through the macro. In that case you will have to end the current debugging session and begin again but this time you can set your first breakpoint in the macro closer to where the error occurred.

Excel templates and their powerful VBA macros provide an important tool to analyze, report, and deliver LIMS data. However, Excel can be finicky and some of its error messages are not very informative. Use the debugging techniques described here and you can easily find the cause of the error. Consider debugging a working template now so you are prepared when you encounter a real error.

Notes from Technical Support

Emailing Preliminary Results

A user recently asked:

Is it possible for LIMS to automatically send an analyte result directly to an email once it is entered? I've looked in the LIMS manual and the closest thing I've seen is sending a sample summary report by email.

MSC-LIMS Messaging can automatically email results when either a single sample is completed or a batch of samples is completed. However, you can also manually email results prior to a sample's completion with a single button click in the Results by Sample screen. Some sites use this feature to send preliminary results. If you have configured messaging for a customer and you want to send a preliminary report showing the sample's current results, you can click the "Resend completion message" button above the customer name on the Results by Sample screen. If you are using a message style based on an Excel template it is easy to have the template mark the report "Preliminary" when the sample is incomplete.

Creating Protected Excel Workbooks

A user who needed to send Excel-based Certificate of Analysis reports submitted this question:

Is it possible to [automatically create] an Excel spreadsheet with a certain range editable by the client? We were trying the attached template. However, all cells could be edited instead of certain cells. The cells to be edited are under the customer section. We need a way for the customer to edit that portion of the COA. To date, we have been making the COA revisions but it is too cumbersome. We would rather send out this Excel COA with a section for them to complete as they wish. Please let me know as this would save us several hours per week.

Yes, you can have an Excel template that creates workbooks with only specific cells enabled for editing. Just unlock the cells you want your client to be able to edit. To unlock a cell, right-click, choose Format Cells and on the Protection tab clear the Locked option. Locking cells has no effect until the worksheet is protected but you can't protect the worksheet in the template since that would prevent the template's AfterTransferFromLIMS macro from making any changes to the worksheet. So, the solution is to add the following single line of VBA code to the bottom (above the `End Sub` statement) of the template's SaveReport macro:

```
Worksheets(1).Protect
Password:="abc123",
DrawingObjects:=True, Contents:=True,
Scenarios:=True
```

Enter your own case-sensitive password in the line above. Note that anyone who has access to the template can "see" the password in the VBA code. But workbooks created from the template have no macros when you remove the infrastructure sheets so you must make sure the Remove Infrastructure option on the Settings sheet in your template is enabled. The SaveReport macro is only invoked when Remove Infrastructure is enabled.

Using Barcode Hardware

A user recently wrote:

I wanted to inquire about using the barcode system within MSC-LIMS. Is there an app we could use for the barcode scanning or do we need to buy a scan gun?

The simplest solution is to use a keyboard-wedge style scanner. These scanners simply connect between the keyboard and the PC, which makes any scanned barcode data appear to have come from the keyboard. Since no software is required, this is an all-hardware solution. MSC-LIMS uses a public domain Code 3 of 9 barcode font to print barcodes so any scanner must be capable of handling Code 3 of 9 (most do).

If you need to print labels with barcodes, the label styles included with MSC-LIMS are all designed for the inexpensive Dymo LabelWriter printers and their 30252 address labels and will accommodate barcodes. With a Full System license custom label styles can be added if you prefer to design your own label.

(Continued on next page)

(Continued from previous page)

Add Samples to an Existing Batch

Recently, a user submitted this question:

Is it possible to add samples to a batch after it has been logged in? There have been a couple of instances where our clients have left samples at their office or what not and want to have everything reported together.

Yes, you can add a sample to an existing batch. First, log the new sample using the Sample Login screen on the Samples menu. Next, use the Sample Login screen to edit the sample just added then add the existing batch number to the sample. You can also edit an existing sample's batch number to move a sample from one batch to another.

Query Multiple Sample Batches

A user recently asked:

Is there a way to query two batches at the same time?

Yes. You can query multiple batches by entering an expression in the Batch field on the Additional tab of the query controls. To query samples in batches with consecutive batch numbers use this expression:

```
Between 50 and 52
```

To query samples in any number of batches, enter a comma-separated list of batch numbers with this expression:

```
In (50, 51, 55)
```

Hover your mouse over the Batch field on the Additional tab of the query controls to see examples of other batch querying options.

For Customers Only

This section of *MSC-LIMS Insights* is devoted to current users of MSC-LIMS. Here we briefly introduce only the most recent additions to MSC-LIMS.com Customers Only pages. Use your login name and password to log on to the Customers Only section of our website.

MSC-LIMS Insights Archive

All past issues of MSC-LIMS Insights are archived on our web site and accessible to current MSC-LIMS users. You will find valuable tips, techniques, and technical articles to help you get the most out of your MSC-LIMS implementation. Use your browser's search function to locate topics in the article index.

Contact Us

Questions, comments, suggestions? Reach us at:

Mountain States Consulting, LLC
970 West Broadway #471
PO Box 30000
Jackson, Wyoming 83002 USA
Ph +1 307-733-1442
Fax +1 303-379-6850

info@msc-lims.com
www.msc-lims.com