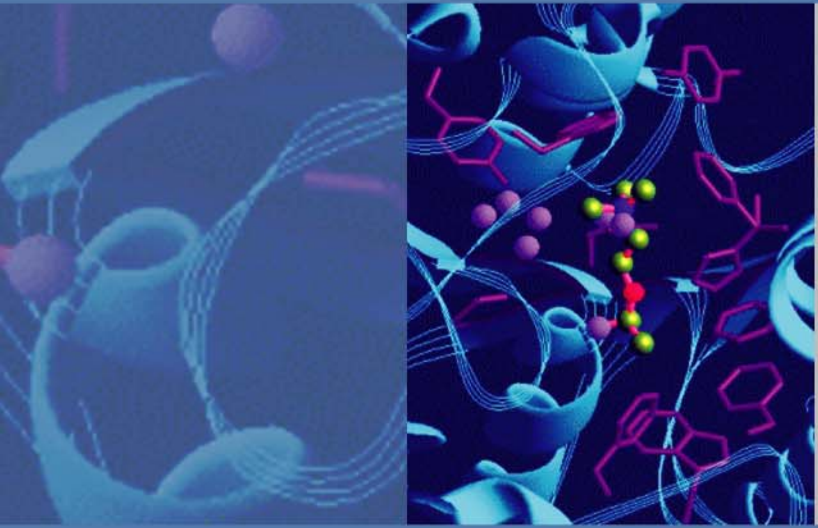


Thermo

TraceFinder

Version 1.1

Custom Reports Tutorial

XCALI-97353 Revision A June 2011

DOCUMENTATION SURVEY

Part of Thermo Fisher Scientific

Thermo SCIENTIFIC

Thermo Fisher Scientific Inc. provides this document to its customers with a product purchase to use in the product operation. This document is copyright protected and any reproduction of the whole or any part of this document is strictly prohibited, except with the written authorization of Thermo Fisher Scientific Inc.

The contents of this document are subject to change without notice. All technical information in this document is for reference purposes only. System configurations and specifications in this document supersede all previous information received by the purchaser.

Thermo Fisher Scientific Inc. makes no representations that this document is complete, accurate or error-free and assumes no responsibility and will not be liable for any errors, omissions, damage or loss that might result from any use of this document, even if the information in the document is followed properly.

This document is not part of any sales contract between Thermo Fisher Scientific Inc. and a purchaser. This document shall in no way govern or modify any Terms and Conditions of Sale, which Terms and Conditions of Sale shall govern all conflicting information between the two documents.

Release history: Revision A

For Research Use Only. Not for use in diagnostic procedures.

Contents

Preface

This tutorial describes how to generate custom TraceFinder reports.

Contents

Related Documentation

In addition to this tutorial, Thermo Fisher Scientific provides the following documents for TraceFinder:

- *TraceFinder User Guide*
- *TraceFinder Administrator Quick Reference Guide*
- *TraceFinder Acquisition Quick Reference Guide*
- *TraceFinder Data Review Quick Reference Guide*
- *TraceFinder Shortcut Menus Quick Reference Guide*

The software also provides Help.

Special Notices

Make sure you follow the precautionary statements presented in this tutorial. The safety and other special notices appear in boxes.

Special notices include the following:

Note Highlights information of general interest.

Tip Highlights helpful information that can make a task easier.

System Requirements

Your system must meet these minimum requirements.

System	Requirements
PC	• 2.33 GHz processor dual core with 2 GB RAM • CD/R-ROM drive • Video card and monitor capable of 1280 × 1024 resolution (XGA) • 75 GB available on the C: drive • NTFS format
Instruments (supported or required)	• AS3000 autosampler
Software	• Microsoft™ Windows™ XP Professional SP3 • Microsoft Office 2007 SP2 or Excel™ 2007 SP2 • Microsoft .NET Framework 3.5 SP 1 • Thermo Foundation™ 1.1 (available on the Xcalibur 2.1.0 CD) • Xcalibur™ 2.1 and Xcalibur 2.1.QF • Adobe Reader 9.0 • NIST 2008

Contacting Us

There are several ways to contact Thermo Fisher Scientific for the information you need.

❖ To contact Technical Support

Phone	800-532-4752
Fax	561-688-8736
E-mail	us.techsupport.analyze@thermofisher.com
Knowledge base	www.thermokb.com

Find software updates and utilities to download at mssupport.thermo.com.

❖ To contact Customer Service for ordering information

Phone	800-532-4752
Fax	561-688-8731
E-mail	us.customer-support.analyze@thermofisher.com
Web site	www.thermo.com/ms

❖ To copy manuals from the Internet

Go to mssupport.thermo.com and click **Customer Manuals** in the left margin of the window.

❖ To suggest changes to documentation or to Help

- Fill out a reader survey online at http://www.surveymonkey.com/s/PQM6P62.

- Send an e-mail message to the Technical Publications Editor at techpubs-lcms@thermofisher.com.

Introduction

Custom reports for the TraceFinder application are based on the Microsoft Excel application using add-in and templates. This tutorial guides you through the process of creating a few typical reports and explains the principles and techniques used to create those reports. After completing this tutorial, you should be able to start creating your own reports for your business needs.

Although it is not mandatory, a knowledge of the Excel application and VBA code is encouraged before beginning the exercises in this tutorial.

In this tutorial, you will create four reports following a step-by-step process. Those reports are representative in different respects and cover most of the fundamentals used:

- Batch Report

  This is the simplest report and highlights essential techniques used for creating the custom reports. The Excel table is used to show a list of samples and their properties in the batch.

- High Density Report

  This report demonstrates how to include graphics in the report and how to create a repeating area for each compound in the template.

- Calibration Report

  This report explains how to create a cross-tab (called a pivot table in the Excel application) type reports.

- DCC Report

  This report uses static method data with help of an internal mapping table to create the report.

Overview

This chapter explains the templates, workflow, customization levels, and Excel application features used in generating custom reports in the TraceFinder application.

Contents

Template Architecture

The custom report templates use a two-tiered design. They use Microsoft Excel's add-in component to provide more generic services, such as the framework and the library, for all templates. At each template level, you can write Visual Basic for Applications (VBA) code to control the report layout and format the output data.

The add-in component provides the following major functions:

- Handles the startup parameters for the template that are passed in from the TraceFinder application. The application uses these parameters to fully automate the generation of custom reports.
- Provides a framework for creating the headers and footers for all reports. A default set of headers and footer is included in the delivered add-in, but you can override these in each template to further customize the reports.
- Provides utility functions specific to the TraceFinder application reports to hide internal complexity and to facilitate customization at the template level. For example, you can use a one-line add-in function to add graphics to the report or to create a cross-tab-type report.

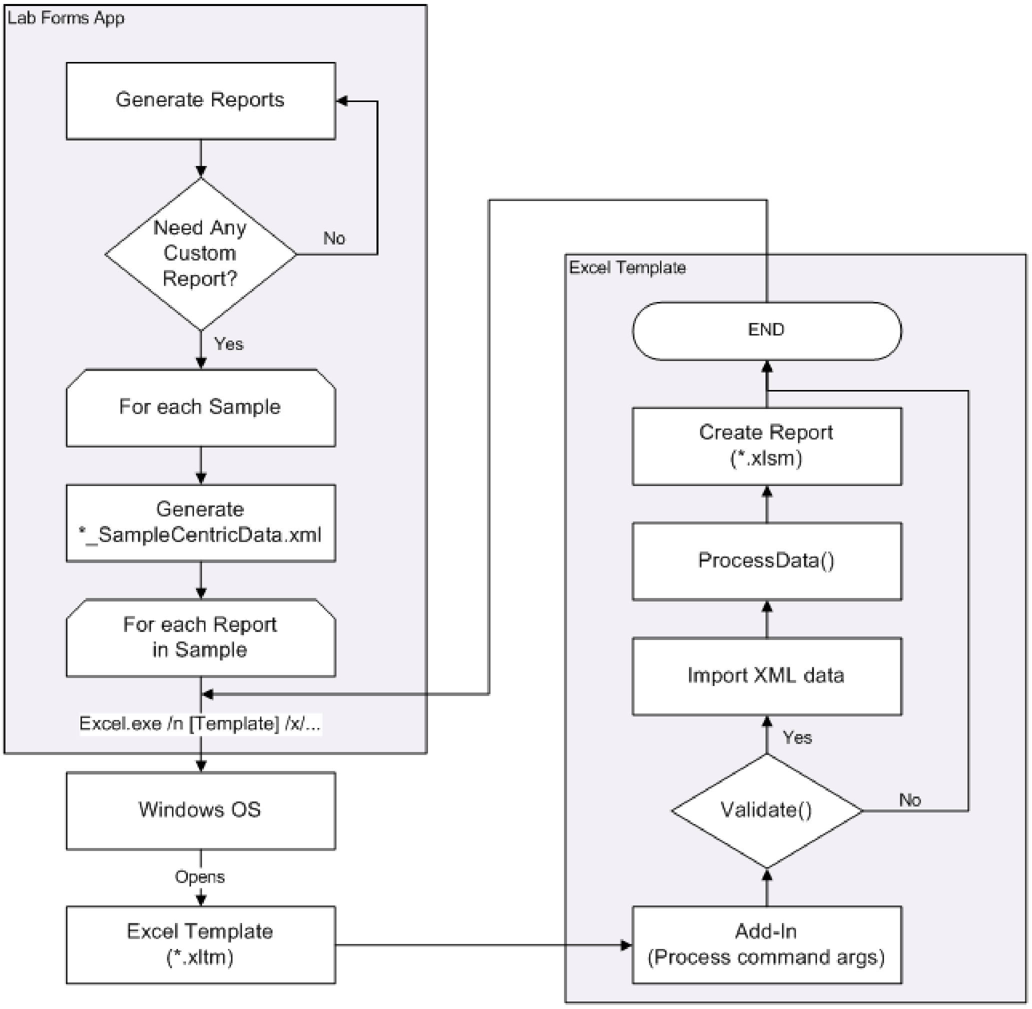
The add-in module is designed so that you rarely need to change it for your customization. It is password-protected. If you need to modify the add-in code, contact your Thermo Fisher Scientific representative for the password.

The VBA code used in each template is kept to a minimum. In general, most of the templates use fewer than 100 lines of VBA code.

Custom Report Generation

The TraceFinder application generates custom reports sequentially. The following picture shows how custom reports are generated.

Figure 1. Generating custom reports



The workflow involved in creating custom reports with the finished templates is as follows:

1. Configure the TraceFinder application to enable the generation of custom reports. See the *TraceFinder User Guide* for more details. You can include all report templates in the C:\Thermo\Shared\Templates\Reports folder in the application.

2. When you submit a custom report for batch processing, the TraceFinder application automatically generates a sample-specific data file for each sample in XML format. These files are located in the Data folder. They have names similar to *RawFileName*_SampleCentricData.xml, where *RawFileName* is the name of the raw file for the sample.

- This XML file includes all the data needed for creating sample-specific reports. In most cases, you need only refer to this one file for a report.

- In some reports, you might need to refer to more than one sample-specific raw data file.

- The *RawFileName*_SampleCentricData.xml file also contains calibration data, which is batch-level data. Because this file is created as each sample is processed, the batch-level data might contain partial results until the last sample is finished.

3. To generate a custom report, the application creates a Windows command line for the Excel application and passes it to operating system. The operating system opens the Excel application with the specified template. The template processes the remaining parameters through the add-in component, which loads and processes the data to generate the report.

   If you are a regular user of the TraceFinder applications, the details of this process are hidden. However, if you are a template developer, it is helpful to understand this process.

   The Excel command line looks like this:

   ```
   excel.exe /n TEMPLATENAME /x/CUSTOMPARAMETERS
   ```

   where:

   - /n tells the Windows operating system that you are using this template to create an Excel spreadsheet, not editing the template itself. Without this switch, the Excel application would open the specified template for editing.

   - /x indicates that all subsequent parameters be passed to the add-in component for processing.

   Chapter 3, "Getting Started," describes the command line in more detail.

   The Windows operating system opens the Excel application with the specified template. It passes all the parameters after /x to the template for further processing. The template processes the parameters through the add-in component, which loads and processes the data to generate the report. After the custom parameters are processed, the first function that the add-in component calls is the Validate() function on the IUtil interface, which is implemented by all templates. This function is primarily designed to protect the template design from accidental changes. If Validate() succeeds, it starts importing XML data. After it imports all the data, the add-in calls the ProcessData() function on the IUtil interface to perform all the post-processing. The function then notifies the TraceFinder application that the report is completed and that it is ready for the next report, if any.

Customization Levels

The Excel application provides a flexible way to customize your reports according to your needs and technical expertise.

- You can customize the appearance of your reports by making changes to the template. You do not need to know VBA programming. For example, if you have some basic familiarity with Excel, you can change the font, resize a column, or format a number. With a little training, you can add or remove columns in a list.
- If you know VBA programming, you can customize the template logic by modifying the code that the template uses. At this level, you do not need to change the add-in code, because you can customize the template code.
- You can perform advanced customization if you need to change the add-in framework. By referring to the existing add-in component, you can create completely different kinds of templates for your needs. The DCC report is an example of this kind of customization.

Excel's Features and Limitations

Following is a short list of Excel's additional features and limitations that affect the creation of custom reports:

- Named ranges are useful features of the Excel application, and the report-generation process uses them extensively.
- Excel's XML support enables you to create custom reports based on a single XML file as a data source.
- Excel's flexible formatting enables you to create professional-looking reports.
- The Excel application does not use relational data. All its data follows a hierarchical structure. As a result, some of the data in source XML files might be duplicated.
- Each mapped XML data source can only be referenced once in the workbook. If you want to use the same field more than once, you must add the same source as an XML map for each usage.
- All images are embedded in the XML file as byte arrays. However, Excel's cells have a data size limit that precludes Excel from loading the byte arrays directly into the cells. Therefore, it loads all images programmatically into the spreadsheet at run time. Placing graphics onto a spreadsheet just creates a placeholder for graphics.
- The Excel application does not offer an Undo function for any macros or VBA code. It clears its undo stack completely when it executes any macro or VBA code.
- Excel's copy/paste function is powerful, but it uses the Windows clipboard to pass around the data. When you use the Windows clipboard to move data on the spreadsheet while running the reporting process in the background, you might inadvertently cause an incorrect report to be generated. The add-in component provides a special copy/paste function circumvent this problem.

Tip In general, the copy/paste function becomes slower as the amount of data on a sheet increases. The Excel application can more easily handle many tabs with a small set of data on each tab, so many reports feature several tabs, with each tab representing one printed page.

3

Getting Started

In this chapter, you will create a ThermoCustomReportBaseTemplate file to be used in subsequent chapters as a base for creating specific report templates.

Contents

Prerequisites

To create custom reports using the Excel application, you must configure your system with the following required components. If you have the TraceFinder application installed and have generated any custom reports, you have met these requirements:

- Microsoft Excel 2007 SP2 (12.0.6524 or above)

  To verify your version of the Excel application, do the following:

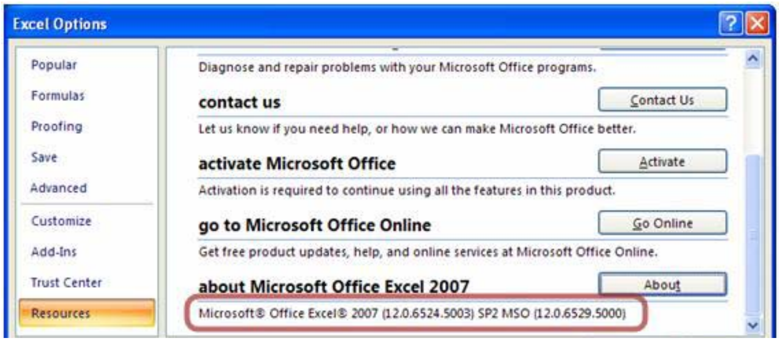
  a. Click the **Excel application** icon, .

  b. Click the **Excel Options** button, .

  c. Click **Resources**.

d. Verify your version in the **About Microsoft Office Excel 2007** area.

Figure 2. Excel Options Resources dialog box



Diagnose and repair problems with your Microsoft Office programs.

contact us

Let us know if you need help, or how we can make Microsoft Office better.

activate Microsoft Office

Activation is required to continue using all the features in this product.

go to Microsoft Office Online

Get free product updates, help, and online services at Microsoft Office Online.

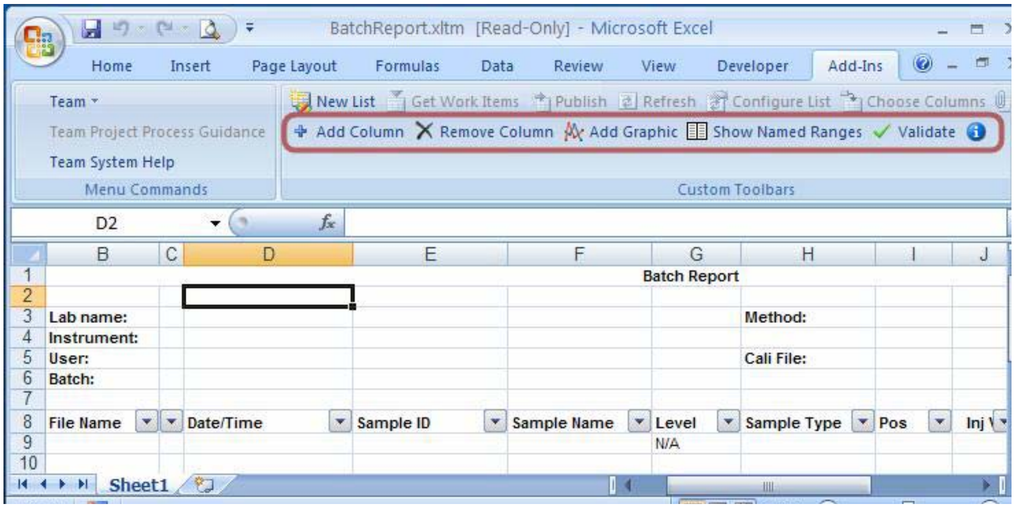
about Microsoft Office Excel 2007

Microsoft® Office Excel® 2007 (12.0.6524.5003) SP2 MSO (12.0.6529.5000)

- Thermo Report Add-In (Thermo Report Add-In.xla)

Load any custom template (*.xltm). If the add-in is successfully loaded, you will see the Add-Ins menus as shown in the following figure.

Figure 3. Add-Ins menus



	B	C	D	E	F	G	H	I	J
1						Batch Report			
2									
3	Lab name:						Method:		
4	Instrument:								
5	User:						Cali File:		
6	Batch:								
7									
8	File Name		Date/Time	Sample ID	Sample Name	Level	Sample Type	Pos	Inj V
9						N/A			
10									

The Thermo Report Add-In.xla file is located in the following folder:

C:\Program Files\Microsoft Office\Office12\Library

The Excel application template files (*.xltm) are installed in the following folder:

C:\Thermo\Shared\Templates\Reports

Creating the Base Template

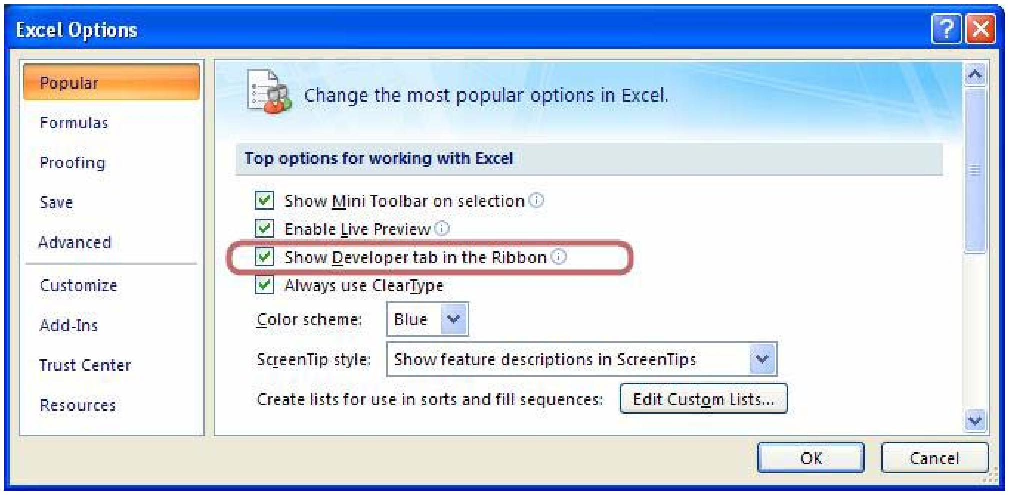
The ThermoCustomReportBaseTemplate file is a basic template that you will use as a starting point to create specific report templates.

❖ To enable the Developer tab in the Excel application

1. Click the **Excel application** icon, .

2. Click the **Excel Options** button.

3. Select the **Show Developer Tab in the Ribbon** check box and click **OK**.



Excel Options

Change the most popular options in Excel.

Top options for working with Excel

☑ Show Mini Toolbar on selection
☑ Enable Live Preview
☑ Show Developer tab in the Ribbon
☑ Always use ClearType

Color scheme: Blue

ScreenTip style: Show feature descriptions in ScreenTips

Create lists for use in sorts and fill sequences: Edit Custom Lists...

4. Start the Excel application.

   The Excel application displays the Developer tab.

❖ To load the ThermoReportAddIn function

1. Click the **Developer** tab.

   The ribbon expands to display the Developer functions.

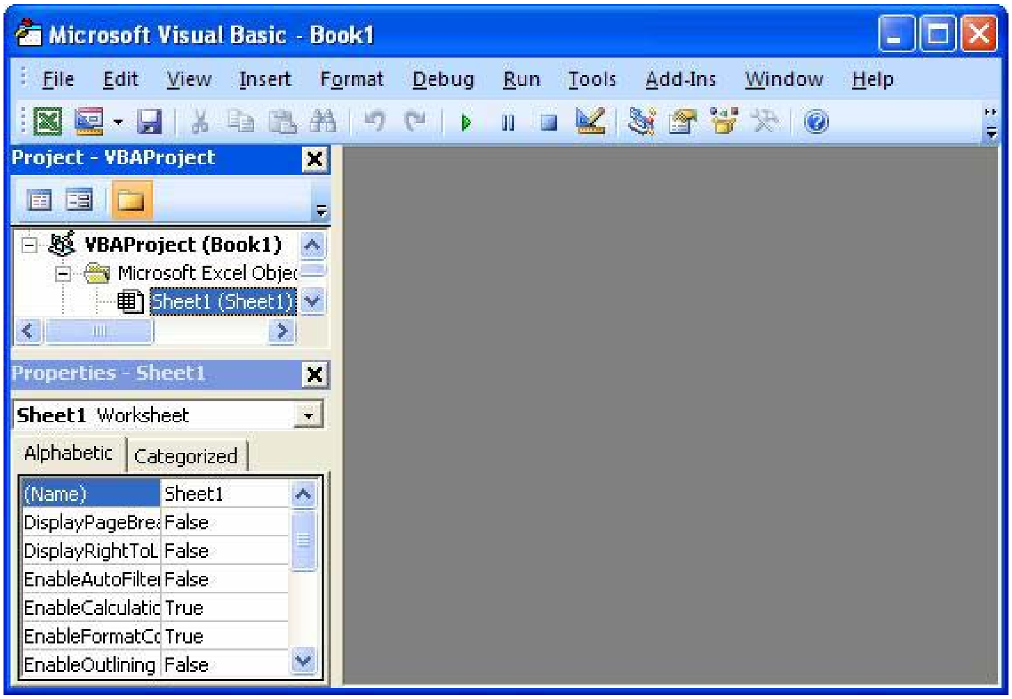
2. Click the **Visual Basic** button.

   The Microsoft Visual Basic window opens.

3. Choose **View > Properties Window**.

   The Properties pane opens.

Figure 4. Microsoft Visual Basic window with Properties pane



VBAProject (Book1)
Microsoft Excel Objec
Sheet1 (Sheet1)

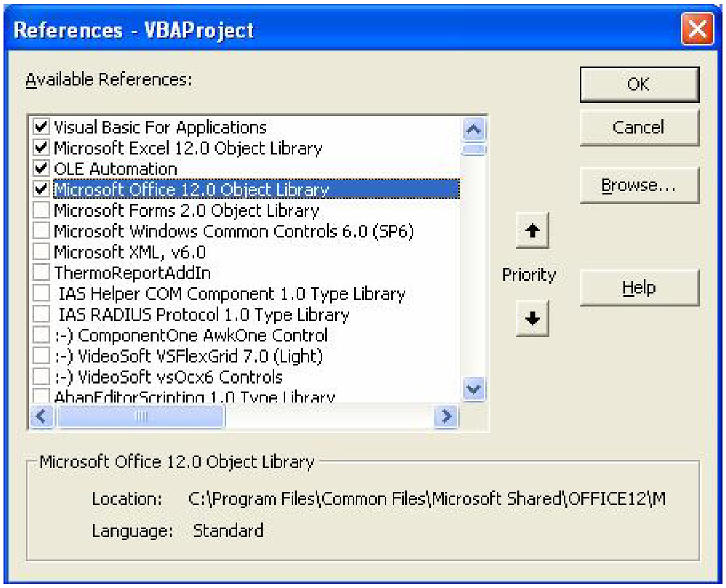
Sheet1 Worksheet

(Name)	Sheet1
DisplayPageBrea	False
DisplayRightToL	False
EnableAutoFilter	False
EnableCalculatic	True
EnableFormatCc	True
EnableOutlining	False

4. Choose **Tools** > **References**.

   The References - VBAProject dialog box opens. The check marks indicate active references, which appear at the top of the list.

Figure 5. References - VBAProject dialog box



- ☑ Visual Basic For Applications
- ☑ Microsoft Excel 12.0 Object Library
- ☑ OLE Automation
- ☑ Microsoft Office 12.0 Object Library
- □ Microsoft Forms 2.0 Object Library
- □ Microsoft Windows Common Controls 6.0 (SP6)
- □ Microsoft XML, v6.0
- □ ThermoReportAddIn
- □ IAS Helper COM Component 1.0 Type Library
- □ IAS RADIUS Protocol 1.0 Type Library
- □ :-) ComponentOne AwkOne Control
- □ :-) VideoSoft VSFlexGrid 7.0 (Light)
- □ :-) VideoSoft vsOcx6 Controls
- □ AhanEditorScripting 1.0 Type Library

Microsoft Office 12.0 Object Library

Location: C:\Program Files\Common Files\Microsoft Shared\OFFICE12\M

Language: Standard

5. To load the ThermoReportAddIn reference, do one of the following:

   – If **ThermoReportAddIn** is selected in the Available References list, it is correctly set up. Click **OK**.

   – If **ThermoReportAddIn** is in the Available References list but not selected, select the check box, and click **OK**.

   > **Tip** Scroll down to find it; unselected references are listed alphabetically.

   – If **ThermoReportAddIn** is not in the Available References list, manually add it as follows:

     i. Click the **Browse** button.

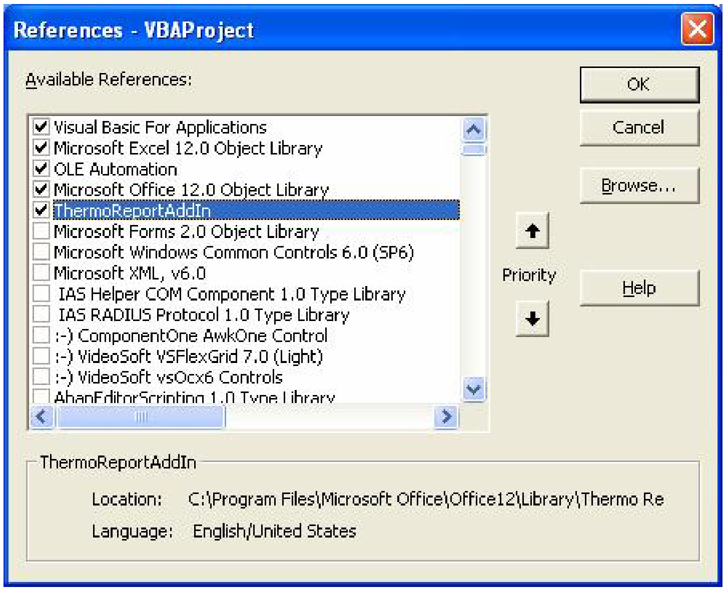
     ii. Navigate to the C:\Program Files\Microsoft Office\Office12\Library folder.

     iii. Change Files of Type to **All files(*.*)**.

     iv. Select the **Thermo Report Add-In.xla** add-in file.

     v. Click **Open**.
     The application adds **ThermoReportAddIn** to the Available References list in the References dialog box.

Figure 6. References - VBAProject dialog box with ThermoReportAddIn



References - VBAProject

Available References:

☑ Visual Basic For Applications
☑ Microsoft Excel 12.0 Object Library
☑ OLE Automation
☑ Microsoft Office 12.0 Object Library
☑ ThermoReportAddIn
☐ Microsoft Forms 2.0 Object Library
☐ Microsoft Windows Common Controls 6.0 (SP6)
☐ Microsoft XML, v6.0
☐ IAS Helper COM Component 1.0 Type Library
☐ IAS RADIUS Protocol 1.0 Type Library
☐ :-) ComponentOne AwkOne Control
☐ :-) VideoSoft VSFlexGrid 7.0 (Light)
☐ :-) VideoSoft vsOcx6 Controls
☐ AbanEditorScripting 1.0 Type Library

ThermoReportAddIn

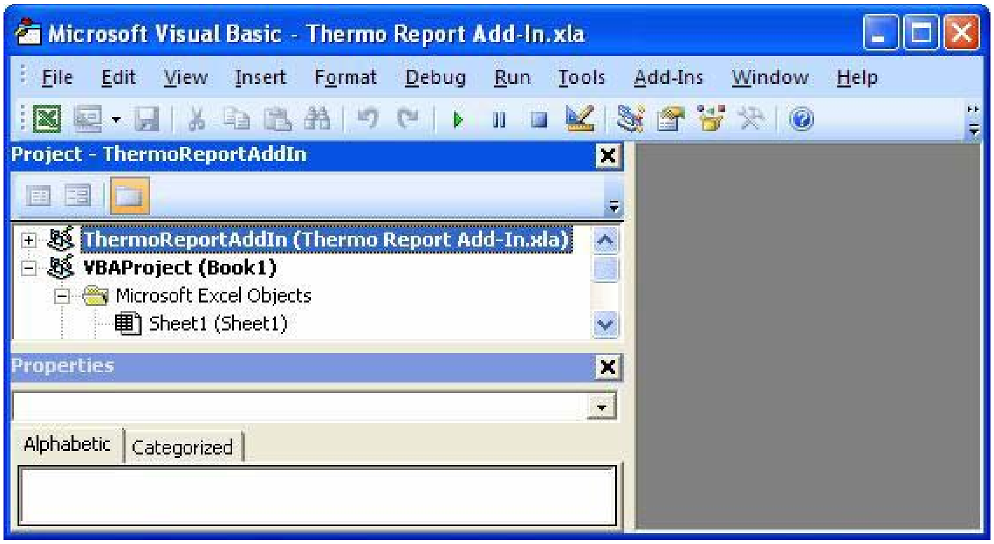
Location: C:\Program Files\Microsoft Office\Office12\Library\Thermo Re

Language: English/United States

     vi. Select the **ThermoReportAddIn** check box and click **OK**.

The Microsoft Visual Basic application loads the ThermoReportAddIn project into the Visual Basic dialog box.

Figure 7. Microsoft Visual Basic dialog box with ThermoReportAddIn


Microsoft Visual Basic - Thermo Report Add-In.xla
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - ThermoReportAddIn
ThermoReportAddIn (Thermo Report Add-In.xla)
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
Properties
Alphabetic
Categorized

> **Note** The ThermoReportAddIn project is password protected. If you try to expand the project by clicking the plus (+) sign, it prompts you for the password. The add-in was designed so that most users do not need to change it to create their custom reports. If you must change the add-in behavior, contact your Thermo Fisher Scientific representative for the password.

The ThermoReportAddIn project is now referenced in your project, but until you use the add-in project, it is not part of your template. If you exit the Excel application and reopen this project, you will not see this reference again.

You are now ready to use the add-in to ensure that the add-in project is associated with your template.

❖ To connect the basic elements to the add-in component

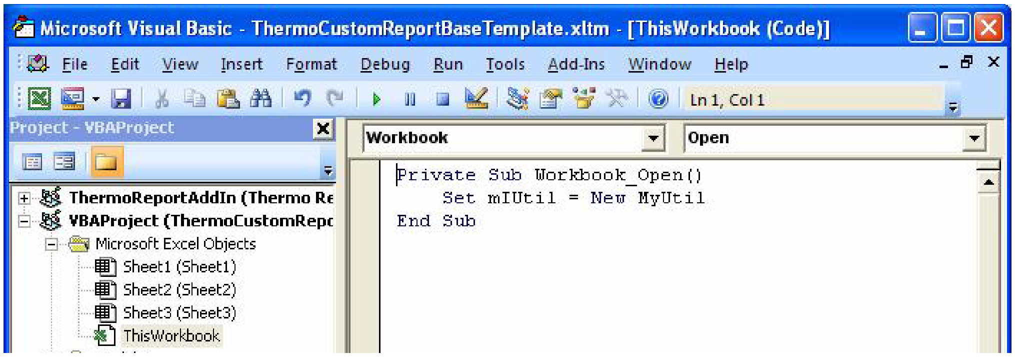
1. Double-click **ThisWorkbook** in the VBAProject.

   The code editor pane opens.

2. Enter the following code in the code editor:

```
Private Sub Workbook_Open()
    Set mIUtil = New MyUtil
End Sub
```

Figure 8. Workbook code editor


Microsoft Visual Basic - ThermoCustomReportBaseTemplate.xltm - [ThisWorkbook (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - VBAProject
ThermoReportAddIn (Thermo Re
VBAProject (ThermoCustomRepo
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Workbook
Open

```
Private Sub Workbook_Open()
    Set mIUtil = New MyUtil
End Sub
```

When you open the workbook, it will call the **MyUtil** function.

The code connects the MyUtil function to the process and implements the IUtil interface. The function connects your template to the add-in component.

You are now ready to add the ProcessData and Validate subroutines to the template. For more explanation on these two functions, see Chapter 2, “Overview.”

❖ To add a module to the project

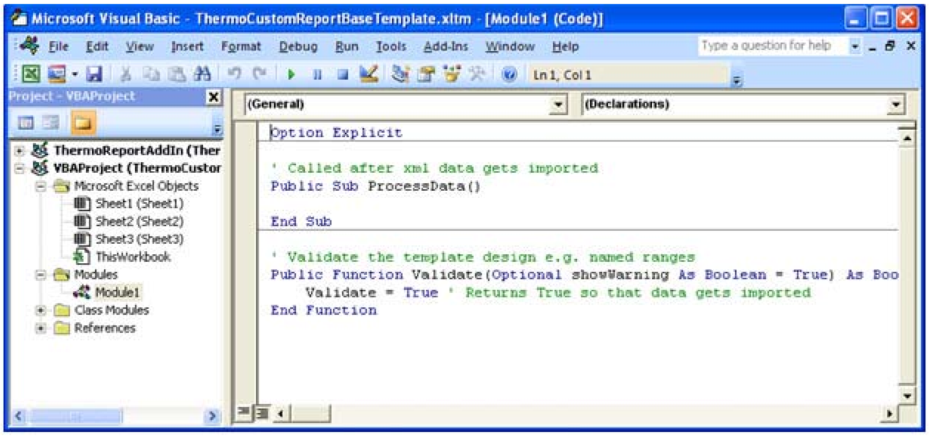
1. Right-click the VBA project and choose **Insert > Module** from the shortcut menu.

   The application adds a folder named Modules to the project, adds a module named Module1 to the folder, and opens a code editor.

2. Enter the following code in the Module1 code editor:

```
Option Explicit

Public Sub ProcessData()

End Sub

Public Function Validate(Optional showWarning As Boolean = True) As
Boolean Validate = True

End Function
```

Figure 9. Module1 code editor



```
Option Explicit

' Called after xml data gets imported
Public Sub ProcessData()

End Sub

' Validate the template design e.g. named ranges
Public Function Validate(Optional showWarning As Boolean = True) As Boo
    Validate = True ' Returns True so that data gets imported
End Function
```

The ProcessData and Validate subroutines are the most fundamental functions in your template.

- **Validate()** is called before the XML data source is loaded into the spreadsheet to verify that everything on the design interface is correct (for example, if name ranges are intact). This function must return True to be able to import data. If it returns False, you will see an error message explaining the problem.

- **ProcessData()** is called after the XML data source is imported and ready for further processing (for example, to extract data).

You are now ready to add functions that give you control of headers and footers in your reports. The **ThermoReportAddIn** function has a default implementation of this interface using **mIUtil**.

❖ To add a MyUtil class to the project

1. Right-click the VBA project and choose **Insert** > **Class Module** from the shortcut menu.

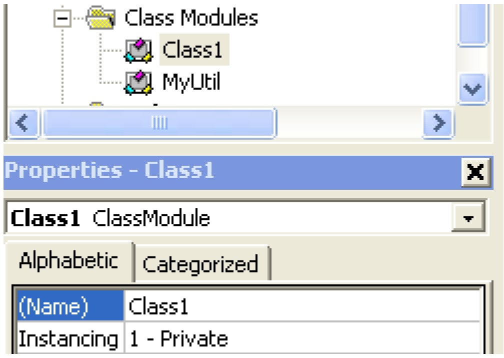
   The application adds a Class Modules folder and Class1 module to the project and opens a code editor.

2. To change the Class1 name to MyUtil, do the following:

   a. Select **Class1**.

   b. Choose **View** > **Properties Window**.

   The Properties pane for Class1 opens.



(Name)	Class1
Instancing	1 - Private

   c. Change Class1 to **MyUtil** and press ENTER.

   d. Close the Properties pane.

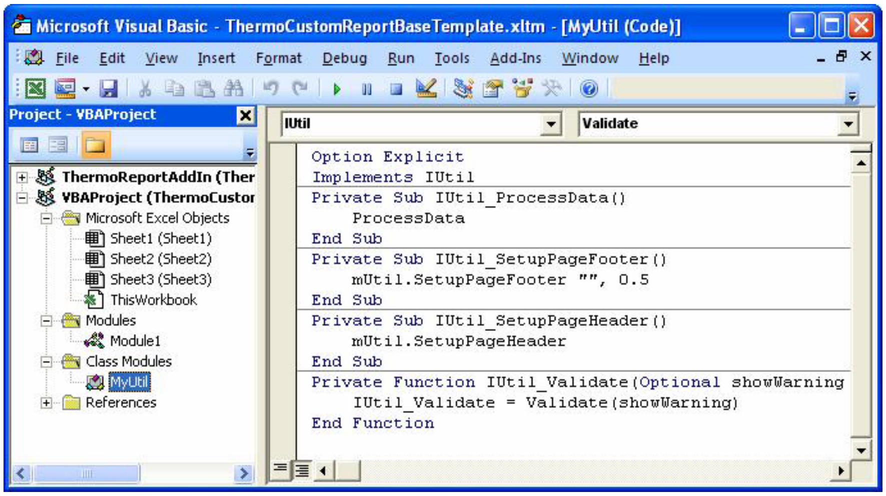
3. Double-click **MyUtil** to open the code editor.

4. Enter the following code in the MyUtil code editor:

```
Option Explicit
Implements IUtil
Private Sub IUtil_ProcessData()
    ProcessData
End Sub
```

```
Private Sub IUtil_SetupPageFooter()
    mUtil.SetupPageFooter "", 0.5
End Sub
Private Sub IUtil_SetupPageHeader()
    mUtil.SetupPageHeader
End Sub
Private Function IUtil_Validate(Optional showWarning As Boolean =
True) As Boolean
    IUtil_Validate = Validate(showWarning)
End Function
```

Figure 10. MyUtil code editor



```
Option Explicit
Implements IUtil
Private Sub IUtil_ProcessData()
    ProcessData
End Sub
Private Sub IUtil_SetupPageFooter()
    mUtil.SetupPageFooter "", 0.5
End Sub
Private Sub IUtil_SetupPageHeader()
    mUtil.SetupPageHeader
End Sub
Private Function IUtil_Validate(Optional showWarning
    IUtil_Validate = Validate(showWarning)
End Function
```

The ProcessData, SetupPageFooter, SetupPageHeader, and Validate functions in the IUtil interface give you more control on the template level.

You are now ready to map the XML data file to the template. To do this you must have a *RawFileName*_SampleCentricData.xml file.

When you created a custom report in the TraceFinder application, the application created a *RawFileName*_SampleCentricData.xml file that acts as a data source file for the custom reports. The sample-centric data is usually located in the Data folder of the batch. For example, C:\Thermo\TraceFinder\Projects\Subprojects\Batches\Batches\ML_BATCH_01\Data.

If you do not have the TraceFinder application installed, you can use any *RawFileName*_SampleCentricData.xml file to create the report templates in this tutorial.

❖ To associate the template with the schema of the data source

1. Close the Microsoft Visual Basic project.

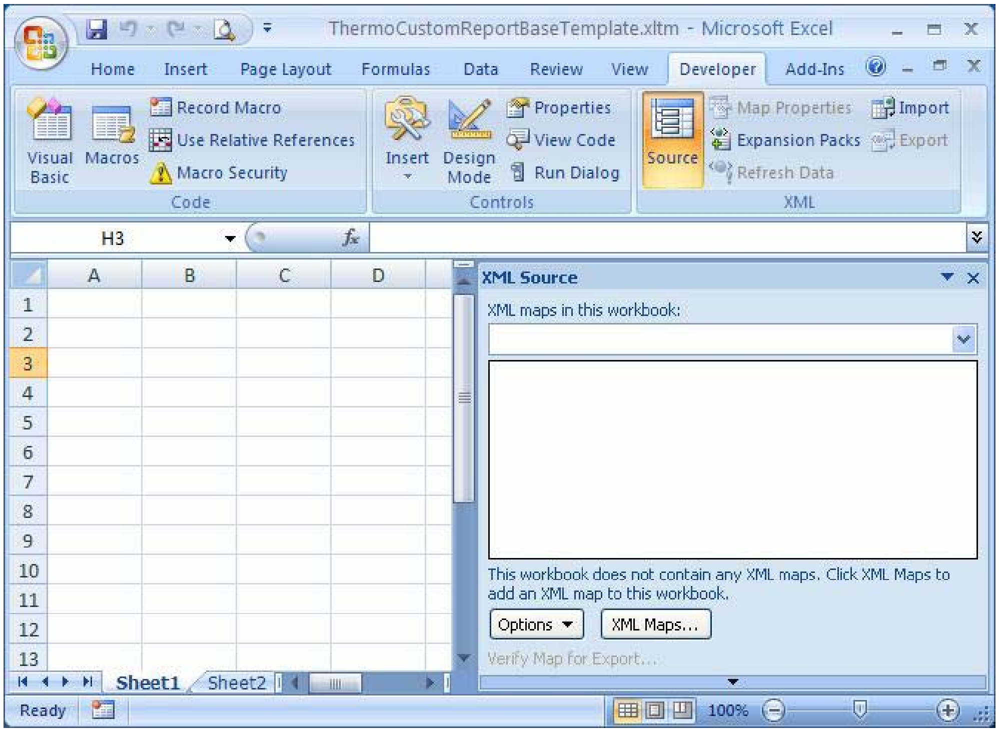
2. In the Excel spreadsheet, click the **Developer** tab to display all the Developer functions.

3. Click the **Source** button, Source.

The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet.

By default, the Excel application automatically creates three sheets. Only two are visible in the following figure.

Figure 11. Excel spreadsheet with XML Source pane

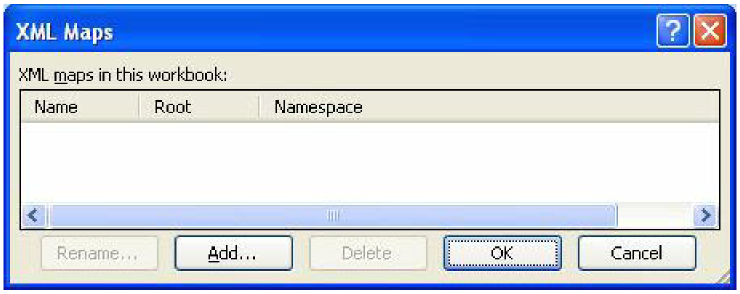


This workbook does not contain any XML maps. Click XML Maps to add an XML map to this workbook.

4. Click **XML Maps**.

   The XML Maps dialog box opens.

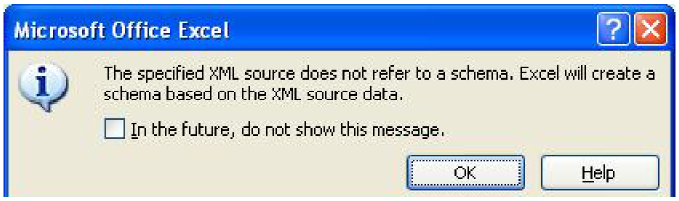
Figure 12. XML Maps dialog box



5. Click **Add** and navigate to the following file:

   *RawFileName*_SampleCentricData.xml

You might get the following warning message:



Microsoft Office Excel

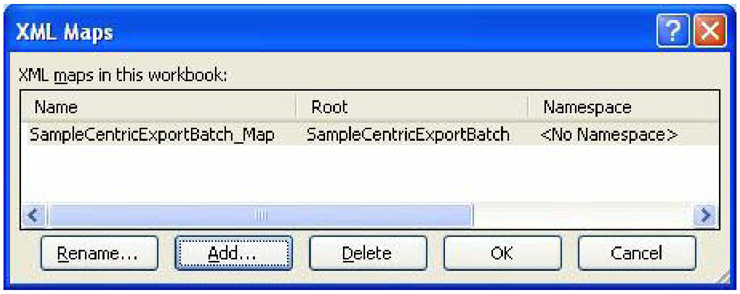
The specified XML source does not refer to a schema. Excel will create a schema based on the XML source data.

□ In the future, do not show this message.

6. Select the **In the Future…** check box and click **OK**.

   The application adds the imported schema to your XML maps list.

Figure 13. XML Maps dialog box with the XML map



XML Maps

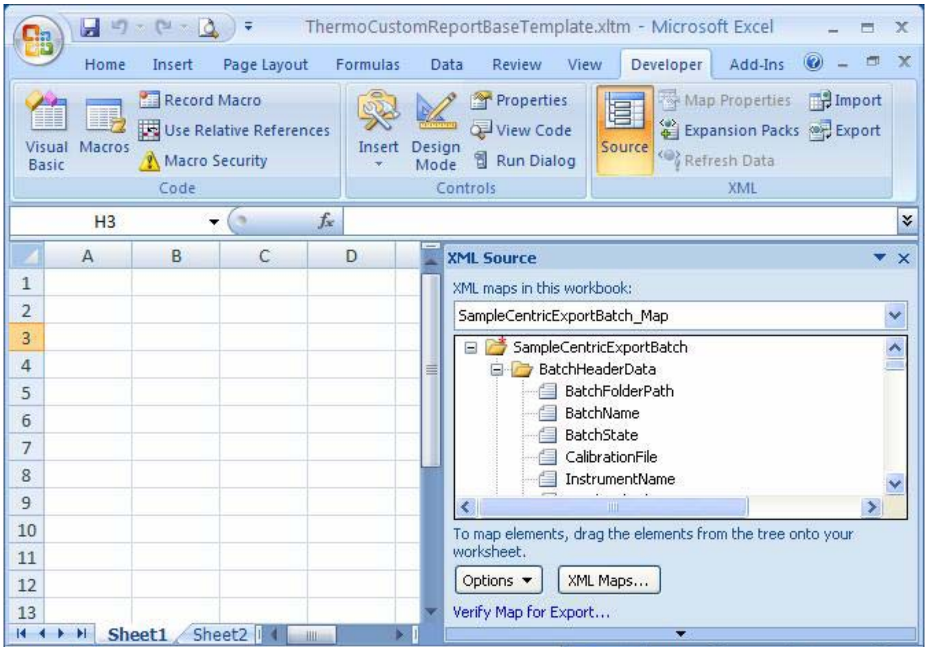
XML maps in this workbook:

Name	Root	Namespace
SampleCentricExportBatch_Map	SampleCentricExportBatch	<No Namespace>

7. Click **OK**.

   The Excel application adds the XML map to the XML Source pane.

Figure 14. XML Source pane with XML map



ThermoCustomReportBaseTemplate.xltm - Microsoft Excel

XML Source

XML maps in this workbook: SampleCentricExportBatch_Map

SampleCentricExportBatch
BatchHeaderData
BatchFolderPath
BatchName
BatchState
CalibrationFile
InstrumentName

To map elements, drag the elements from the tree onto your worksheet.

8. Scroll through the XML Source list to see the available data fields.

The template is now ready to use for creating reports. You are now ready to save the template to use for all the custom reports in this tutorial.

❖ To save the Excel workbook as a macro-enabled template

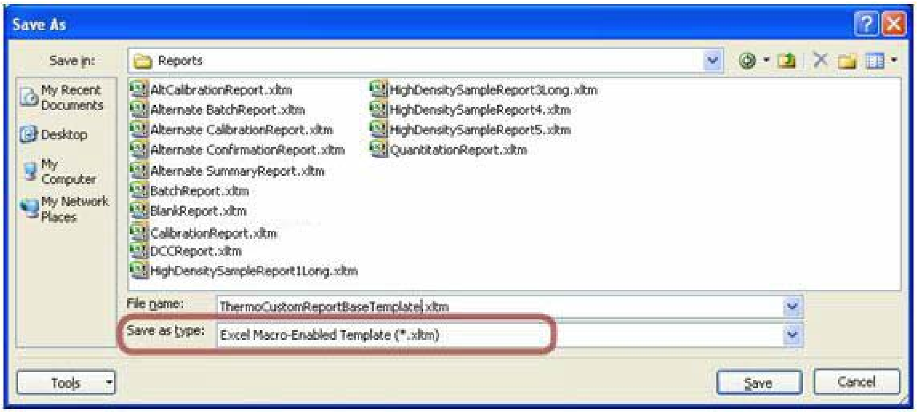
1. Click the **Excel application** icon, , and choose **Save As**.

   By default, the Excel application tries to save this workbook as **Excel Workbook (*.xlsx)**, which is a spreadsheet without macro support.

2. Change the Save as Type to **Excel Macro-Enabled Template (*.xltm)**.

3. Change the File Name to **ThermoCustomReportBaseTemplate.**

Figure 15. Save As dialog box



Save in: Reports

AltCalibrationReport.xltm
Alternate BatchReport.xltm
Alternate CalibrationReport.xltm
Alternate ConfirmationReport.xltm
Alternate SummaryReport.xltm
BatchReport.xltm
BlankReport.xltm
CalibrationReport.xltm
DCCReport.xltm
HighDensitySampleReport1Long.xltm
HighDensitySampleReport3Long.xltm
HighDensitySampleReport4.xltm
HighDensitySampleReport5.xltm
QuantitationReport.xltm

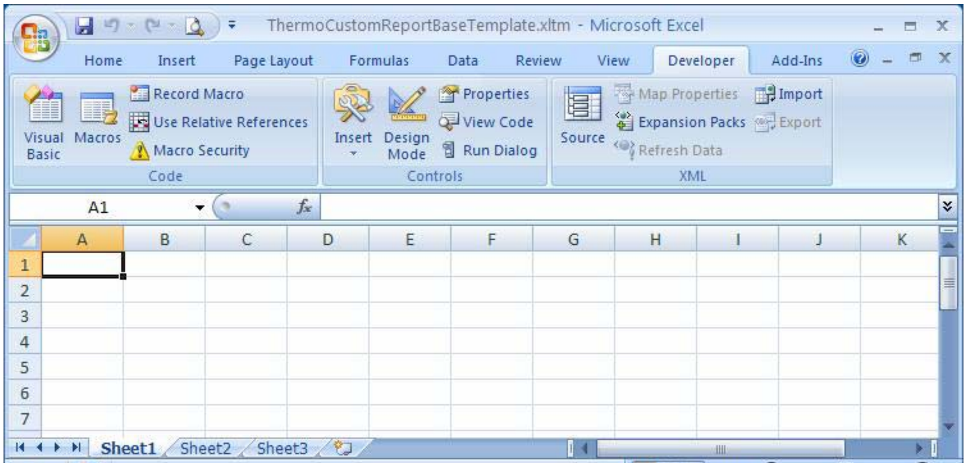
File name: ThermoCustomReportBaseTemplate.xltm

Save as type: Excel Macro-Enabled Template (*.xltm)

> **Note** By default, the TraceFinder application will use only the Excel templates in the folder C:\Thermo\Shared\Templates\Reports.

This template will serve as the base for all of your custom report templates in this tutorial. Throughout this tutorial, you will name your custom report templates with the prefix “My” to avoid a naming conflict with the delivered reports.

Figure 16. Completed base template



If you were unable to complete this base template, you can copy the ThermoCustomReportBaseTemplate.xltm file from the following location:

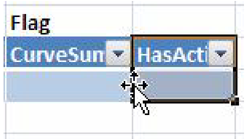
```
C:\Thermo\Custom Report Tutorials
```

Working with Columns

This section describes the procedures you can use to add, insert, move, or delete a column.

❖ To move a column in a data table

1. Click the column header to select the column.
2. Pause your mouse on the border area of the selected column.



3. When the mouse pointer changes to the four-way arrow, drag the column to the new location.

   - If you drag the column to an empty column, the contents of the dragged column replace the empty column.
   - If you drag the column to an occupied column, a warning dialog box asks if you want to replace the contents of the destination column.

❖ To insert a column in a data table

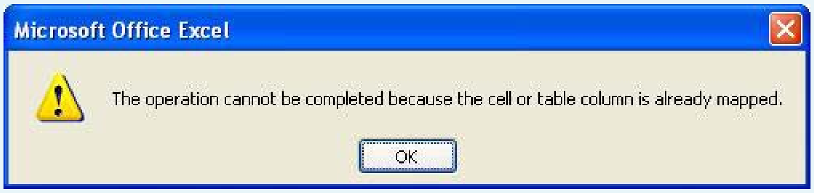
1. Insert an empty column by right-clicking an adjacent column header and selecting any of the commands from the **Insert** menu.

   The Excel application inserts an empty, unmapped column to the left or right of the selected column. By default, the new column is named Column*n*.

2. Drag a tree node to the empty column.

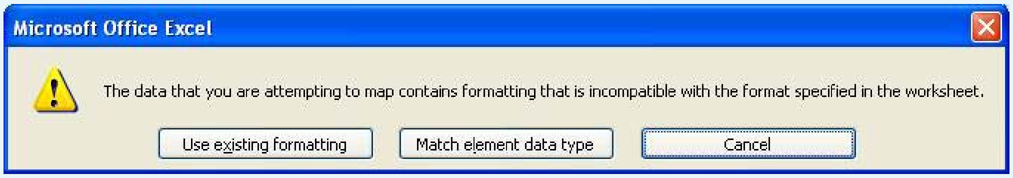
Note The Excel application does not let you insert a new column by directly dragging a tree node into the middle of the data table; you must drag a node to an empty column and overwrite the empty column or drag the node to the beginning or end of the data table and move it later.

If you attempt to overwrite a mapped column, you get the following error message:


Microsoft Office Excel

The operation cannot be completed because the cell or table column is already mapped.

Note If you get the following error message, click the **Match element data type** button to dismiss it.



Microsoft Office Excel

The data that you are attempting to map contains formatting that is incompatible with the format specified in the worksheet.

If you are concerned that your node is not correctly mapped, there are several ways to verify a mapped node.

❖ To verify a mapped field in the template

Do one of the following:

- Select the column.

  If the column is mapped to a tree node, the Excel application selects the mapped node in the XML Source tree and displays it in bold.

- Select the mapped node in the XML Source tree.

  All mapped nodes are displayed in bold font. If the node is mapped to a column, the Excel application automatically selects the column in the data table.

When you insert an empty column and then drag an XML Source node to the empty column, the default column header does not change.

Note A "node" in the XML Source tree creates a "field" in the spreadsheet.

❖ To restore a mapped name to a column header

Select and delete the default cell name, for example **Column1**.

Because a column header in a data table cannot be empty, the Excel application automatically uses the mapped node name.

❖ To delete an entire column from a data table

Right-click the column header cell and choose **Delete** from the shortcut menu.

❖ To delete a portion of a column in a data table

1. Click the column header to select the column.
2. Right-click the column and choose **Delete > Table Columns** from the shortcut menu.

Note Do not press the **Delete** key to delete the column.

❖ To remove the mapping from the XML Source tree

1. Select the node in the XML Source pane.
2. Right-click the selected (bold) node, and choose **Remove Element** from the shortcut menu.

   The Excel application removes the link between the node and the data table column. It does not delete the data table column. The data table column is unmapped and the previous mapped name remains.

Security Settings

For the macro-enabled templates to run in the Excel application, you must add the directory of the add-in and the template files to the list of trusted locations.

- When you installed the TraceFinder application, the default template folder, C:\Thermo\Shared\Templates\Reports, was added to the Excel application's list of trusted locations. The default security setting works as long as you run your templates from this folder.
- If you want to save your templates to a different folder, you can add that folder to the trusted locations.

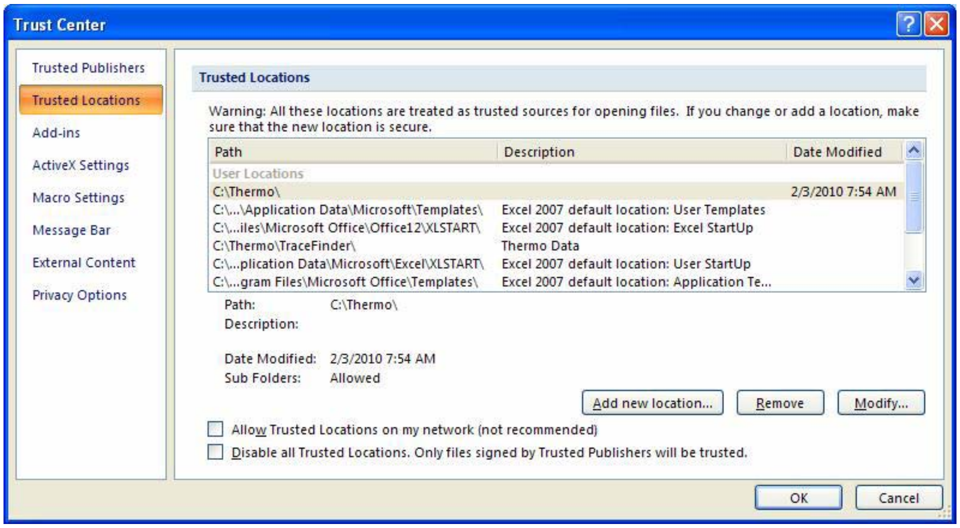
❖ To add a folder to the Trusted Locations list

1. Click the **Excel application** icon, .
2. Click the **Excel Options** button, Excel Options.

   The Excel Options dialog box opens.
3. Click **Trust Center**, and then click **Trust Center Settings**.

   The Trust Center dialog box opens.

Figure 17. Trust Center dialog box



Trusted Locations

Warning: All these locations are treated as trusted sources for opening files. If you change or add a location, make sure that the new location is secure.

Path	Description	Date Modified
User Locations		
C:\Thermo\		2/3/2010 7:54 AM
C:\...\Application Data\Microsoft\Templates\	Excel 2007 default location: User Templates	
C:\...iles\Microsoft Office\Office12\XLSTART\	Excel 2007 default location: Excel StartUp	
C:\Thermo\TraceFinder\	Thermo Data	
C:\...plication Data\Microsoft\Excel\XLSTART\	Excel 2007 default location: User StartUp	
C:\...gram Files\Microsoft Office\Templates\	Excel 2007 default location: Application Te...	

Path: C:\Thermo\
Description:

Date Modified: 2/3/2010 7:54 AM
Sub Folders: Allowed

□ Allow Trusted Locations on my network (not recommended)
□ Disable all Trusted Locations. Only files signed by Trusted Publishers will be trusted.

4. Click **Trusted Locations**.
5. Click the **Add New Location** button and navigate to the location of your folder.
6. Click **OK** in the Microsoft Office Trusted Location dialog box.
7. Click **OK** in the Trust Center dialog box.
8. Click **OK** in the Excel Options dialog box.

Your macro-enabled templates will now run in the Excel application.

Testing the Template

You are now ready to run a test from the Windows **Start > Run** box. In this test, the code you enter does the following:

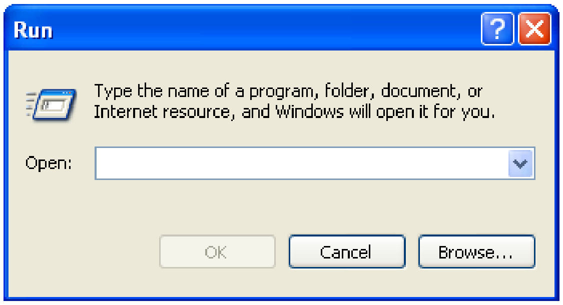
- Opens the Excel application.
- Creates a new workbook using the ThermoCustomReportBaseTemplate.xltm template.
- Passes the parameters to the template.
- Generates a report based on the 1ngrep5_SampleCentricData.xml source file.
- Saves the report to the ThermoCustomReportBaseTemplate.xlsm file.

Note Because you did not map any nodes from the XML Source tree to your spreadsheet, the Excel application will display an error that “no elements have been mapped.” You can click **OK** to dismiss this error and continue to generate an empty report, or you can temporarily map a node before you begin this test. For instructions on adding a mapped node, see “Adding a Temporary Node to the Template (Optional)” on page 27.

❖ To run the test

1. Choose **Start > Run**.

   The Run dialog box opens.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

2. Enter this code in the Open box:

```
Excel.exe /n “C:\Thermo\Custom Report
Tutorials\ThermoCustomReportBaseTemplate.xltm”
/x/C:\Thermo\Custom|Report|Tutorials\Data\1ngrep5_SampleCentricData.x
ml/C:\Thermo\Custom|Report|Tutorials\Reports\MyThermoCustomReportBase
Template.xlsm/s/h
```

   Where:

   - The **/n** switch tells the Windows operating system that you are using this template to create an Excel spreadsheet, not editing the template itself. Without this switch, the Excel application would open the specified template for editing.
   - You must enclose the template file path in quotes when there are spaces in it, for example:

     ```
     "C:\Thermo\Custom Report Tutorials\MyReportName.xltm"
     ```

Otherwise, the Excel application considers each item separated by a space as a separate Excel workbook and tries to load it (causing an error).

- The space before **/x** tells Windows to pass everything after this space to the Excel template. Do not add quotes after this space; otherwise, Windows considers this as a second spreadsheet to open (causing an error).

- The **/x** switch is an internal switch specific to custom reports. The Thermo Custom Report Add-In utility processes all the parameters after **/x**.

- Parameters after **/x** cannot contain spaces. If you need a space, use a vertical bar "|" instead. The vertical bar is replaced with spaces in the add-in. The following example uses the Custom Report Tutorials folder to emphasize this fact:

  ```
  .../x/C:\Thermo\Custom|Report|Tutorials\...
  ```

- Custom parameters after **/x** use a slash "/" as a separator. The first two parameters after **/x** are required; all others are optional.

  - The first parameter after **/x** tells the add-in the path of the XML Source data. The template location does not matter, as long as you correctly specify the path. However, for the TraceFinder application to use the template, the template must be located in the dedicated folder.

  - The second parameter after **/x** tells the add-in the path of the generated Excel spreadsheet.

- The **/s** parameter tells the add-in to save the file (the second parameter) before closing the spreadsheet. Without this switch, the spreadsheet is generated but not saved.

- The **/h** parameter tells the Excel application to hide the screen update. This makes the report generation faster and reduces screen flashes. However, if you want to debug the template, you must remove this switch.

- The Thermo Custom Report Add-In supports additional parameters and controls that affect how reports are generated. For a complete list of parameters, see "How Do I Use the Command Line to Generate the Excel Report?" on page 166.

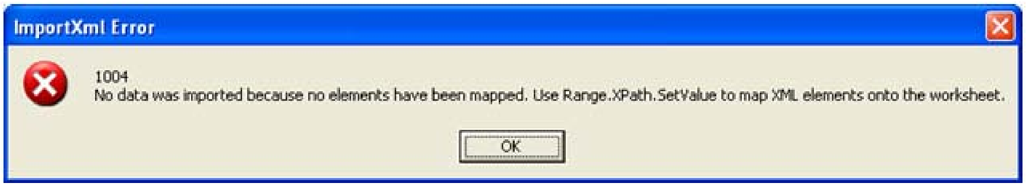
Tip By default, the TraceFinder application file structure results in a long file path. The Windows Run dialog box limits you to 256 characters. To work around this limitation, either copy files to a folder close to the root directory or put the command line into a batch file.

You are now ready to use this base template to create any of the custom report templates in this tutorial:

- Chapter 4, "Creating a Batch Report."
- Chapter 5, "Creating a High Density Sample Report."
- Chapter 6, "Creating a Calibration Report."
- Chapter 7, "Creating a DCC Report."

Adding a Temporary Node to the Template (Optional)

When you tested the template, you probably saw this error:



ImportXml Error

1004
No data was imported because no elements have been mapped. Use Range.XPath.SetValue to map XML elements onto the worksheet.

This error occurred because you did not use any nodes from the XML map in your template. In this optional exercise, you will add a node from the map and generate a report to show that you have mapped elements in your template.

❖ To add a mapped field to the template

1. Click the **Developer** tab to display all the Developer features.

2. Click the **Source** button, Source .

   The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet.

3. From the XML Source tree, drag at least one node to any cell on your spreadsheet.

   You can drag as many nodes as you like.

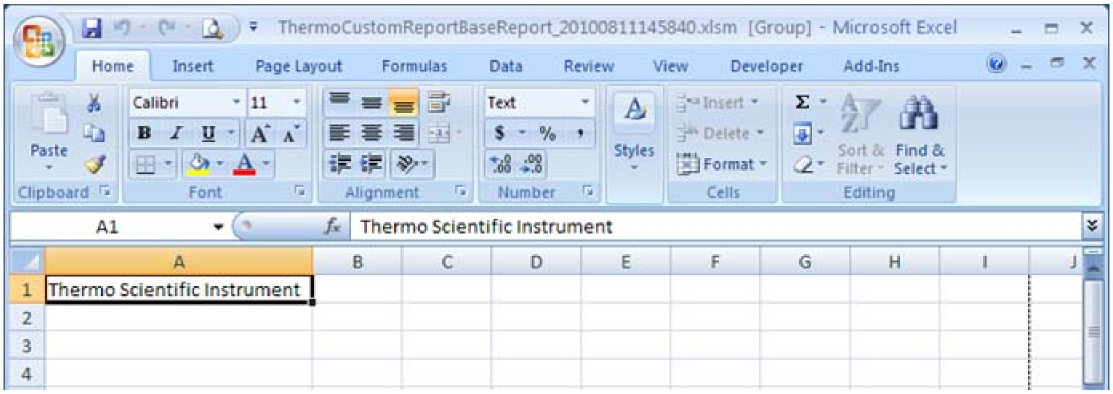
4. Save the template.

   > **IMPORTANT** You must remove these mapped fields from the base template before you use it to create any of the custom reports in this tutorial.

5. Repeat all the instructions in "Testing the Template" on page 25.

The resulting report displays the nodes that you added to the spreadsheet. In this example, the node **InstrumentName** was mapped to cell A1.

Figure 18. Example report



ThermoCustomReportBaseReport_20100811145840.xlsm [Group] - Microsoft Excel

A1 Thermo Scientific Instrument

	A
1	Thermo Scientific Instrument
2	
3	
4	

Note Unlike Crystal Reports in the TraceFinder application, custom reports can have a time stamp in their names. Custom reports do not overwrite existing reports. When a report is generated the first time, no time stamp is added to the file name. If you generate the same report again, the new report will have a time stamp added to its name.

You are now ready to integrate the template into the TraceFinder application to see what a custom report looks like when generated from the application.

❖ To use the template in the TraceFinder application

1. Copy your new template into the dedicated folder of the application:

   C:\Thermo\Shared\Templates\Reports

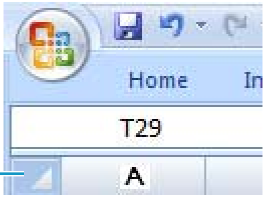
2. Use this template the same way as the other delivered reports in the TraceFinder application.

   For detailed information about how to use custom reports, refer to the *TraceFinder User Guide*.

To make ThermoCustomReportBaseTemplate.xltm the base template in this tutorial, you must remove the temporary mapped fields as follows.

❖ To remove the temporary mapped fields

1. Right-click the upper left corner cell.



2. Choose **Delete** from the shortcut menu.

   The Excel application removes all data on the spreadsheet. Choosing Delete removes the text in cell but does not remove the mapping.

3. Save the template.

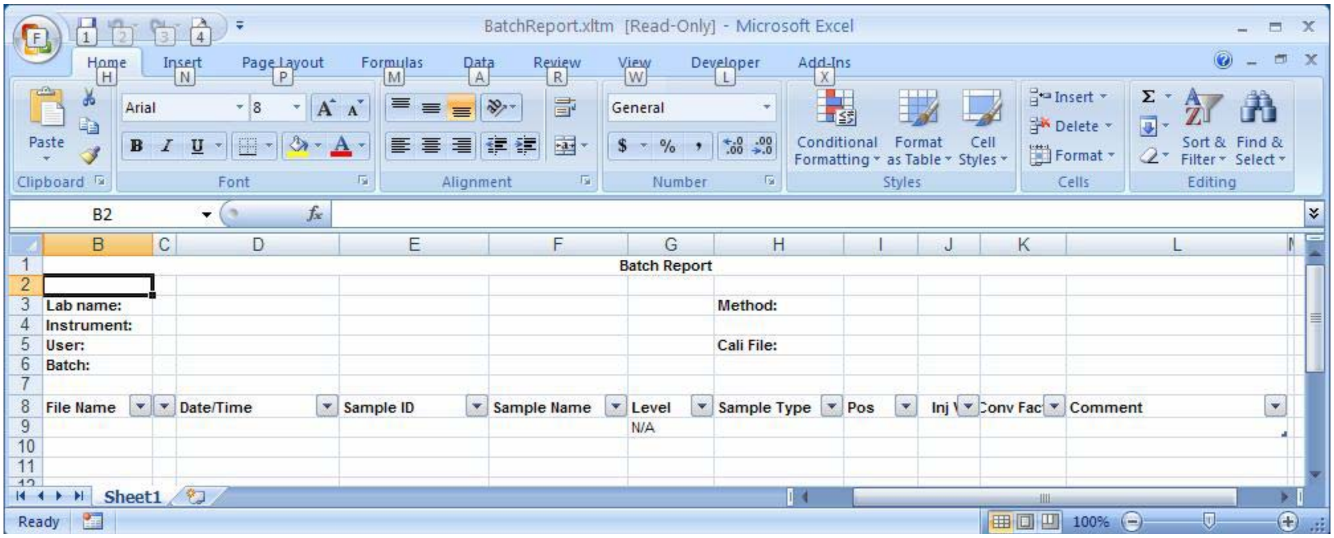
Creating a Batch Report

In this chapter, you will create an Excel template for a batch report similar to the standard Batch Report delivered with the TraceFinder application.

Contents

A Batch Report in the TraceFinder application is a batch-level, summary report that lists all samples and certain properties in a batch.

Figure 19. Sample batch report template



	B	C	D	E	F	G	H	I	J	K	L
1						Batch Report					
2											
3	Lab name:						Method:				
4	Instrument:										
5	User:						Cali File:				
6	Batch:										
7											
8	File Name		Date/Time	Sample ID	Sample Name	Level	Sample Type	Pos	Inj V	Conv Fac	Comment
9						N/A					
10											
11											

A batch report highlights many concepts and principles used for creating custom reports. In this tutorial, you will learn the following:

- How to add static labels
- How to add data fields from the XML Source map
- How to add a list (or table) field from the XML Source map
- How to add and use an empty column in the data table
- How to add and use a hidden column in the data table
- How to add and use named ranges to reference data
- How to use the interface exposed by add-in components and write basic VBA code to connect the template to the add-in
- How to use the **Start > Run** command to generate the report.

Opening the Report Template

In the following procedures, you will use the ThermoCustomReportBaseTemplate.xltm file that you created in Chapter 3, "Getting Started," as the basic template.

❖ To begin your batch report template

In Windows, make a copy of the ThermoCustomReportBaseTemplate.xltm file and name the new copy **MyBatchReport.xltm**.

❖ To load the template file

1. Open the Excel application.
2. Click the **Excel application** icon, , and choose **Open**.
3. In the Open dialog box, double-click **MyBatchReport.xltm**.

Tip Do not open this template file from Windows Explorer, because instead of opening the template for editing, you create a new spreadsheet based on this template.

Adding Labels, Mapped Fields, and Mapped Tables

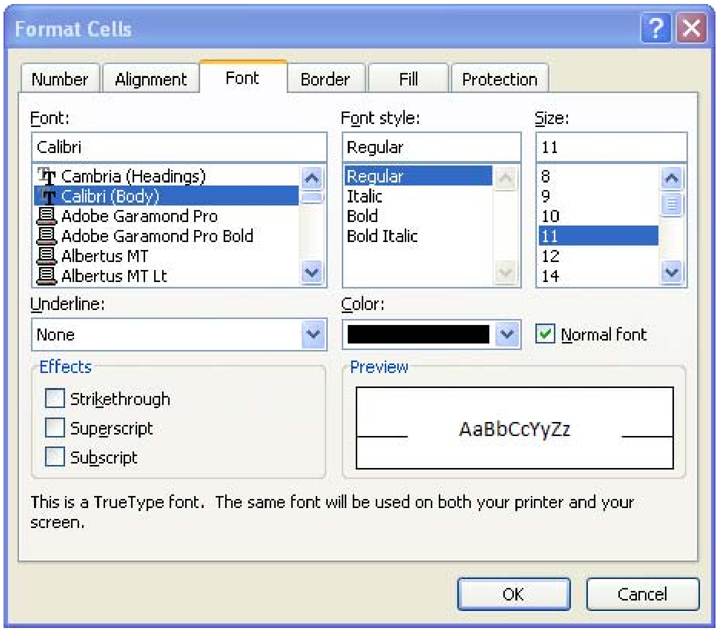
In the following procedures, you will create and name the static column headers in your template, add mapped fields, and add a mapped table.

❖ To enter static labels for the report heading

1. Type **Batch Report** in cell B1.
2. Type **Lab name:** in cell B3.
3. Type **Instrument:** in cell B4.
4. Type **User:** in cell B5.
5. Type **Batch:** in cell B6.
6. Type **Method:** in cell H3.
7. Type **Cali File:** in cell H5.
8. Press **CTRL** and select each of the labeled cells.
9. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

Figure 20. Format Cells dialog box - Font page



Format Cells

Font: Calibri

Font style: Regular

Size: 11

Underline: None

Color:

☑ Normal font

Effects

☐ Strikethrough

☐ Superscript

☐ Subscript

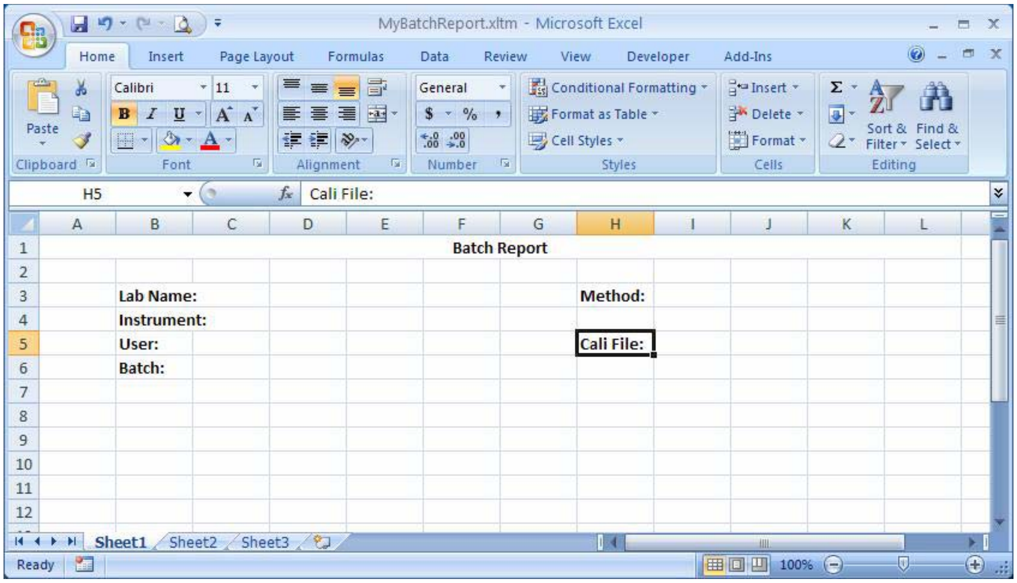
Preview

This is a TrueType font. The same font will be used on both your printer and your screen.

10. Click the **Font** tab.
11. In the Font Style box, select **Bold**.
12. Click **OK** in the Format Cells dialog box.

Ensure that your template looks like the following figure.

Figure 21. Template with static labels



H5 | Cali File:

	A	B	C	D	E	F	G	H
1						Batch Report		
2								
3		Lab Name:						Method:
4		Instrument:						
5		User:						Cali File:
6		Batch:						

Note You start from column B instead of column A because later you will use column A as a hidden column.

Note You can fix any column size issues and fine tune the format at the end, after all the elements are in place.

You are now ready to use the XML Source pane to add mapped fields to your template.

Tip Before proceeding, make sure the Excel window shows the Developer tab, which is hidden by default. If your Excel window does not show the Developer tab, see Chapter 3, “Getting Started.”

❖ To add mapped fields to the template

1. Click the **Developer** tab.

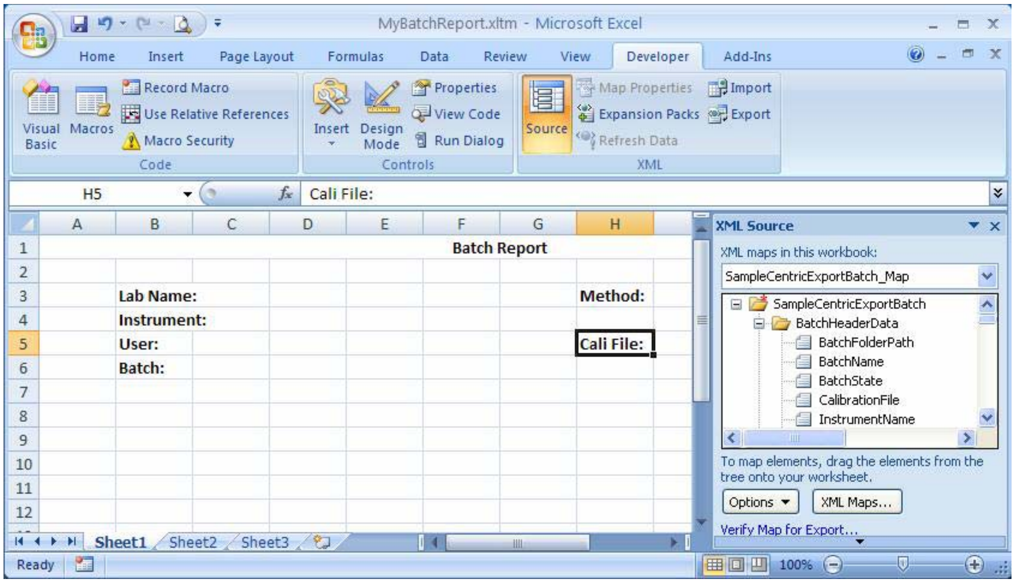
   The ribbon expands to display the Developer features.

2. Click the **Source** button, Source .

   The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet.

Ensure that your template looks like the following figure.

Figure 22. Template with XML Source pane



H5 Cali File:

	A	B	C	D	E	F	G	H
1						Batch Report		
2								
3		Lab Name:						Method:
4		Instrument:						
5		User:						Cali File:
6		Batch:						
7								
8								
9								
10								
11								
12								

XML maps in this workbook:

SampleCentricExportBatch_Map

```
SampleCentricExportBatch
  BatchHeaderData
    BatchFolderPath
    BatchName
    BatchState
    CalibrationFile
    InstrumentName
```

To map elements, drag the elements from the tree onto your worksheet.

Before you begin creating a template for any report, identify the data points on the report and understand how they map to the nodes in the XML Source tree.

Note A “node” in the XML Source tree creates a “field” in the spreadsheet.

You are ready to create mapped fields from the nodes in the SampleCentricExportBatch folder.

3. Open the MethodHeaderData folder and drag the **LabName** node from the XML Source tree to cell C3.

4. Open the MethodHeaderData folder and drag the **InstrumentMethodName** node from the XML Source tree to cell C4.

5. Open the BatchHeaderData folder and drag the **UserName** node from the XML Source tree to cell C5.

6. Open the BatchHeaderData folder and drag the **BatchName** node from the XML Source tree to cell C6.

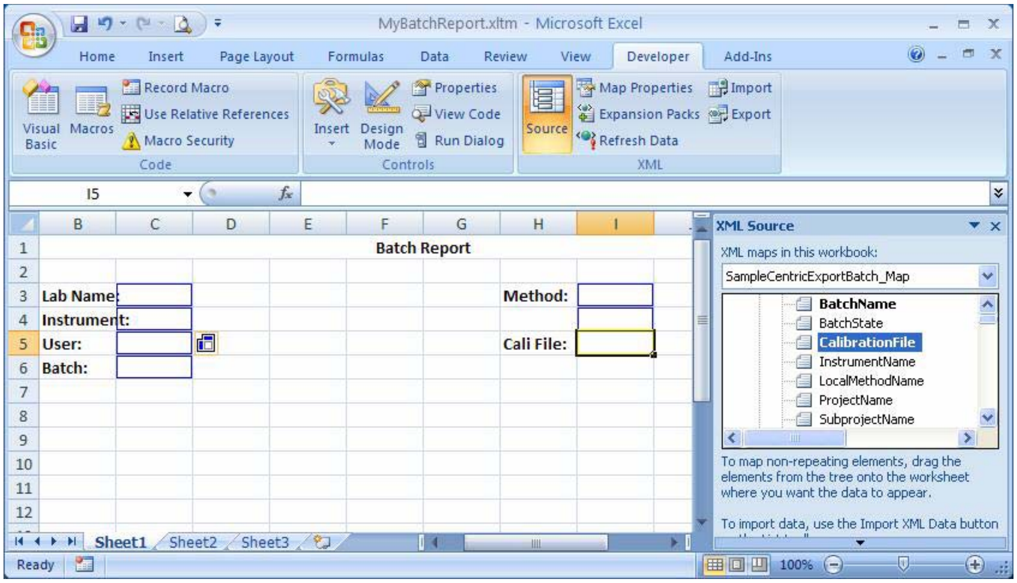
7. Open the MethodHeaderData folder and drag the **MethodName** node from the XML Source tree to cell I3.

8. Open the MethodHeaderData folder and drag the **MasterMethodName** node from the XML Source tree to cell I4.

9. Open the BatchHeaderData folder and drag the **CalibrationFile** node from the XML Source tree to cell I5.

Ensure that your template looks like the following figure.

Figure 23. Template with mapped fields



Batch Report

Lab Name:

Instrument:

User:

Batch:

Method:

Cali File:

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

To import data, use the Import XML Data button

Tip The mapped field on the spreadsheet is shown in a blue box as long as the XML Source pane is visible. The mapped node in the XML Source tree is highlighted in bold.

Note Each node in the XML Source tree can be used only once. In future tutorials where the same field is used more than once, you will work around this limitation by importing the same mapping schema as many times as needed.

You are now ready to create the sample list for the batch. The sample list is presented in a mapped data table. The Excel application lets you drag the mapped table fields from the XML source pane.

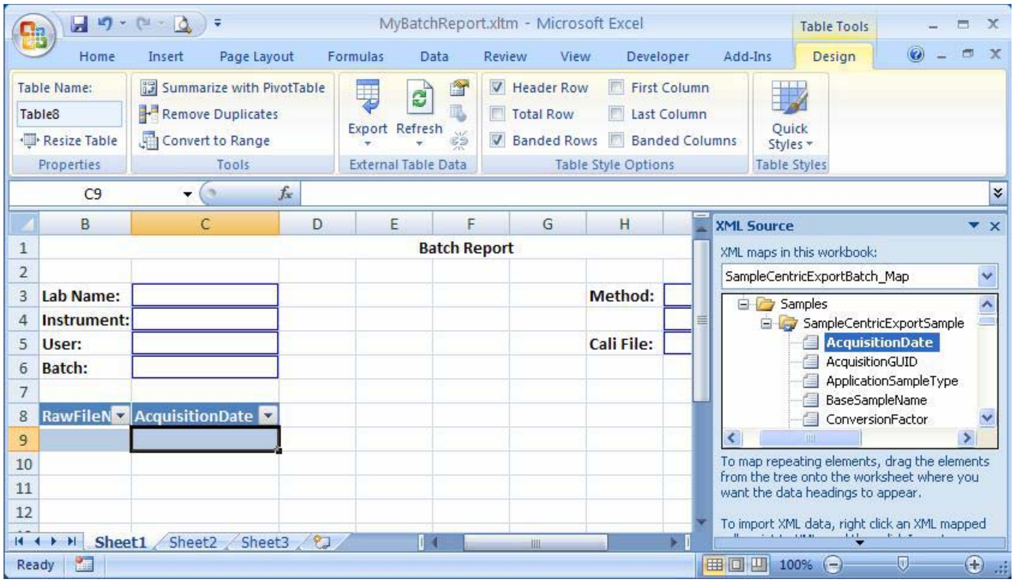
You are ready to create mapped fields from the nodes in the SampleCentricExportBatch/Samples/SampleCentricExportSample folder.

❖ To add a mapped table to the template

1. Drag the **RawFileName** node from the XML Source tree to cell B8.
2. Drag the **AcquisitionDate** node from the XML Source tree to cell C8.

Ensure that your template looks like the following figure.

Figure 24. Template with mapped fields



To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped

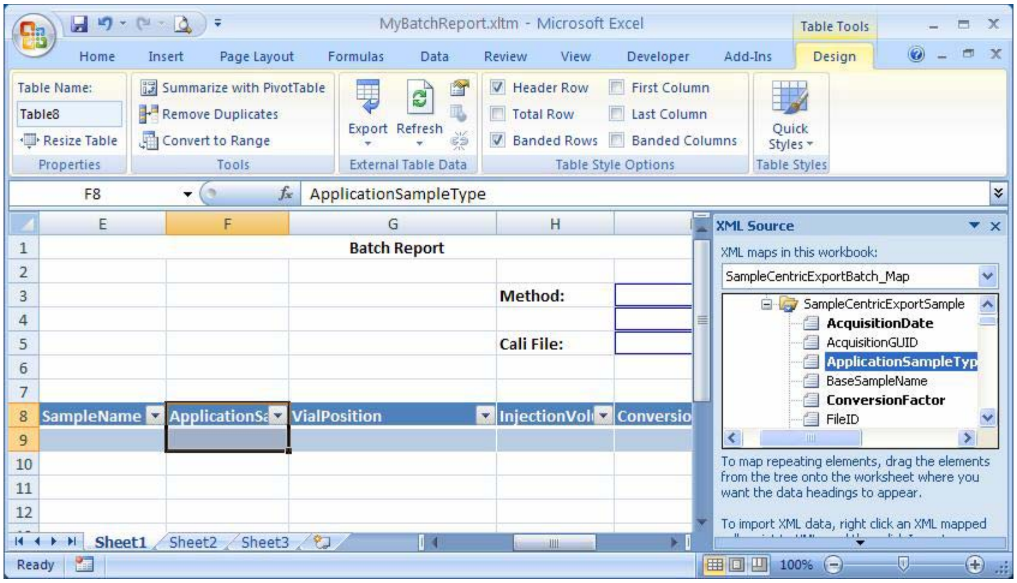
Note The spreadsheet displays mapped fields and mapped tables (a list) differently. You can see this difference in the XML Source tree: A regular node has a regular folder icon, 📂. A repeating or collection container node has a folder icon with an extra small arrow pointing down, 📂. For example, in a batch, there is only one set of BatchHeaderData (regular node), but you can have many samples and each SampleCentricExportSample (repeating node) represents one of them.

You can now create additional mapped fields from the nodes in the SampleCentricExportBatch/Samples/SampleCentricExportSample folder.

3. Drag the **SampleId** node from the XML Source tree to cell D8.
4. Drag the **SampleName** node from the XML Source tree to cell E8.
5. Drag the **ApplicationSampleType** node from the XML Source tree to cell F8.
6. Drag the **VialPosition** node from the XML Source tree to cell G8.
7. Drag the **InjectionVolume** node from the XML Source tree to cell H8.
8. Drag the **ConversionFactor** node from the XML Source tree to cell I8.
9. Drag the **SampleComments** node from the XML Source tree to cell J8.

Ensure that your template looks like the following figure.

Figure 25. Template with mapped fields and tables



F8 | ApplicationSampleType

	E	F	G	H	
1			Batch Report		
2					
3				Method:	
4					
5				Cali File:	
6					
7					
8	SampleName	ApplicationSa	VialPosition	InjectionVol	Conversio
9					
10					
11					
12					

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped

This template, compared to the final one, has two columns that are missing. The final template shows an empty column at C and a Level column at G. Those two columns are important concepts in creating the custom reports and will be explained in the next two procedures.

Adding Empty Columns

When you use the Excel application as reporting tool, you might need to adjust the size of columns. In a spreadsheet, a column must run from top to bottom with the same width, because the Excel application does not support the concept of sections. Sometimes, however, you need more space only for some rows but not others.

If a data table is not involved, you can make columns smaller and leave space between them as needed. However, when a data table is on the same sheet, you cannot use this method. For more details, see "Why Do So Many Data Tables Appear on My Spreadsheet?" on page 182.

As a workaround, you can add empty columns to the data table. An empty column can provide the extra space needed for the previous column, because the text in the previous column can run over into it.

In this sample list, the RawFileName column needs some more space. However, you do not want to make column B too wide because you do not want to leave too much space between the label and the mapped field from B3 to C6. Adding an empty column next to the RawFileName column resolves this problem.

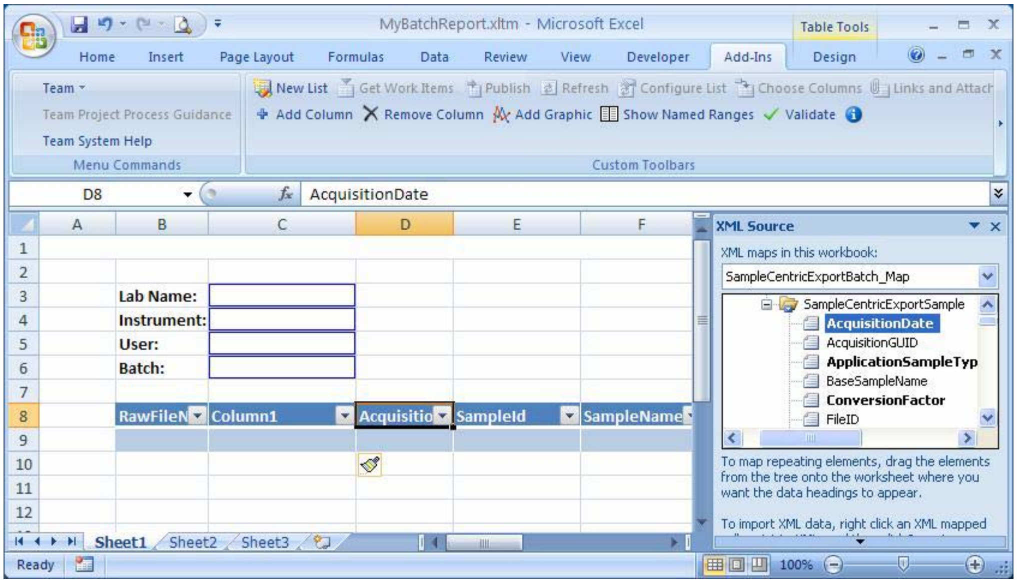
❖ To add an empty column to the template

Right-click cell C8 and choose **Insert > Table Columns to the Left** from the shortcut menu.

The application inserts an empty column at column C and names the column **Column1**.

Ensure that your template looks like the following figure.

Figure 26. Template with empty column



To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped

Normally, you must add hidden columns because you cannot use the XML source data field directly in a linked table.

Adding Derived Columns

You are ready to hide the mapped field in the first column and then use the Excel formula to fill in the Level column. Now you want to make Column1 invisible. You cannot simply delete it, because Excel requires that the column header be unique and not empty. A space character is allowed as a valid header, so you can change “Column1" to a space character to hide it.

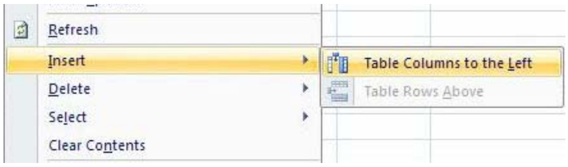
❖ To add a derived column to the template

1. Open the SampleCentricExportBatch/Samples/SampleCentricExportSample folder and drag the **SampleLevel** node from the XML Source tree to cell A8.

   The SampleLevel field has a value for the calibration standard sample type and is empty for the non-calibration standard sample type. However, displaying SampleLevel in a report requires the level column to show N/A instead of a empty field if the sample type is not a calibration standard. Therefore, the SampleLevel field cannot be used directly.

   The workaround for this problem is to make SampleLevel a hidden column and add a derived column for display that uses an Excel formula to evaluate SampleLevel.

2. Right-click the ApplicationSampleType cell and choose **Insert > Table Columns to the Left** from the shortcut menu.

   The application inserts an empty column **G** before the ApplicationSampleType column and inserts the header name **Column2** into cell G8.

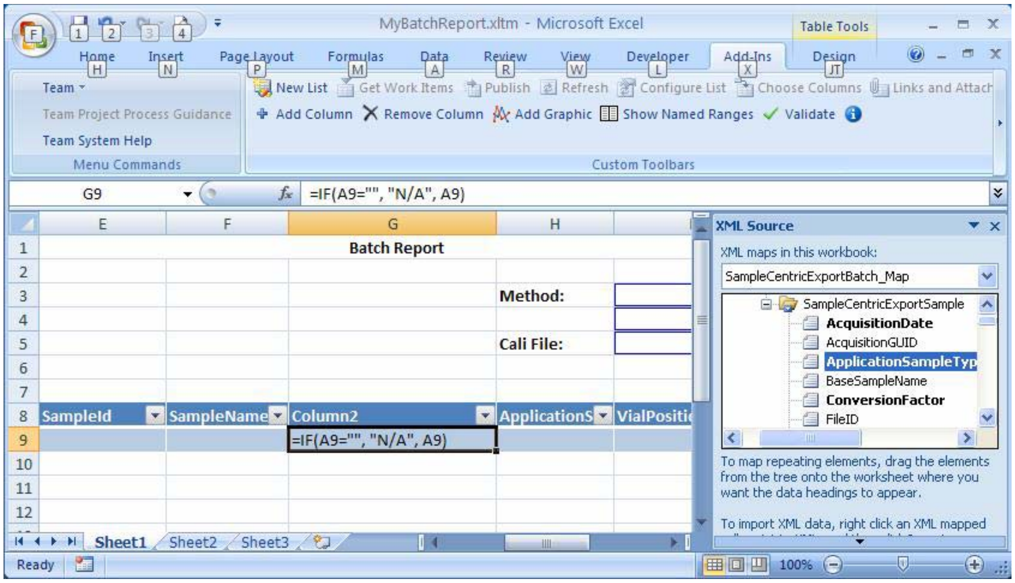
3. In cell G9, type the following formula:

   ```
   =IF(A9="", "N/A", A9)
   ```

   This tells the table to show “N/A” when the SampleLevel at A9 is empty, and otherwise, to show the SampleLevel value. You enter this formula at G9 only, but it automatically expands to all the cells in this column when you import the data into this table.

Ensure that your template looks like the following figure.

Figure 27. Template with formula referencing a hidden column



G9 =IF(A9="", "N/A", A9)

	E	F	G	H	
1			Batch Report		
2					
3				Method:	
4					
5				Cali File:	
6					
7					
8	SampleId	SampleName	Column2	ApplicationS	VialPositi
9			=IF(A9="", "N/A", A9)		
10					
11					
12					

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped

4. Change the cell format to the General type to use this column as a formula.

   When you add an empty column, the cell type defaults to the text type. The text type column does not expand when you import data. For more information, see "Why Does My Formula in a Data Table Not Expand?" on page 191.

5. Right-click the column A header and choose **Hide** from the shortcut menu.

> **Tip** You can hide any column in the Excel application. However, if you need hidden columns for the TraceFinder custom reports, always use the first columns. A hidden column in the middle of the table can cause trouble when copying blocks of data for the repeating area.

Completing the Layout

The following procedures show you how to rename the default table header names, name the mapped table, and finish the layout.

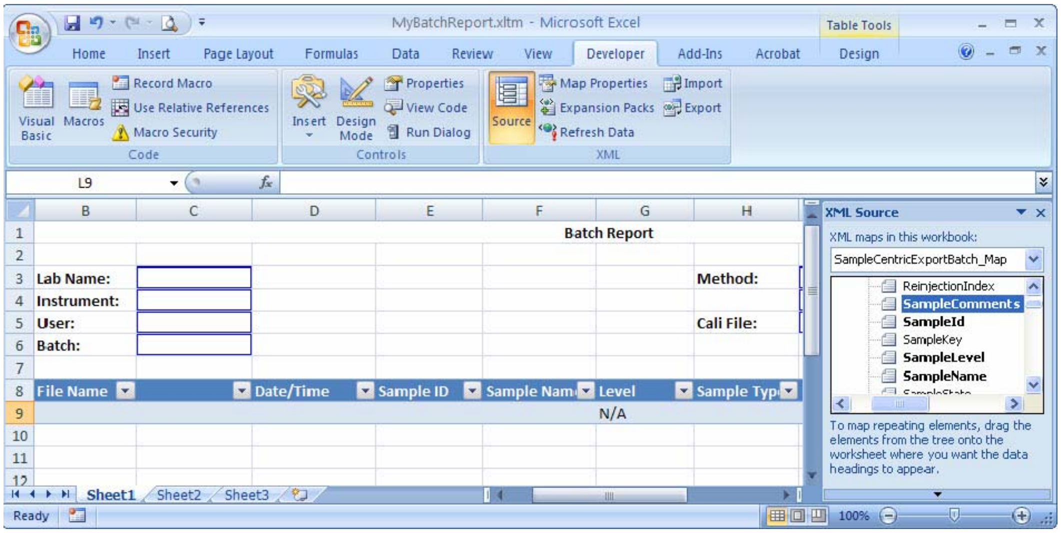
By default, when a node is dropped on the spreadsheet, the node name becomes the default header name. You are now ready to rename the default header names. In the following procedures, you will rename the default table header name.

❖ To rename mapped table header names in the template

1. Change B8 from RawFileName to **File Name**.
2. Change C8 from Column1 to “ ” (one space).
3. Change D8 from AcquisitionDate to **Date/Time**.
4. Change E8 from SampleId to **Sample ID**.
5. Change F8 from SampleName to **Sample Name**.
6. Change G8 from Column2 to **Level**.
7. Change H8 from ApplicationSampleType to **Sample Type**.
8. Change I8 from VialPosition to **Pos**.
9. Change J8 from InjectionVolume to **Inj Vol**.
10. Change K8 from ConversionFactor to **Conv Factor**.
11. Change L8 from SampleComments to **Comment**.

Ensure that your template looks like the following figure.

Figure 28. Template with custom header text



	B	C	D	E	F	G	H
1					Batch Report		
2							
3	Lab Name:						Method:
4	Instrument:						
5	User:						Cali File:
6	Batch:						
7							
8	File Name		Date/Time	Sample ID	Sample Nam	Level	Sample Typ
9						N/A	
10							
11							
12							

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

You are now ready to use the Excel named range feature to name the table in the template. You can name a range of cells and then use the name to refer to this area.

If there is no data to display in custom reports, “No data in this report” appears on the report. In this case, you also want to hide the data table completely. Two add-in functions can implement this functionality.

There are two ways to refer to cells in VBA code: you can use direct cell ranges (for example, Sheet1!A5:B6) or you can use named ranges. However, using named ranges is preferred because it provides more flexibility in design.

In the following procedures, you will define two named ranges. First, define a named range called "NoData" for cells where "No data in this report" is to be displayed. You will use an add-in function to display this message.

Second, Excel automatically gives each data table a name, for example, Table1. In order to reference the table in the VBA code, change it to a specific name.

❖ To name the data table in the template

1. Click the **Formulas** tab and then click **Name Manager**.

   > **Note** The Excel application automatically creates a name for all tables, such as Table1 or Table2. Each time you create a new table, it increments the table name, even if you deleted a previous one.

2. In the Name Manager dialog box, double-click **Table1**.

   The table name might vary. Use the sheet, row, and column numbers in the Refers To column to verify the table.

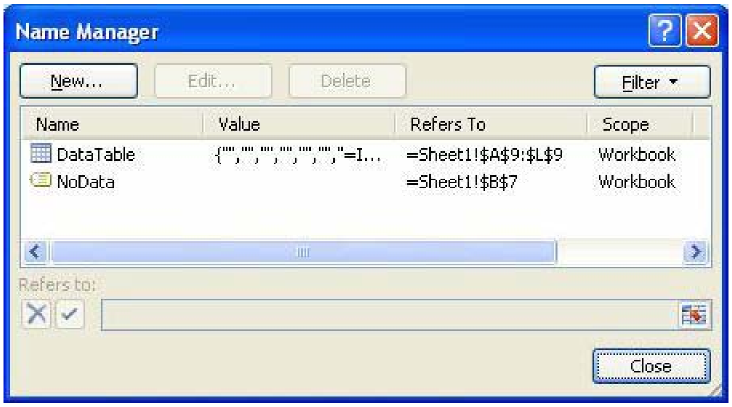
   The Edit Name dialog box opens.

3. Change the default name to **DataTable**.

4. Click **OK** in the Edit Name dialog box.

You now have two names defined, as shown in this figure:

Figure 29. Name Manager dialog box



Name	Value	Refers To	Scope
DataTable	{"","","","","","","=I...	=Sheet1!$A$9:$L$9	Workbook
NoData		=Sheet1!$B$7	Workbook

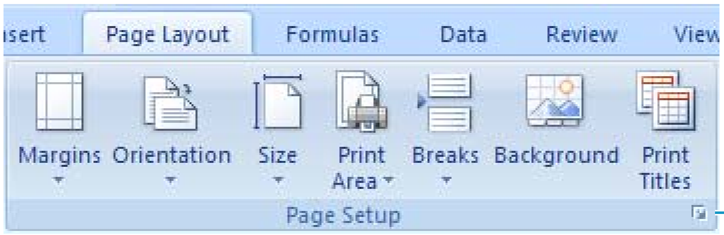
You will later use VBA code to show "No data in the report" at cell B7 when the table is empty. See "Adding VBA Code" on page 46.

5. Click **Close** in the Name Manager dialog box.

You are now ready to polish the layout of the template.

❖ To resize the columns in the template

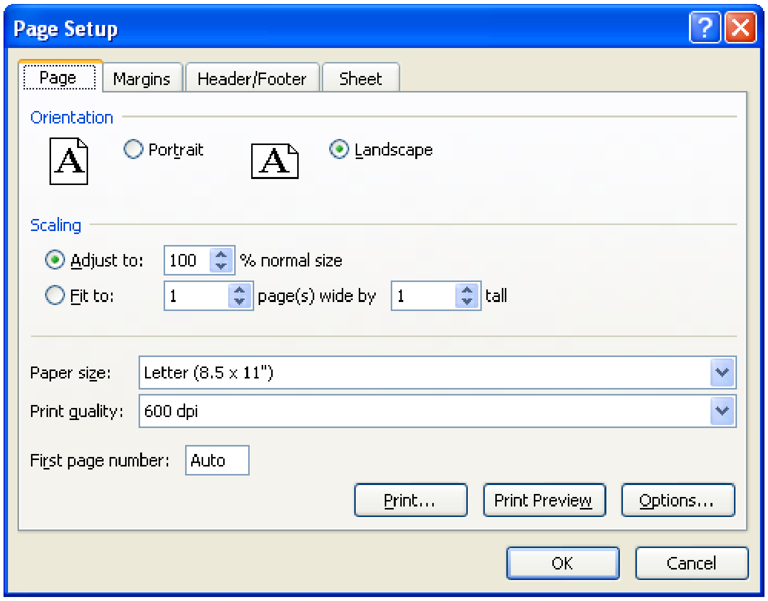
1. Click the **Page Layout** tab to display the layout editing features.



Click to open Page Setup dialog box

2. Click the arrow in the lower-right corner to open the Page Setup dialog box.

Figure 30. Page Setup dialog box



Orientation

○ Portrait ● Landscape

Scaling

● Adjust to: 100 % normal size

○ Fit to: 1 page(s) wide by 1 tall

Paper size: Letter (8.5 x 11")

Print quality: 600 dpi

First page number: Auto

3. On the Page Setup dialog box, make the following changes:
   a. Click the **Page** tab and change the orientation from Portrait to **Landscape**.

      For a Batch Report, the table is too wide to fit into portrait format.

   b. Click the **Margins** tab and change the page margins to the following (all units are in inches):

      Top: **0.65**, Left: **0.25**, Right: **0.25**, Bottom: **0**, Header: **0.3**, Footer: **0.3**

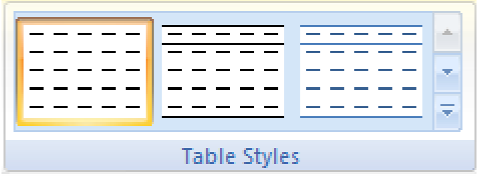
   c. Click the **Header/Footer** tab and select the **Align with Page Margins** check box.
   d. Click **OK** in the Page Setup dialog box.
4. Drag the dividers in the column headings to make all the columns fit within the page range.

   When you are in page preview mode, the right margin of the page is indicated by dashed line.

❖ To change the background in the template

1. Click the header of the data table to display the Design tab.
2. Click the **Design** tab.

   The Design tab displays the Table Styles functions.

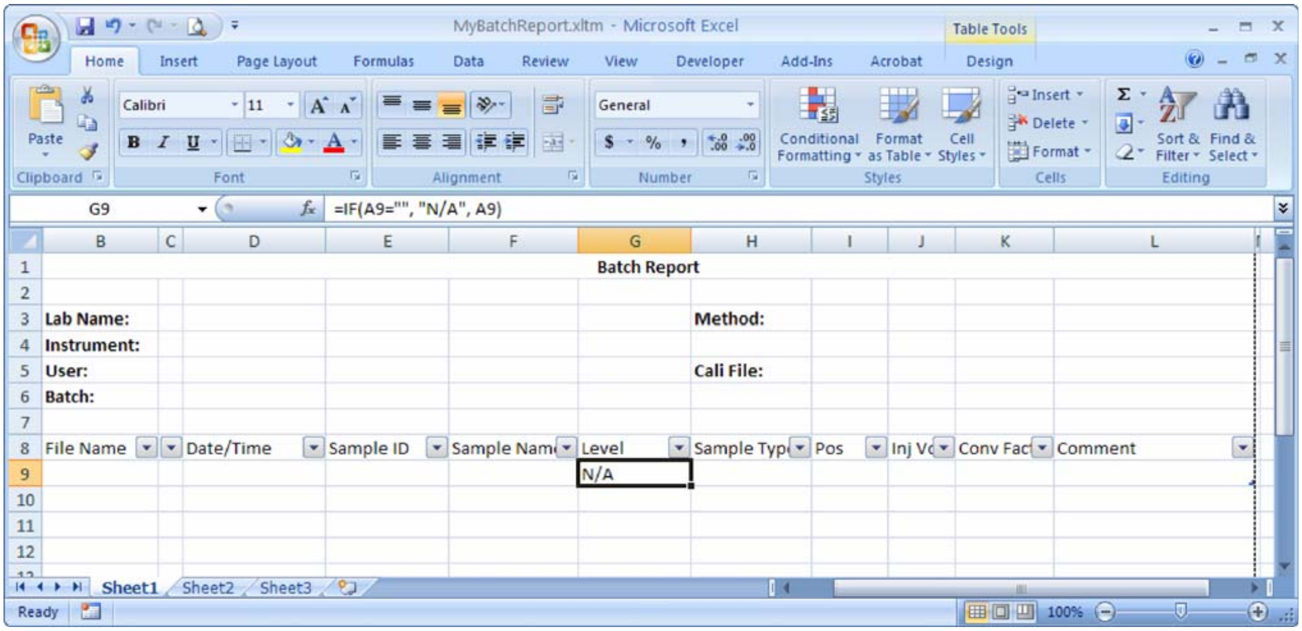


3. Expand the Table Styles gallery and click **Clear**, Clear , in the lower left corner of the gallery.

   The background changes to a plain background.

Ensure that your template looks like the following figure.

Figure 31. Column width example with plain background



G9 =IF(A9="", "N/A", A9)

	B	C	D	E	F	G	H	I	J	K	L
1						Batch Report					
2											
3	Lab Name:						Method:				
4	Instrument:										
5	User:						Cali File:				
6	Batch:										
7											
8	File Name		Date/Time	Sample ID	Sample Nam	Level	Sample Typ	Pos	Inj Vo	Conv Fac	Comment
9						N/A					
10											
11											
12											

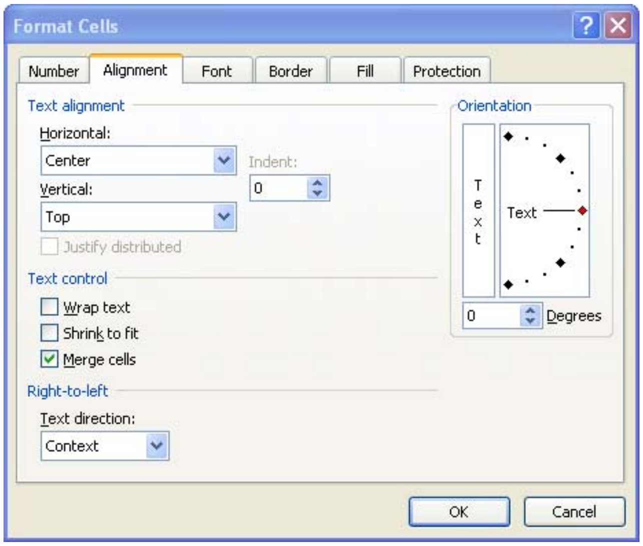
4. To prevent the text in the last column of the page from running over its boundary and creating a new page, do the following:

   a. Select the last column.

   b. Right-click and choose **Format Cells** from the shortcut menu.

      The Format Cells dialog box opens.

Figure 32. Format Cells dialog box - Alignment page



Horizontal: Center

Indent: 0

Vertical: Top

□ Justify distributed

□ Wrap text

□ Shrink to fit

☑ Merge cells

Text direction: Context

0 Degrees

c. Click the **Alignment** tab.

d. In the Text Control area, select **Wrap Text**.

e. Click **OK**.

The template design is complete. You are now ready to add VBA code to connect the template to the data.

Adding VBA Code

For this Batch Report, you will add two functions using VBA code. These functions do the following:

- Verifies that the table named **DataTable** exists.
- If there is no data in the table, displays “No data in this report” in the name range defined as **NoData**.

❖ To connect the VBA data to the template

Add the following code to Module1:

```
Option Explicit

Public Sub ProcessData()
    If Sheet1.ListObjects("DataTable").Range.Rows.Count = 0 Then
        mUtil.SetNoData "NoData"
        mUtil.HideRows "DataTable" 'Hide all data area
    End If
End Sub

Public Function Validate(Optional showWarning As Boolean = True) As
Boolean
    Dim err_msg As String, info_msg As String, msg As String

   ' Check required names
   mUtil.CheckRequiredName err_msg, "DataTable", "'DataTable' must
be defined."

   ' Display messages if any
    mUtil.ShowErrMsg err_msg, info_msg, showWarning

    Validate = err_msg = ""
End Function
```

Where:

- IUtil_Validate() in MyUtil class calls the Sub **Validate()**.

  Before the source XML data is imported, IUtil_Validate() verifies that named ranges are correctly defined. For example, if you add a column to a table or remove a column from a table, you must update the associated named range so that the other functions depending on this name range still work.

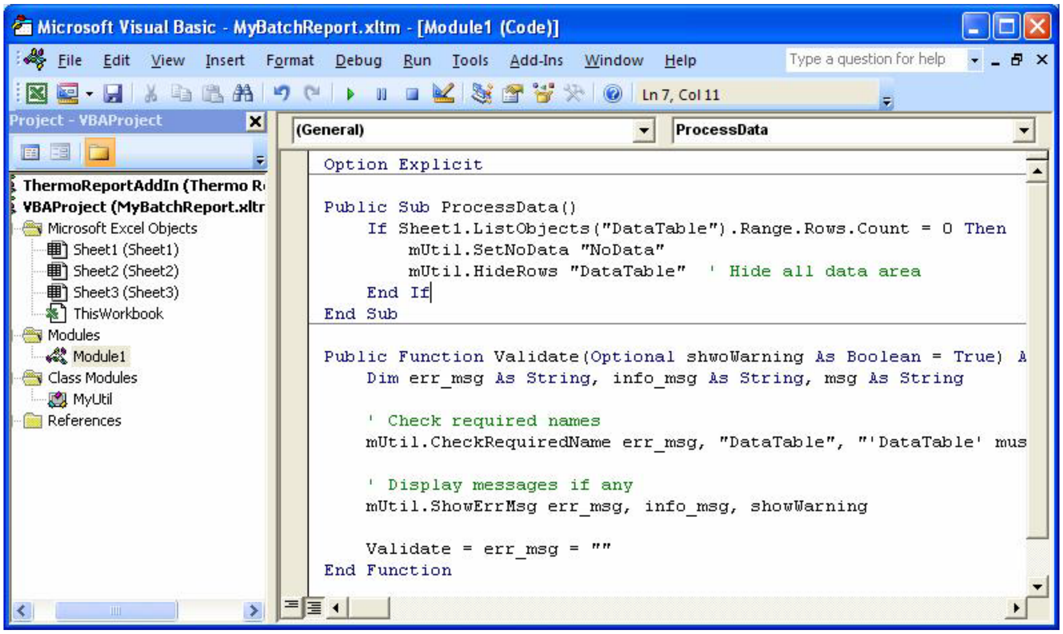
- Workbook_Open() in ThisWorkbook calls Sub **ProcessData()**.

  Workbook_Open() is called after the source XML file is imported. In the Workbook_Open() code, the application checks if the table DataTable has any rows. If not, it sets “No data in this report” to the name range NoData using the SetNoData function defined in the add-in. In this case, it hides the DataTable rows using the HideRows add-in function.

5. To use the Visual Basic compile function to check for errors, choose **Debug > Compile VBAProject**.

Your Visual Basic code should look like the following figure.

Figure 33. VBA data



```
Option Explicit

Public Sub ProcessData()
    If Sheet1.ListObjects("DataTable").Range.Rows.Count = 0 Then
        mUtil.SetNoData "NoData"
        mUtil.HideRows "DataTable"  ' Hide all data area
    End If
End Sub

Public Function Validate(Optional shwoWarning As Boolean = True) A
    Dim err_msg As String, info_msg As String, msg As String

    ' Check required names
    mUtil.CheckRequiredName err_msg, "DataTable", "'DataTable' mus

    ' Display messages if any
    mUtil.ShowErrMsg err_msg, info_msg, showWarning

    Validate = err_msg = ""
End Function
```

You are now ready to use your template to generate a report.

Testing the Template

You are now ready to run a test from the Windows **Start** > **Run** box. In this test, the code you enter does the following:

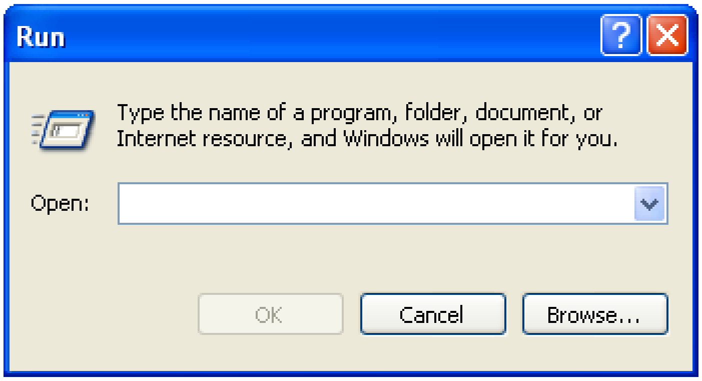
- Opens the Excel application.
- Creates a new workbook using the MyBatchReport.xltm template.
- Passes the parameters to the template.
- Generates a report based on the 1ngrep5_SampleCentricData.xml source file.
- Saves the report to the MyBatchReport.xlsm file.

❖ To run the test

1. Choose **Start** > **Run**.

   The Run dialog box opens.

   The Run dialog box opens.

   Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

   Open:

   Tip By default, the TraceFinder application file structure results in a long file path. The Windows Run dialog box limits you to 256 characters. To work around this limitation, either copy files to a folder close to the root directory or put the command line into a batch file.

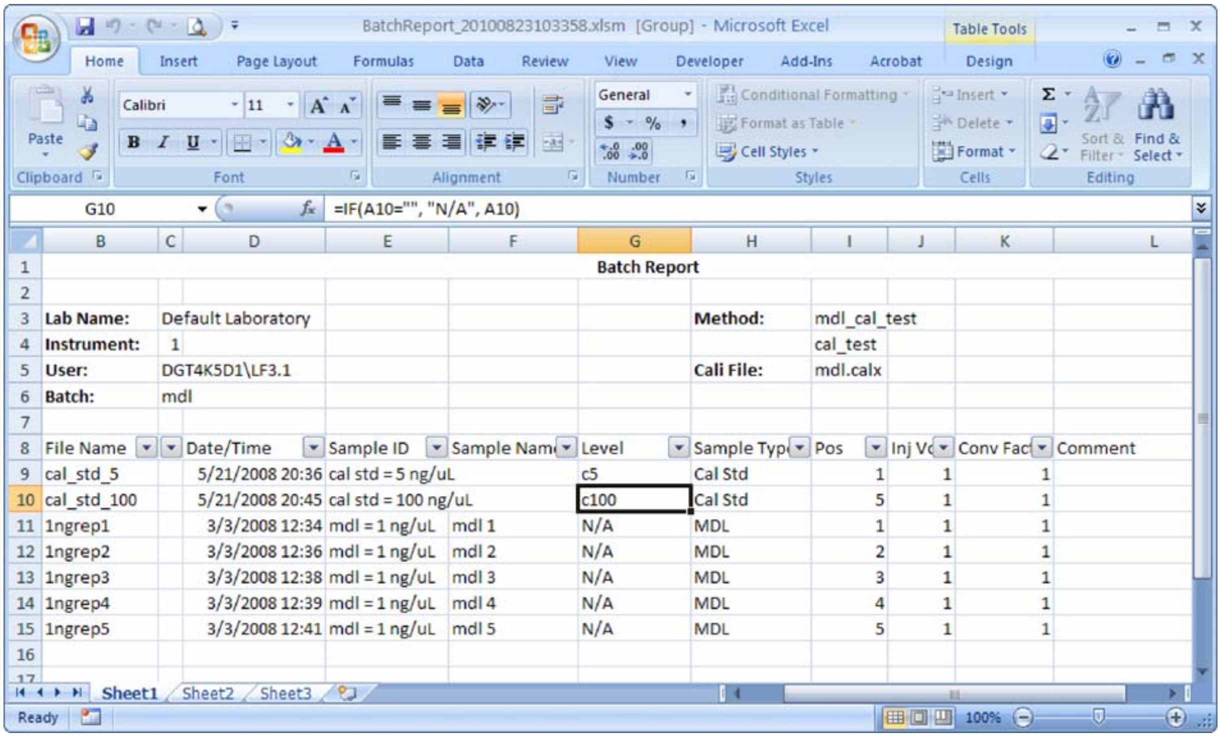
2. Enter this code in the Open box:

   ```
   Excel.exe /n “C:\Thermo\Custom Report Tutorials\MyBatchReport.xltm”
   /x/C:\Thermo\Custom|Report|Tutorials\Data\1ngrep5_SampleCentricData.x
   ml/C:\Thermo\Custom|Report|Tutorials\Reports\BatchReport.xlsm/s/h
   ```

   For detailed descriptions of the parameters used in this code, see “Testing the Template” on page 25.

The generated report should appear on your screen as in the following figure (actual data may vary, depending on the batch that you used).

Figure 34. Generated report



G10 =IF(A10="", "N/A", A10)

	B	C	D	E	F	G	H	I	J	K	L
1						Batch Report					
2											
3	**Lab Name:**	Default Laboratory					**Method:**	mdl_cal_test			
4	**Instrument:**	1						cal_test			
5	**User:**	DGT4K5D1\LF3.1					**Cali File:**	mdl.calx			
6	**Batch:**	mdl									
7											
8	File Name		Date/Time	Sample ID	Sample Nam	Level	Sample Typ	Pos	Inj Vo	Conv Fac	Comment
9	cal_std_5		5/21/2008 20:36	cal std = 5 ng/uL		c5	Cal Std	1	1	1	
10	cal_std_100		5/21/2008 20:45	cal std = 100 ng/uL		c100	Cal Std	5	1	1	
11	1ngrep1		3/3/2008 12:34	mdl = 1 ng/uL	mdl 1	N/A	MDL	1	1	1	
12	1ngrep2		3/3/2008 12:36	mdl = 1 ng/uL	mdl 2	N/A	MDL	2	1	1	
13	1ngrep3		3/3/2008 12:38	mdl = 1 ng/uL	mdl 3	N/A	MDL	3	1	1	
14	1ngrep4		3/3/2008 12:39	mdl = 1 ng/uL	mdl 4	N/A	MDL	4	1	1	
15	1ngrep5		3/3/2008 12:41	mdl = 1 ng/uL	mdl 5	N/A	MDL	5	1	1	
16											

You are now ready to integrate the template into the TraceFinder application.

❖ To use the template in the TraceFinder application

1. Copy your new template into the dedicated folder of the application:

   C:\Thermo\Shared\Templates\Reports

2. Use this template the same as the delivered reports in the TraceFinder application.

   For detailed information about how to use custom reports, refer to the *TraceFinder User Guide* or Chapter 3, “Getting Started.”

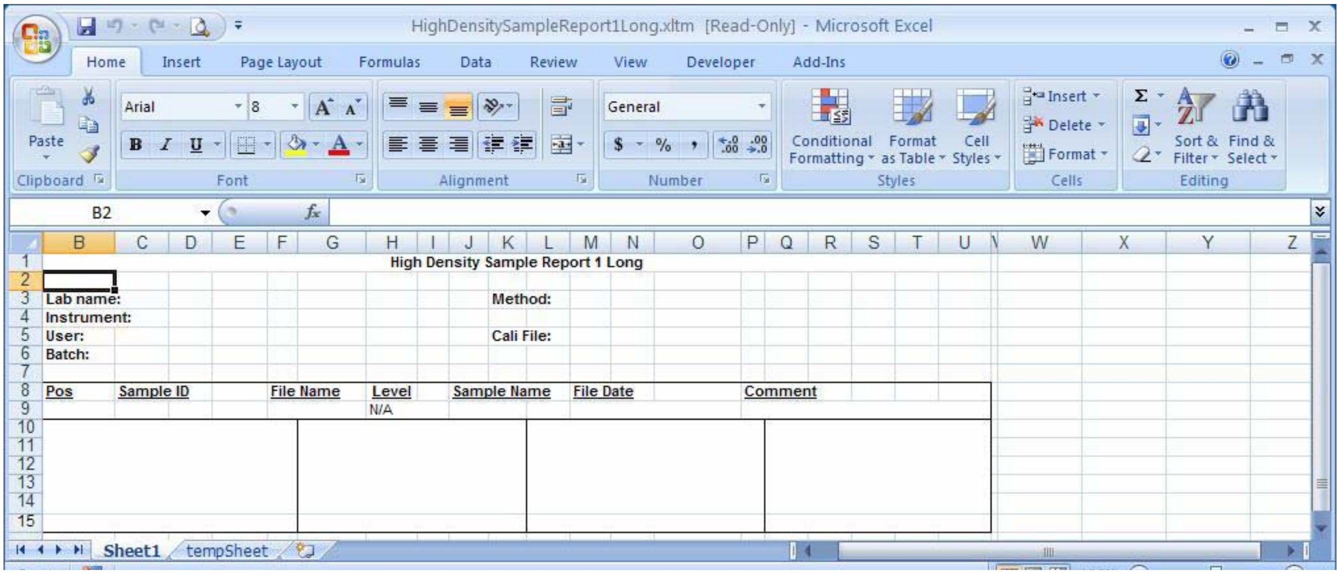
Creating a High Density Sample Report

In this chapter, you will create an Excel template for a high density sample report similar to the High Density Sample Report 1 Long delivered with the TraceFinder application.

Contents

A high density sample report in the TraceFinder application is a sample-level report that presents chromatograms of quantitative peaks and related results for each compound in a highly condensed format.

Figure 35. Sample high density report template



HighDensitySampleReport1Long.xltm [Read-Only] - Microsoft Excel

B2

High Density Sample Report 1 Long

Lab name: Method:

Instrument:

User: Cali File:

Batch:

Pos	Sample ID	File Name	Level	Sample Name	File Date	Comment
			N/A			

As you create a high-density sample report template, you will learn the following:

- How to use a data table on a temporary sheet
- How to add images to a report
- How to use a repeating area as a design interface that is duplicated at run-time
- How to use formulas with the data table to combine or reformat values
- How to duplicate sheets to create multi-sheet reports
- How to use the **Start > Run** command to generate the report

Opening the Report Template

In the following procedures, you will use the ThermoCustomReportBaseTemplate.xltm file that you created in the Getting Started section as the basic template.

❖ To begin your high density report template

In Windows, make a copy of the ThermoCustomReportBaseTemplate.xltm file and name the new copy **MyHighDensitySampleReport1Long.xltm**.

❖ To load the template file

1. Open the Excel application.
2. Click the **Excel application** icon, , and choose **Open**.
3. In the Open dialog box, double-click **MyHighDensitySampleReport1Long.xltm**.

Tip Do not open this template file from Windows Explorer, because instead of opening the template for editing, you create a new spreadsheet based on this template.

Adding Labels and Mapped Nodes

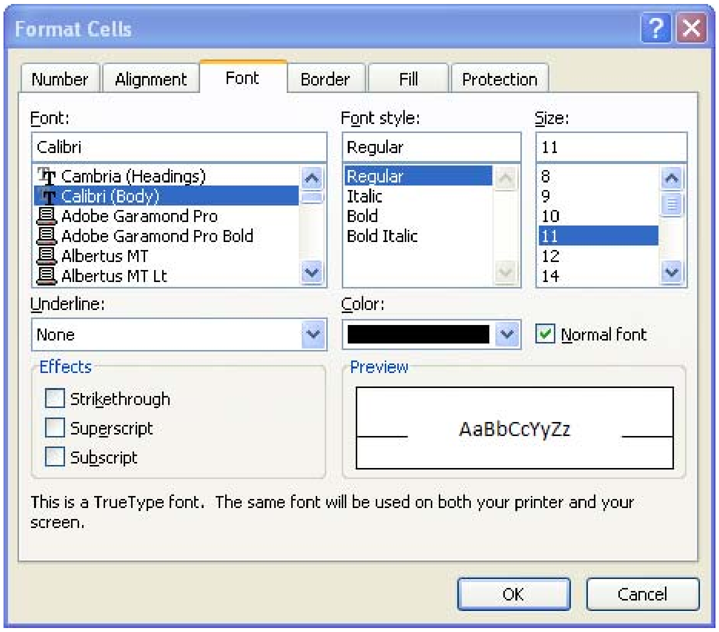
You are now ready to create and name the static column headers in your template.

❖ To enter static labels for the report heading

1. Type the report header **High Density Sample Report 1 Long** in cell B1.
2. Type **Lab name:** in cell B3.
3. Type **Instrument:** in cell B4.
4. Type **User:** in cell B5.
5. Type **Batch:** in cell B6.
6. Type **Method:** in cell K3.
7. Type **Cali File:** in cell K5.
8. Select cells **B1** through **K5**.
9. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

   Figure 36. Format Cells dialog box



Font: Calibri

Font style: Regular

Size: 11

Underline: None

Color:

☑ Normal font

☐ Strikethrough

☐ Superscript

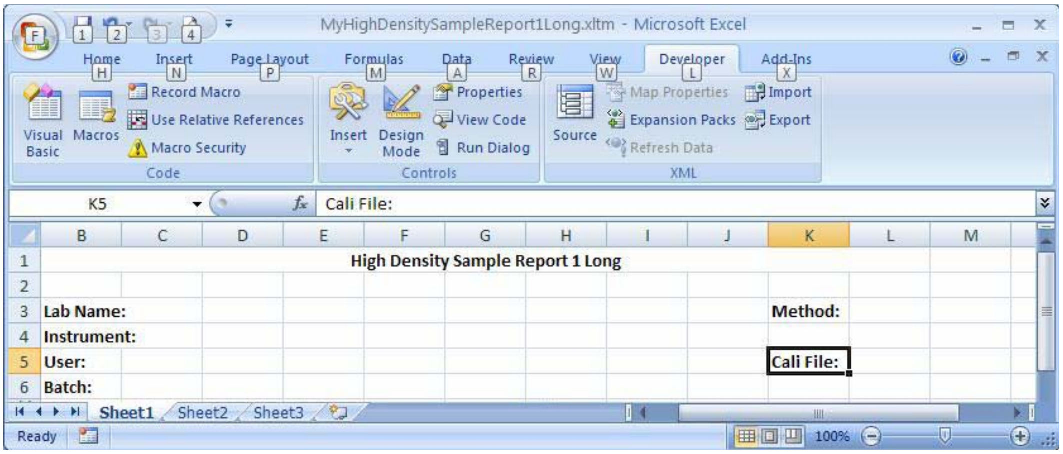
☐ Subscript

This is a TrueType font. The same font will be used on both your printer and your screen.

10. Click the **Font** tab.
11. In the Font Style box, select **Bold**.
12. Click **OK** in the Format Cells dialog box.

Ensure that your template looks like the following figure.

Figure 37. Template with static labels



K5 | Cali File:

	B	C	D	E	F	G	H	I	J	K	L	M
1					High Density Sample Report 1 Long							
2												
3	Lab Name:									Method:		
4	Instrument:											
5	User:									Cali File:		
6	Batch:											

You are now ready to display the XML Source pane.

Tip Before proceeding, make sure the Excel window shows the Developer tab, which is hidden by default. If your Excel window does not show the Developer tab, see Chapter 3, "Getting Started."

❖ To display the XML Source pane

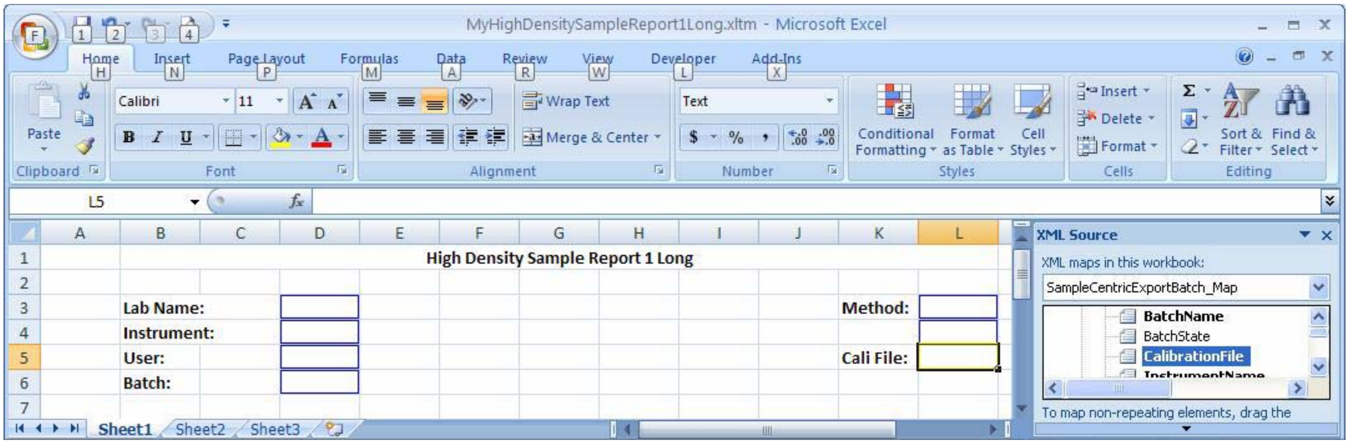
1. Click the **Developer** tab to display all the Developer features.
2. Click the **Source** button, Source.

   The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet.

Before you begin creating a template for any report, identify the data points on the report and understand how they map to the nodes in the XML Source.

Ensure that your template looks like the following figure.

Figure 38. Template with XML Source pane



	A	B	C	D	E	F	G	H	I	J	K	L
1						High Density Sample Report 1 Long						
2												
3		Lab Name:									Method:	
4		Instrument:										
5		User:									Cali File:	
6		Batch:										
7												

The file list at the sample level contains one line only, but the table format provides better readability. Most of the TraceFinder sample-level reports use this kind of file list as part of report header.

You are now ready to add a file list that the application uses as part of the report header. To create this file list, you will add static labels and then drag nodes from the XML Source pane.

❖ To enter static labels for the column headers

1. Type **Pos** in cell B8.
2. Type **SampleID** in cell C8.
3. Type **File Name** in cell F8.
4. Type **Level** in cell H8.
5. Type **Sample Name** in cell J8.
6. Type **File Date** in cell M8.
7. Type **Comment** in cell P8.
8. Select cells **B8** through **P8**.
9. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See “Format Cells dialog box” on page 53.
10. Click the **Font** tab.
11. In the Font Style box, select **Bold**.
12. From the list in the Underline box, select **Single**.
13. Click **OK** in the Format Cells dialog box.

You are now ready to use the XML Source pane to add mapped fields to your template.

Note A “node” in the XML Source tree creates a “field” in the spreadsheet.

❖ To add mapped fields to the template

1. To add mapped nodes from the SampleCentricExportBatch/Samples/ SampleCentricExportSample folder, do the following:

   a. Drag the **VialPosition** node from the XML Source tree to cell B9.

   b. Drag the **SampleId** node from the XML Source tree to cell C9.

   c. Drag the **RawFileName** node from the XML Source tree to cell F9.

   d. Drag the **SampleLevel** node from the XML Source tree to cell A9.

   e. Drag the **SampleName** node from the XML Source tree to cell J9.

   f. Drag the **AcquisitionDate** node from the XML Source tree to cell M9.

   g. Drag the **SampleComments** node from the XML Source tree to cell P9.

2. In cell H9, type the following formula:

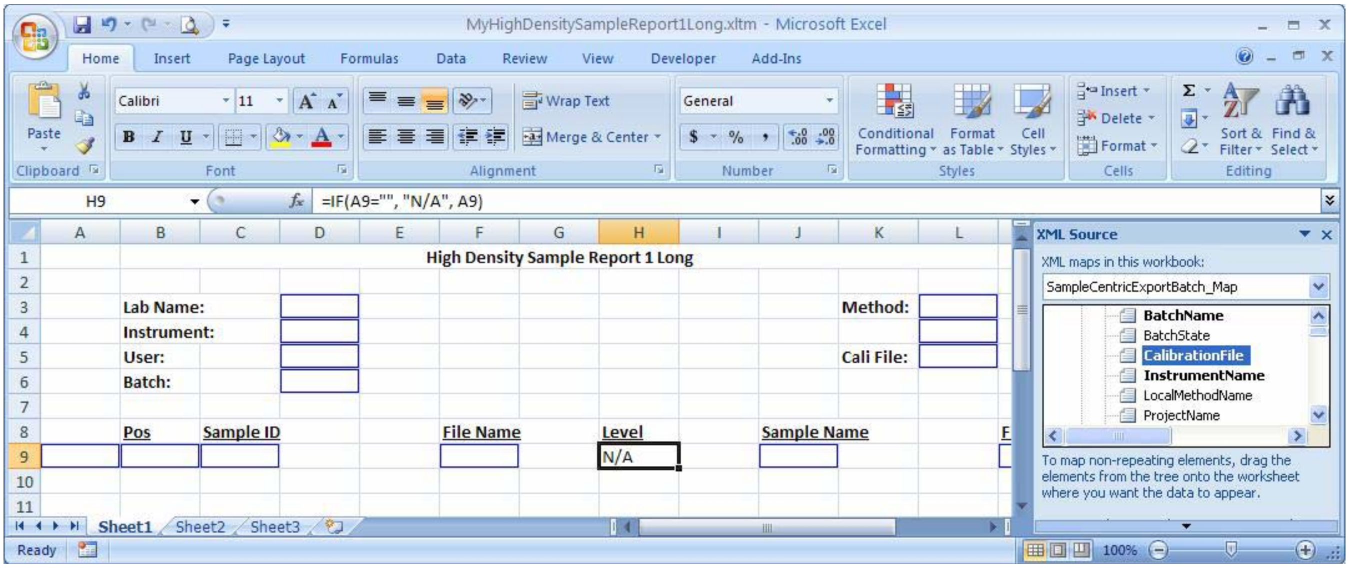
   ```
   =IF(A9="", "N/A", A9)
   ```

   This tells the table to show “N/A” when the SampleLevel at A9 is empty (column A is a hidden column), and otherwise, to show the SampleLevel value. You enter this formula at H9 only, but it will be automatically expanded to all the cells in this column when the data is imported into this table. Column A is a hidden column.

   Note Hidden columns, for example, the Level column on the final report, are normally added, because the XML Source data field cannot be used directly in the mapped table. This column is special because you want to display N/A instead of a blank when the field is empty, so you cannot just drag the field from the XML Source.

Ensure that your template looks like the following figure.

Figure 39. Template with file list



H9 =IF(A9="", "N/A", A9)

	A	B	C	D	E	F	G	H	I	J	K	L
1						High Density Sample Report 1 Long						
2												
3		Lab Name:									Method:	
4		Instrument:										
5		User:									Cali File:	
6		Batch:										
7												
8		Pos	Sample ID			File Name		Level		Sample Name		F
9								N/A				
10												
11												

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

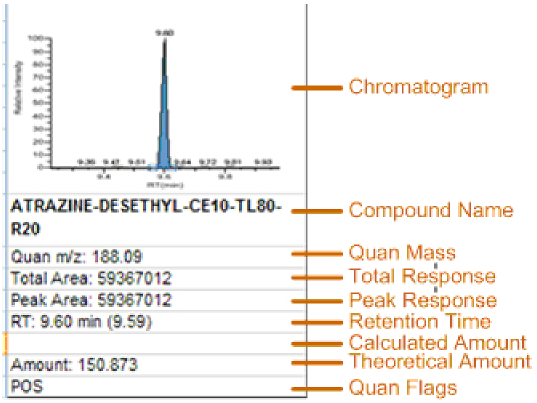
> **Note** You can format the cells in many ways to obtain the same final result. This tutorial creates a layout as close as possible to the delivered report. Adjusting the columns and cells to create professional-looking reports that meet your business requirements is a time-consuming process.

Creating a Repeating Area

The high density sample report repeats the same information for each compound in the sample.

The following figure shows the required information for each sample.

Figure 40. Required sample information



ATRAZINE-DESETHYL-CE10-TL80-R20

Quan m/z: 188.09

Total Area: 59367012

Peak Area: 59367012

RT: 9.60 min (9.59)

Amount: 150.873

POS

Where:

- The **Chromatogram** for the compound is exported in the XML file as a byte array. This image refers to the quantitation peak image in the SampleCentricExportBatch/Samples/SampleCentricExportSample/Compounds/SampleCentricCompoundExportData/QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/PeakImage folder.
- The **Compound Name** is the complete name identified in the library.
- The **Quan Mass** is the mass-to-charge ratio for the quantitation peak.
- The **Total Response** is either Total Area or Total Height, depending on the ResponseValue setting.
- The **Peak Response** is either Peak Area or Peak Height depending on the ResponseValue. For a single quantitation peak, the total response and peak response represent the same value.

  If the peak is manually integrated, a label "M" is appended to the entry.
- The **Calculated Amount** is either a calibration level amount (for Cal Std sample) or a QC level amount (for Chk Std sample) depending on the sample type.
- The **Calculated Amount** is a calibration level amount for a Cal Std sample.
- The **Calculated Amount** is either a calibration level amount (for Cal Std sample) or a QC level amount (for Chk Std sample) depending on the sample type.

- The **Theoretical Amount** is the Reported In Sample Concentration value of the quantitation result for target compounds. For internal standard compounds, this is the amount set in the method.
- The **Quan Flag** is the flag for the quantitation peak.

Adding Images to a Report

The Excel application embeds images as byte arrays in an exported XML file. The Excel spreadsheet cannot directly handle embedded images because the images are normally too large. You can create a named range that acts as placeholder for an image when you drag an image node from the XML Source tree. At run time, the application loads images from the XML file directly to the placeholders through the VBA code.

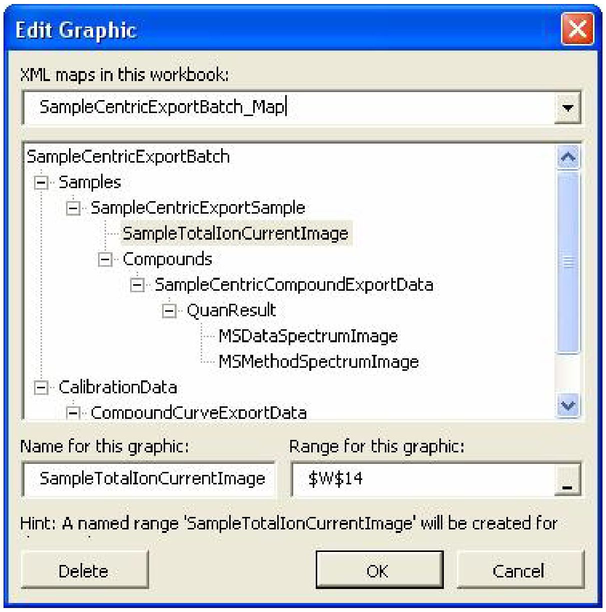
Tip All images file names in the exported XML file have an "Image" suffix, for example, SampleTotalIonCurrentImage for the TIC image of the sample. This naming convention identifies the node as an image file so that the application processes the node correctly when you drag it to your spreadsheet.

❖ To add an image file to the template

1. Open the SampleCentricExportBatch/Samples/SampleCentricExportSample folder and drag the **SampleTotalIonCurrentImage** image node from the XML Source tree to cell W14.

   The Edit Graphic dialog box opens.

   Figure 41. Edit Graphic dialog box



Edit Graphic

XML maps in this workbook: SampleCentricExportBatch_Map

Name for this graphic: SampleTotalIonCurrentImage

Range for this graphic: $W$14

Hint: A named range 'SampleTotalIonCurrentImage' will be created for

   The tree view in the dialog box lists only image nodes. The currently selected tree node is automatically highlighted.

2. In the Name for This Graphic box, leave the default name.

3. In the Range for This Graphic box, change the default to **$B$10:$F$18**.

Tip You can type the range or you can click the button in the Range for This Graphic box and select cells on the spreadsheet.

It is not necessary to create a named range placeholder for images because the VBA code can directly reference the cells. However, creating named ranges for images makes the maintenance easier.

4. Click **OK** in the Edit Graphic dialog box.

The application loads the images on the spreadsheet at run time using the following VBA code and the CreateImage add-in function:

```
mUtil.CreateImage NAMEDRANGEOFIMAGE, IMAGENODEPATH
```

Where:

- *NAMEDRANGEOFIMAGE* is the named range for the image you created at design time.
- *IMAGENODEPATH* is the XPath to the image node in the XML Source tree.

The following code inserts a peak image into this report:

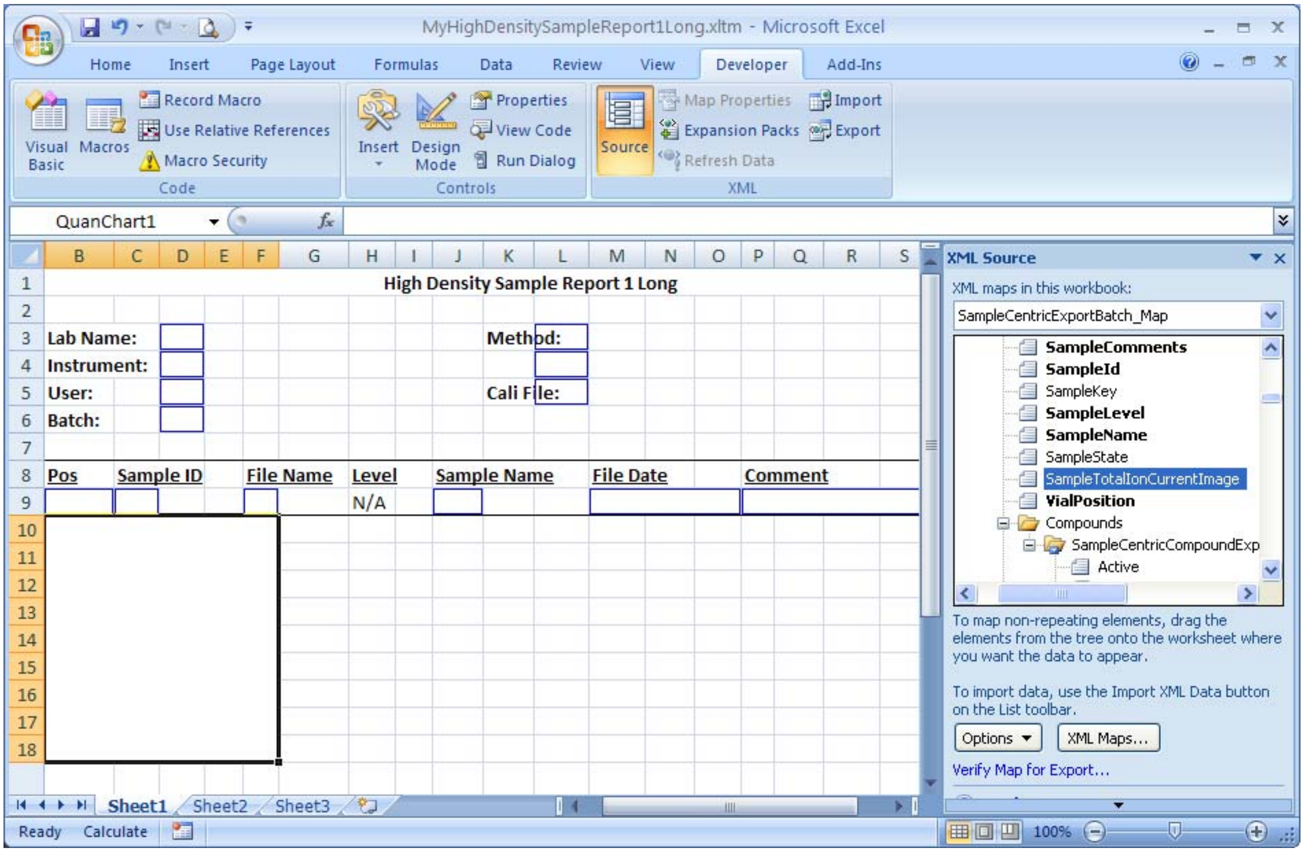
```
mUtil.CreateImage mUtil.shiftRange("QuanChart" & cc, 0, offset, 0,
offset),
"SampleCentricExportBatch/Samples/SampleCentricExportSample/Compounds
/SampleCentricCompoundExportData[CompoundKey='" & compKey &
"']/QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/PeakI
mage"
```

In this example, *NAMEDRANGEOFIMAGE* is dynamically calculated for each compound through the ShiftRange add-in function. *IMAGENODEPATH* uses an XPath expression to select the image node based on the compound key.

Note The XPath (the XML Path Language), defined by the World Web Consortium (W3C), is a query language for selecting nodes from an XML document. For additional XPath information, refer to the appropriate external documentation.

Ensure that your template looks like the following figure.

Figure 42. Template with image field



High Density Sample Report 1 Long

Lab Name:

Instrument:

User:

Batch:

Method:

Cali File:

Pos	Sample ID	File Name	Level	Sample Name	File Date	Comment
			N/A			

XML Source

XML maps in this workbook:

SampleCentricExportBatch_Map

- SampleComments
- SampleId
- SampleKey
- SampleLevel
- SampleName
- SampleState
- SampleTotalIonCurrentImage
- VialPosition
- Compounds
  - SampleCentricCompoundExp
    - Active

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

To import data, use the Import XML Data button on the List toolbar.

Options ▾ XML Maps...

Verify Map for Export...

Creating a Temporary Data Sheet

For this report, you cannot directly use the data as you did for the Batch Report because the data must be preprocessed. You will put the data on a separate sheet and use formulas to combine and reformat them before the VBA code uses them to generate the report.

❖ To create a temporary sheet

1. Right-click Sheet2 and choose **Rename** from the shortcut menu.

   The Excel application highlights the sheet name for editing.

2. Type **tempSheet** and press ENTER.
3. Right-click Sheet3 and choose **Delete** from the shortcut menu.
4. Click **Delete** in the confirmation dialog box.

You will use tempSheet as a data sheet to hold the temporary data used during the creation of the report, and it will be automatically deleted after the report is generated.

You are now ready to add mapped nodes from the SampleCentricExportBatch folder to create a data table.

❖ To add mapped nodes to the template

1. Open the Samples/SampleCentricExportSample/Compounds folder.
2. Open the SampleCentricCompoundExportData folder and drag the **CompoundName** node from the XML Source tree to cell A1.
3. Open the SampleCentricCompoundExportData folder and drag the **CompoundKey** node from the XML Source tree to cell B1.
4. Open the SampleCentricCompoundExportData folder and drag the **ResponseValue** node from the XML Source tree to cell C1.
5. Open the QuanResult folder and drag the **TotalResponse** node from the XML Source tree to cell D1.
6. Open the QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/ResultPeak folder and drag the **Area** node from the XML Source tree to cell E1.
7. Open the QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/ResultPeak folder and drag the **ApexHeightAboveBaseLine** node from the XML Source tree to cell F1.
8. Open the QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData folder and drag the **ManualFlagSet** node from the XML Source tree to cell G1.
9. Open the QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/ResultPeak folder and drag the **ApexRetentionTime** node from the XML Source tree to cell H1.

10. Open the SampleCentricCompoundExportData/QuanPeakIdentifiers/QuanPeakIdentifierExportData folder and drag the **DisplayRetentionTime** node from the XML Source tree to cell I1.
11. Open the SampleCentricCompoundExportData folder and drag the **LevelAmount** node from the XML Source tree to cell J1.
12. Open the QuanResult folder and drag the **CalculatedAmount** node from the XML Source tree to cell K1.
13. Open the SampleCentricCompoundExportData folder and drag the **Units** node from the XML Source tree to cell L1.
14. Open the SampleCentricCompoundExportData folder and drag the **CompoundType** node from the XML Source tree to cell M1.
15. Open the SampleCentricCompoundExportData folder and drag the **Amount** node from the XML Source tree to cell N1.
16. Open the QuanResult folder and drag the **ReportedInSampleConcentration** node from the XML Source tree to cell O1.
17. Open the SampleCentricCompoundExportData folder and drag the **TotalExpectedRetentionTimes** node from the XML Source tree to cell P1.
18. Open the SampleCentricCompoundExportData/QuanPeakIdentifiers/QuanPeakIdentifierExportData folder and drag the **DisplayMassRange** node from the XML Source tree to cell Q1.
19. Open the QuanResult folder and drag the **QuanFlags** node from the XML Source tree to cell R1.
20. Open the QuanResult folder and drag the **Active** node from the XML Source tree to cell S1.
21. Open the QuanResult folder and drag the **IsValidCompound** node from the XML Source tree to cell T1.

You are now ready to name the data table you created.

❖ To name the data table in the template

1. Click the **Formulas** tab and then click **Name Manager**.

   Note The Excel application automatically creates a name for all tables, such as Table1 or Table2. Each time you create a new table, it increments the table name, even if you deleted a previous one.

2. In the Name Manager dialog box, double-click **Table1**.

   The table name might vary. Use the sheet, row, and column numbers in the Refers To column to verify the table.

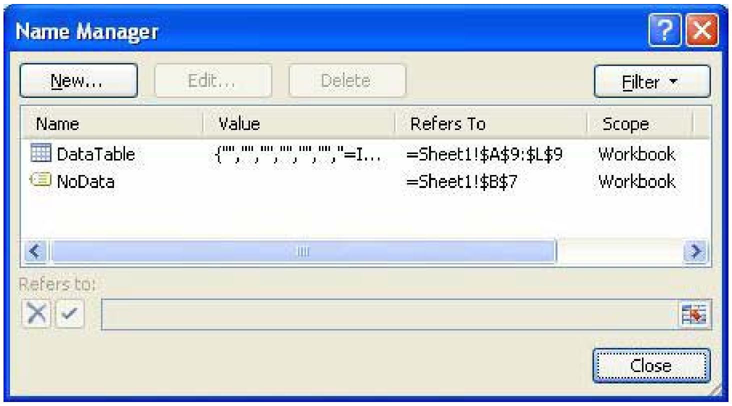
The Edit Name dialog box opens.

3. Change the default name to **DataTable**.

4. Click **OK** in the Edit Name dialog box.

   You now have two names defined, as shown in the following figure.

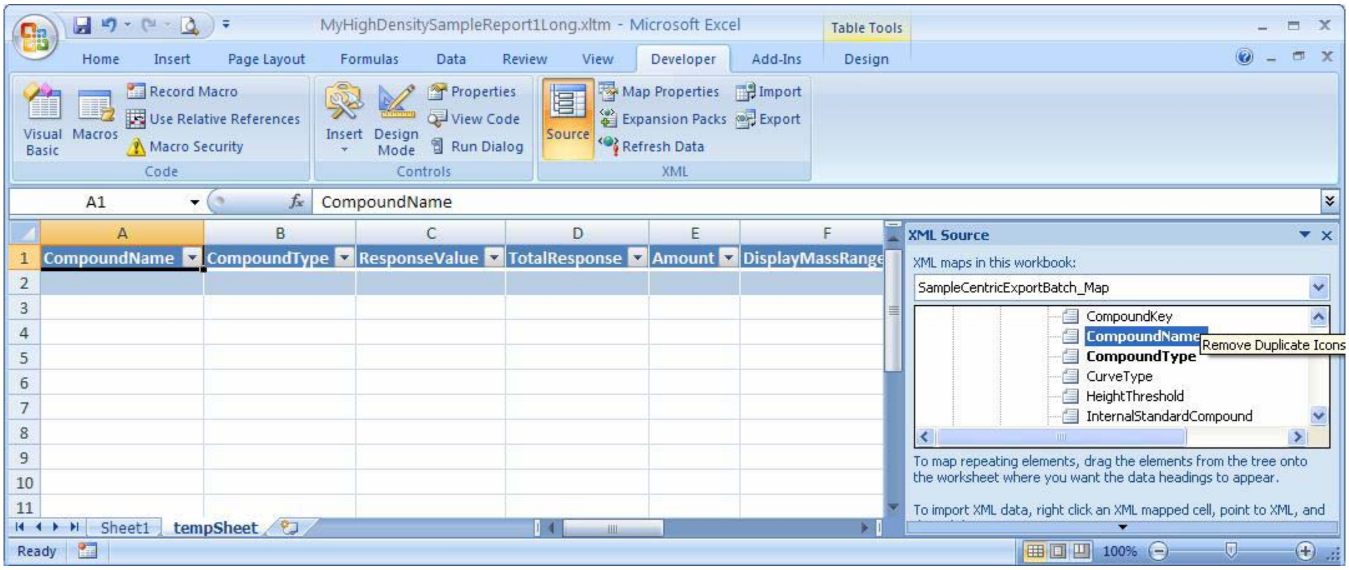
Figure 43. Name Manager dialog box



Name	Value	Refers To	Scope
DataTable	{"","","","","","","=I...	=Sheet1!$A$9:$L$9	Workbook
NoData		=Sheet1!$B$7	Workbook

Ensure that your template looks like the following figure.

Figure 44. Template with mapped nodes



Some values are exported more than once, mostly in different formats and at different tree nodes or levels, to facilitate custom report creation. Sometimes a display value can depend on several other fields, and putting all the logic on the spreadsheet can be overwhelming.

For help in finding the right data point from the XML Source, see Appendix B, "Custom Report Add-in API."

Using Data in a Repeating Area

In a repeating area, some of the fields are populated by using VBA code and others by using formulas within the Excel application.

Populating Fields Using Formulas

For this report, you will use formulas at two levels to simplify the logic at each level.

- The first-level formulas are at the data table level. In the Excel application, you can add formula columns to a data table. The formula is evaluated when you import the data into the data table and expanded to each imported data row. This level of formula lets you combine values or reformat values to make them ready for display.
- The second-level formulas are embedded in the repeating area. This uses the Excel VLookup function to get the data element for a specific compound.

You can add the formula columns anywhere in the data table but for the TraceFinder reports, we add them to the beginning of the data table, to make the use of the Excel VLookup function (which is based on column index) easier. For the Excel VLookup function to work, the first column of the data table must be the "key" column. In our reports, the key is the compound name, so you must keep CompoundName as the first column.

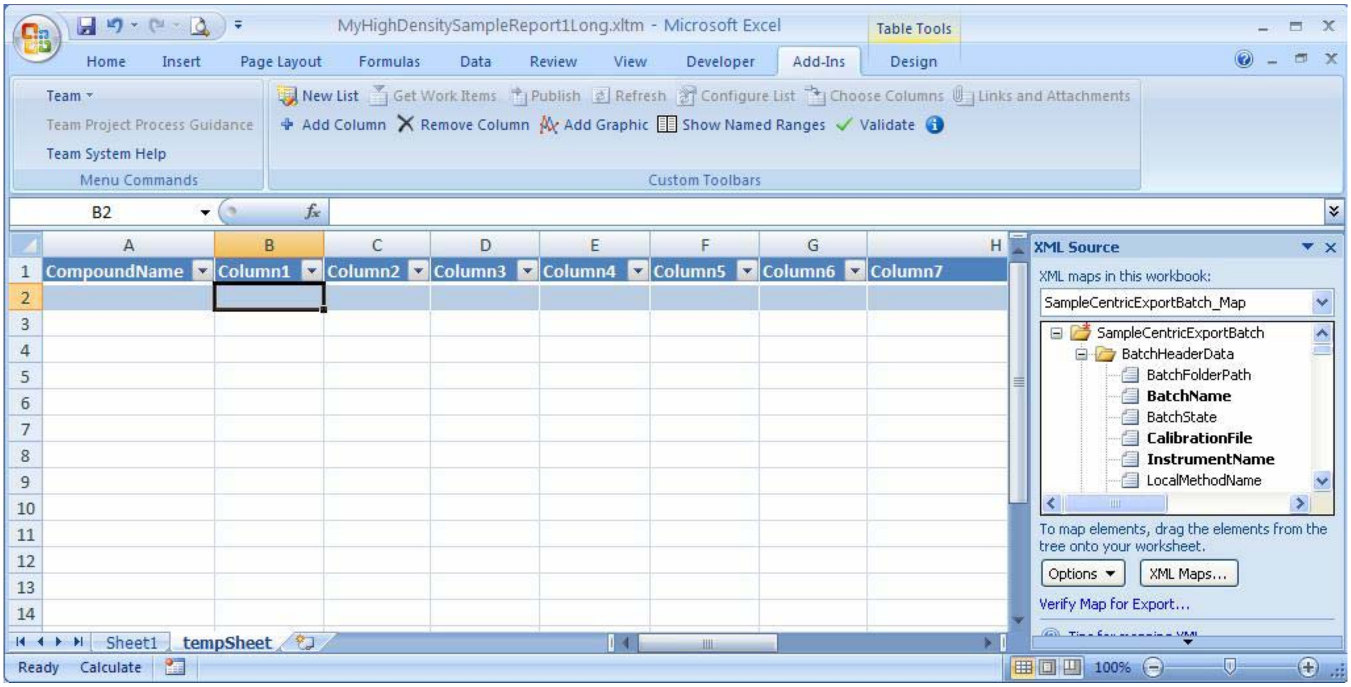
You are now ready to add columns to your data table.

❖ To create the columns in the data table

1. Right-click the CompoundType cell and choose **Insert > Table Columns to the Left** from the shortcut menu.
2. Repeat Step 1 seven times, using the default column names **Column1** through **Column7**.

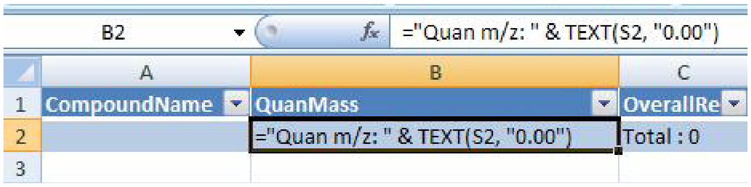
Ensure that your template looks like the following figure.

Figure 45. Template with inserted columns



	A	B	C	D	E	F	G	H
1	CompoundName	Column1	Column2	Column3	Column4	Column5	Column6	Column7

When you add a new column to the data table, the cell format defaults to Text. In this format, the formula you enter into the cell appears as text (notice the equal sign in the cell, instead of an evaluated result). As text, the formula will not expand when data is imported into data table. For example:



B2 ="Quan m/z: " & TEXT(S2, "0.00")

	A	B	C
1	CompoundName	QuanMass	OverallRe
2		="Quan m/z: " & TEXT(S2, "0.00")	Total : 0
3			

You are now ready to manually change the cell formats from Text to General.

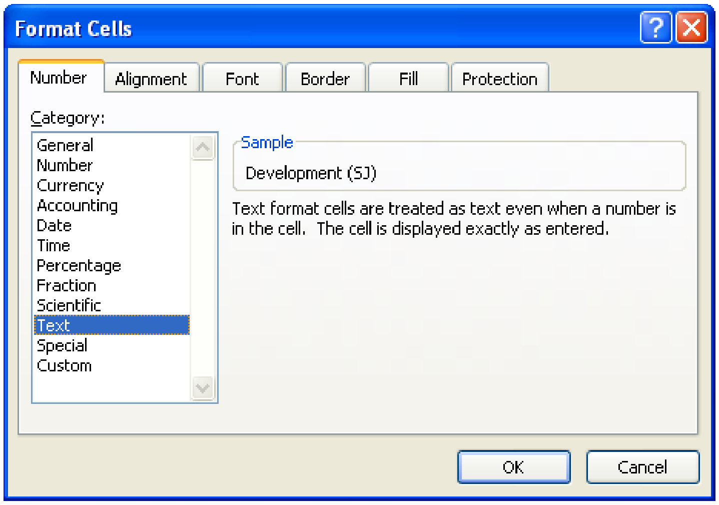
❖ To change the cell format in the template

1. Select the **Column1** cell.

2. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

Figure 46. Format Cells dialog box - Number page



Sample: Development (SJ)

Text format cells are treated as text even when a number is in the cell. The cell is displayed exactly as entered.

3. Click the **Number** tab.

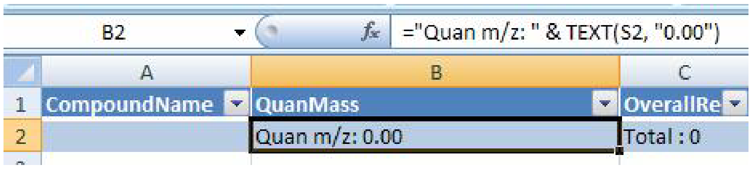
4. Select **General** in the Category list and click **OK**.

5. While the Column1 cell is still selected, press ENTER.

   The Excel application does not automatically update the new cell format. Pressing ENTER manually updates it.

6. Repeat steps 1-4 for each of the seven columns that you added.

The formula in cells with the General format (no equal sign at the beginning of the cell content) expands when data is imported into the data table, as shown in the following figure.



B2 =“Quan m/z: ” & TEXT(S2, “0.00”)

	A	B	C
1	CompoundName	QuanMass	OverallRe
2		Quan m/z: 0.00	Total : 0

You are now ready to give the column headers meaningful names.

❖ To rename the column headers and enter the formulas

1. In cell B1, change Column1 to **QuanMass**.

2. In cell B2, enter the following formula:

```
="Quan m/z: " & TEXT(S2, "0.00")
```

3. In cell C1, change Column2 to **OverallResponse**.

4. In cell C2, enter the following formula:

```
="Total " & J2 & ": " & ROUND(K2, 0)
```

5. In cell D1, change Column3 to **PeakResponse**.

6. In cell D2, enter the following formula:

   ```
   ="Peak" & J2 & ": " & ROUND(IF(J2="Area", L2, M2), 0) & IF(N2, "M", "")
   ```

7. In cell E1, change Column4 to **RT**.

8. In cell E2, enter the following formula:

   ```
   ="RT: " & TEXT(O2, "0.00") & " min (" & TEXT(P2, "0.00") & ")"
   ```

9. In cell F1, change Column5 to **CalculatedAmount**.

10. In cell F2, enter the following formula:

    ```
    =IF(Q2="", "", "TAmount: " & TEXT(Q2, "0.000") & " " & U2)
    ```

11. In cell G1, change Column6 to **TheoreticalAmount**.

12. In cell G2, enter the following formula:

    ```
    ="Amount: " & IF(I2="Internal Standard", TEXT(N2, "0.000"), TEXT(V2,
    "0.000")) & " " & U2
    ```

13. In cell H1, change Column7 to **QuanFlag**.

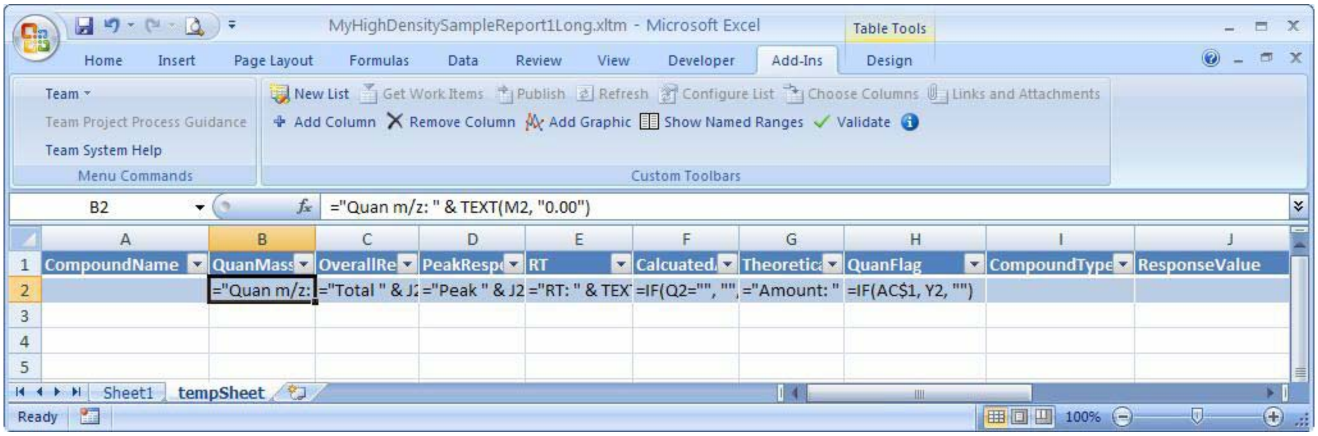
14. In cell H2, enter the following formula:

    ```
    =IF(AC$1, Y2, "")
    ```

15. Double-check the formulas to confirm that the referenced columns are correct.

Ensure that your template looks like the following figure.

Figure 47. Template with formulas



B2 ="Quan m/z: " & TEXT(M2, "0.00")

	A	B	C	D	E	F	G	H	I	J
1	CompoundName	QuanMass	OverallRe	PeakResp	RT	Calcuated	Theoretic	QuanFlag	CompoundType	ResponseValue
2		="Quan m/z:	="Total " & J2	="Peak " & J2	="RT: " & TEX	=IF(Q2="", "",	="Amount: "	=IF(AC$1, Y2, "")		
3										
4										
5										

You are now ready to specify formulas in the repeating area on Sheet1.

❖ To specify the formulas in the template

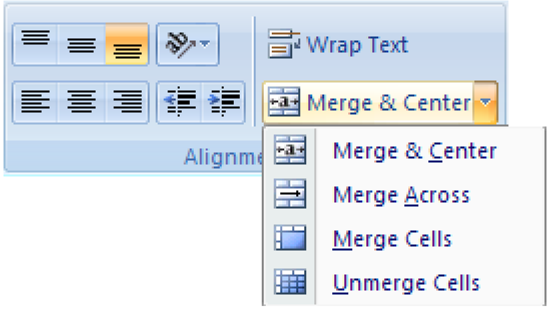
1. Click the **Sheet1** tab.

2. Select cells **B19** through **F26**.

3. Click the **Home** tab to display the Home functions.

4. Click the down arrow next to **Merge & Center** in the Alignment area.

Figure 48. Merging cells



5. Choose **Merge Cells** from the menu.

   Merging these cells creates more space for each data element.

   Cells B19 through F19 are used for the Compound Name. You use VBA code to set up the compound name. The other fields in the repeating area use CompoundName as a reference to retrieve data from the DataTable on the tempSheet.

6. Select cells **B20** through **F20**, and enter the following formula for Quan Mass:

   ```
   =IF(ISNA(VLOOKUP(B19, DataTable, 2, FALSE)), "",VLOOKUP(B19,
   DataTable, 2, FALSE))
   ```

7. Select cells **B21** through **F21**, and enter the following formula for Total Response:

   ```
   =IF(ISNA(VLOOKUP(B19, DataTable, 3, FALSE)), "", VLOOKUP(B19,
   DataTable, 3, FALSE))
   ```

8. Select cells **B22** through **F22**, and enter the following formula for Peak Response:

   ```
   =IF(ISNA(VLOOKUP(B19, DataTable, 4, FALSE)), "",
   VLOOKUP(B19,DataTable, 4, FALSE))
   ```

9. Select cells **B23** through **F23**, and enter the following formula for RT:

   ```
   =IF(ISNA(VLOOKUP(B19, DataTable, 5, FALSE)), "",
   VLOOKUP(B19,DataTable, 5, FALSE))
   ```

10. In the cell range **B24** through **F24**, and enter the following formula for Calculated Amount:

    ```
    =IF(ISNA(VLOOKUP(B19, DataTable, 6, FALSE)), "", VLOOKUP(B19,
    DataTable, 6, FALSE))
    ```

11. Select cells **B25** through **F25**, and enter the following formula for Theoretical Amount:

    ```
    =IF(ISNA(VLOOKUP(B19, DataTable, 7, FALSE)), "", VLOOKUP(B19,
    DataTable, 7, FALSE))
    ```

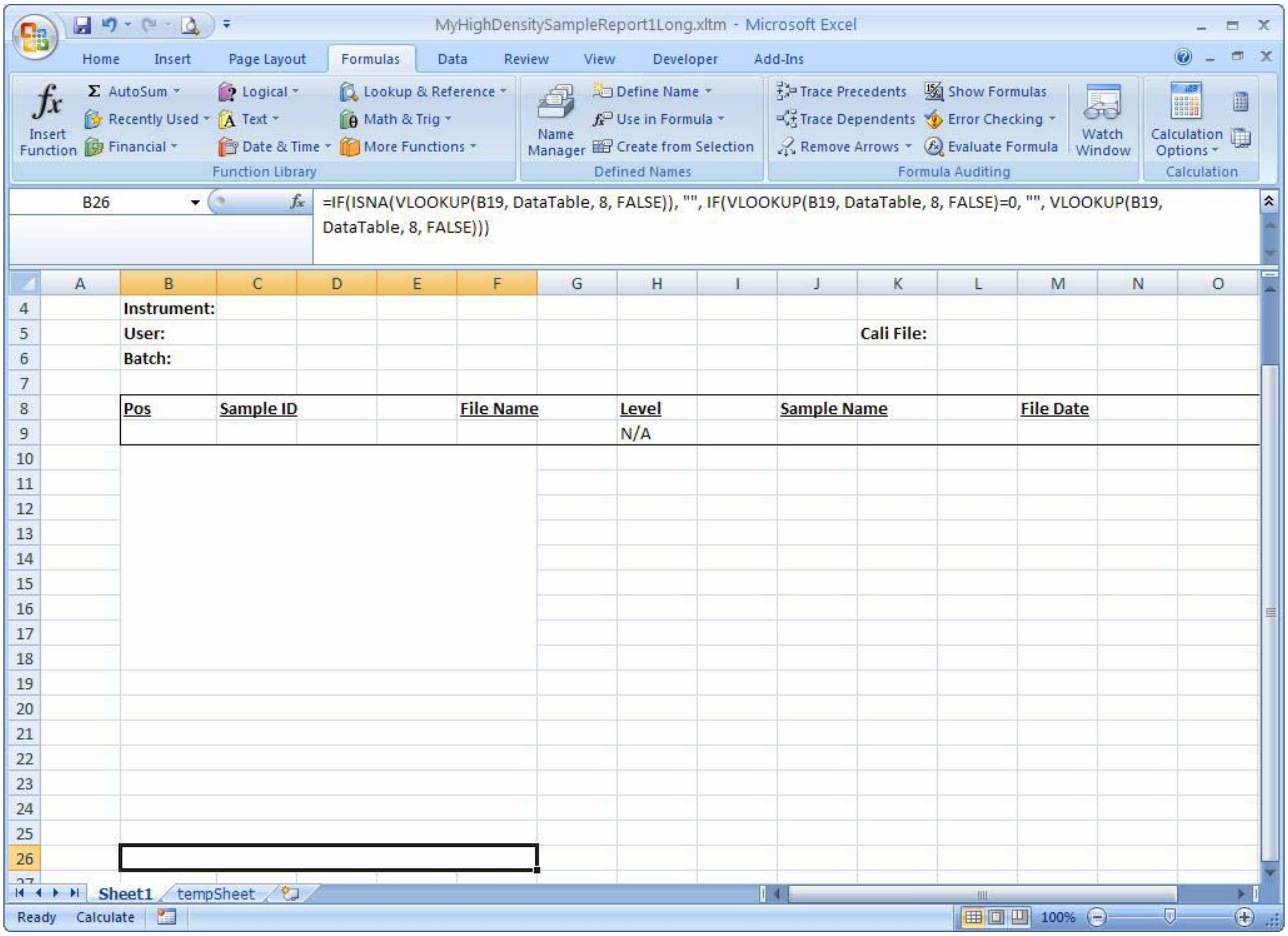
12. Select cells **B26** through **F26**, and enter the following formula for Quan Flag:

    ```
    =IF(ISNA(VLOOKUP(B19, DataTable, 8, FALSE)), "", IF(VLOOKUP(B19,
    DataTable, 8, FALSE)=0, "", VLOOKUP(B19, DataTable, 8, FALSE)))
    ```

The formulas are designed so that nothing is visible when the CompoundName cell is empty.

Ensure that your template looks like the following figure.

Figure 49. Template with formulas in a repeating area



B26

```
=IF(ISNA(VLOOKUP(B19, DataTable, 8, FALSE)), "", IF(VLOOKUP(B19, DataTable, 8, FALSE)=0, "", VLOOKUP(B19,
DataTable, 8, FALSE)))
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
4		**Instrument:**													
5		**User:**									**Cali File:**				
6		**Batch:**													
7															
8		**Pos**	**Sample ID**			**File Name**		**Level**		**Sample Name**			**File Date**		
9								N/A							

The template for the repeating area is complete. You are now ready to duplicate the repeating area for each compound with four compounds in a row and three rows on a page (a total of 12 on each page).

Completing the Layout

In the following procedures, you will modify the layout and create named ranges.

You are now ready to hide column A.

❖ To hide the SampleLevel column

Right-click the column A header and choose **Hide** from the shortcut menu.

> **Tip** You can hide any column in the Excel application. However, if you need hidden columns for the TraceFinder custom reports, always use the first columns. A hidden column in the middle of the table can cause trouble when copying blocks of data for the repeating area.

You are now ready to modify the font in the merged cells.

❖ To modify row 19 in the template

1. Click the number for row 19.
2. Drag the horizontal separator between row 19 and row 20 until row 19 is twice its original height.

   > **Tip** Column width is controlled by the right separator; row height is controlled by the bottom separator.

   Because of this extra height, row 19 can now accommodate long compound names.

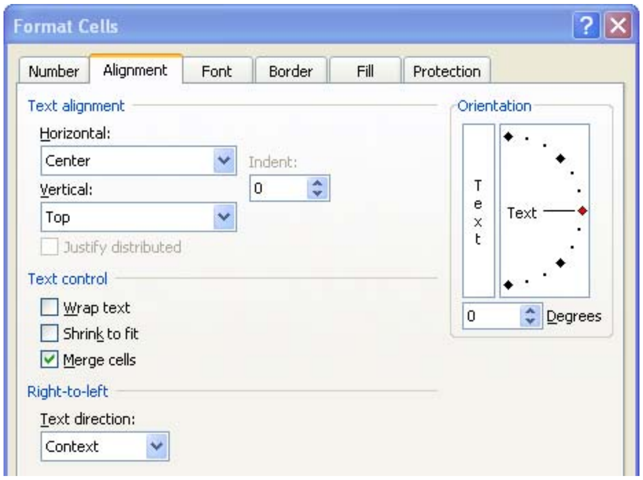
3. To prevent the text in merged cells from running over its boundary, do the following:

   a. Select the merged cells.

   b. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

Figure 50. Format Cells dialog box - Alignment page



Format Cells

Number | Alignment | Font | Border | Fill | Protection

Text alignment

Horizontal: Center

Indent: 0

Vertical: Top

☐ Justify distributed

Text control

☐ Wrap text

☐ Shrink to fit

☑ Merge cells

Right-to-left

Text direction: Context

Orientation

0 Degrees

c. Click the **Alignment** tab.

d. In the Text Control area, select **Wrap Text**.

e. In the Vertical box, click the down arrow and select **Top**.

f. Click the **Font** tab.

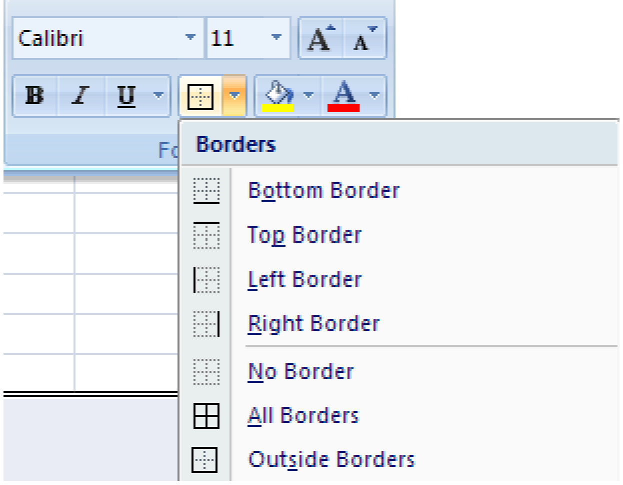
g. In the Font Style box, select **Bold**.

h. Click **OK** in the Format Cells dialog box.

You are now ready to add a border to the merged cells.

❖ To add a border around the merged cells

1. Select the cells **B10** through **F26**.
2. Click the **Home** tab to display the Home functions.
3. Click the down arrow next to the Borders icon in the Font area.



4. Choose **Outside Borders** from the menu.

   This border groups the compound information.

❖ To copy the repeating area

1. Select the cells **B10** through **F26**.

   You cannot simply click the bordered area; you must drag your cursor across all the cells to select them.

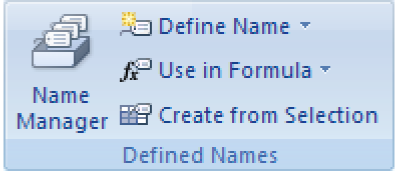
2. Right-click and choose **Copy** from the shortcut menu.
3. Right-click in cell G10, and choose **Paste** from the shortcut menu.
4. Repeat Step 3 for cells L10 and Q10.

You are now ready to name the QuanChart ranges.

❖ To name the QuanChart repeated areas

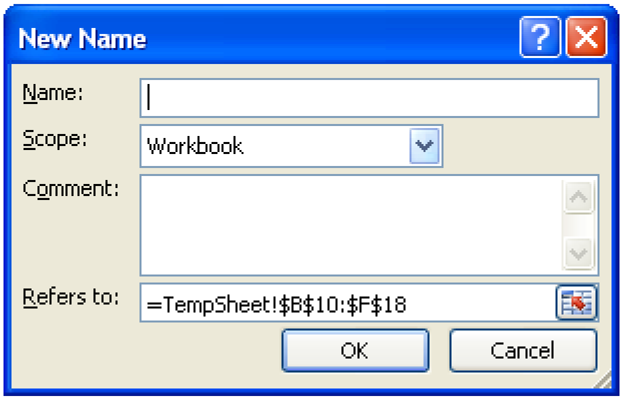
1. Select cells **B10** through **F18**.

2. Click the **Formulas** tab, and then click **Define Names**.



The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.

Figure 51. New Name dialog box



Name:

Scope: Workbook

Comment:

Refers to: =TempSheet!$B$10:$F$18

3. In the Name box, type **QuanChart1**.

4. Click **OK**.

5. Repeat steps 1 through 4 using the following values:

   - Cells G10 through K18 - QuanChart2
   - Cells L10 through P18 - QuanChart3
   - Cells Q10 through U18 - QuanChart4

❖ To create the CompoundName named ranges

1. Select cell **B19**.

2. Click the **Formulas** tab, and then click **Define Name**.

   The selected cell automatically appears in the **Refers To** box of the New Name dialog box.

3. In the Name box, type **CompoundName1**.

4. Click **OK**.

5. Repeat steps 1 through 4 for the following values:
   - Cell G19 - CompoundName2
   - Cell L19 - CompoundName3
   - Cell Q19 - CompoundName4

❖ To create a named range that includes all QuanChart ranges

1. Select cells **B10** through **U26**.

   The selected cells include all four QuanChart ranges.

2. Click the **Formulas** tab, and then click **Define Name**.

   The selected cell automatically appears in the **Refers To** box of the New Name dialog box.

3. In the Name box, type **RepeatArea**.
4. Click **OK**.

   You will treat this range as a single unit of the repeating area.

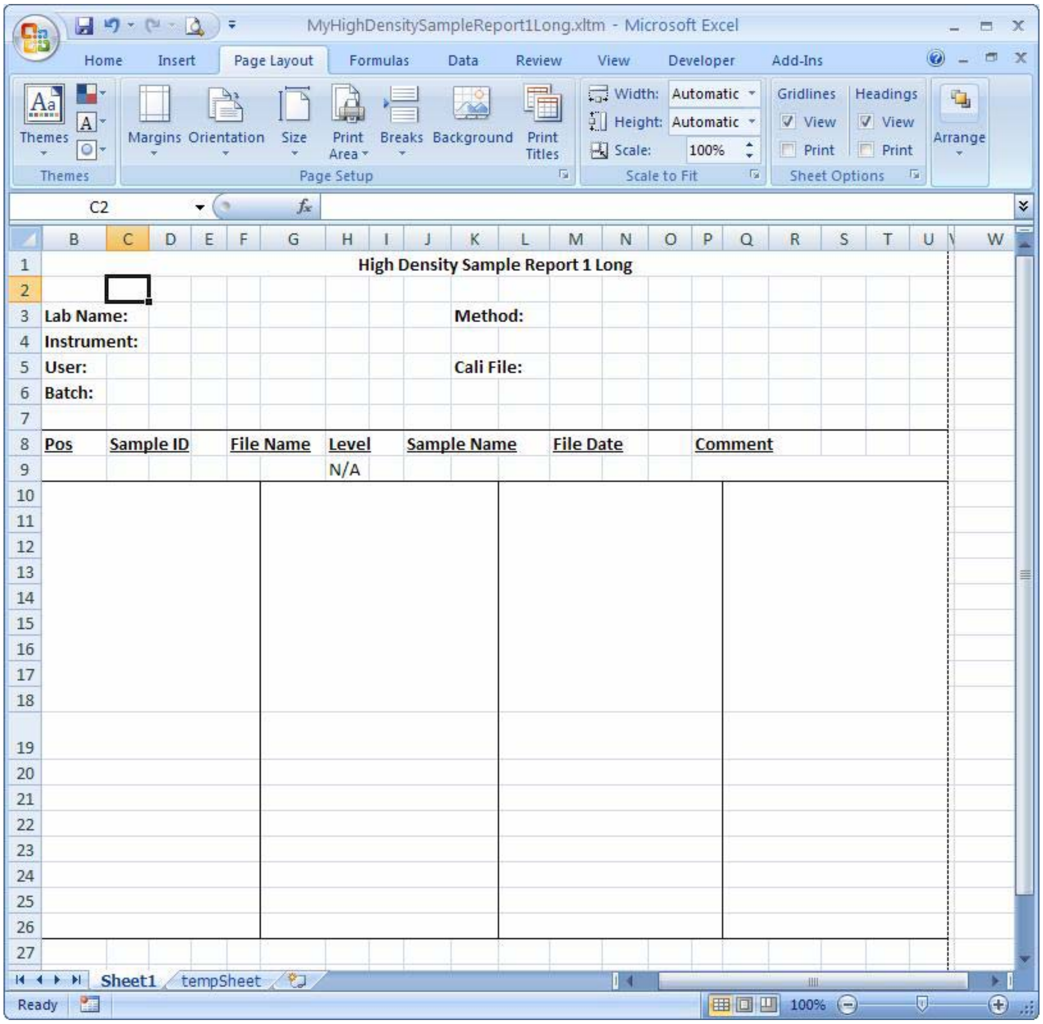
❖ To adjust the QuanChart columns

Drag the dividers in the column headings to make all the columns fit within the page range.

When you are in page preview mode, the right margin of the page is indicated by a dashed line.

Ensure that your template looks like the following figure.

Figure 52. Completed template design



MyHighDensitySampleReport1Long.xltm - Microsoft Excel

C2

Width: Automatic
Height: Automatic
Scale: 100%

Gridlines: ☑ View ☐ Print
Headings: ☑ View ☐ Print

High Density Sample Report 1 Long

Lab Name: Method:
Instrument:
User: Cali File:
Batch:

Pos	Sample ID	File Name	Level	Sample Name	File Date	Comment
			N/A			

100%

Adding VBA Code to the Template

The template interface design is complete. You are ready to add VBA code to connect the data to the interface.

Using the VBA code, you will do the following:

- Populate the RepeatArea for each compound by making each sheet a single page and creating as many sheets as needed.

  Note There are two ways to populate the RepeatArea for each compound:

  - Make multiple copies of the RepeatArea on the same sheet to create one sheet containing the entire report.

    Creating a single sheet for the entire report can have a negative performance impact. The Excel application can become very slow as the sheet becomes large. A sheet with more than 10 printed pages is not recommended.

  - Make each sheet a single page and create as many sheets as needed.

    The Excel application can better handle many sheets with a small set of data on each sheet. This is the preferred approach.

- Put 12 compounds on each sheet (which makes up a single printed page) and create as many sheets as needed.
- Name each sheet after the compound name of the first compound on the sheet—with some special characters removed to make it eligible as an Excel application tab name.
- Group the sheets and use an add-in component to make the page numbering continuous across the sheets.
- Use Excel’s Sheet.Copy function to copy a sheet.

Using the CopyPaste Add-in Function

The Excel application’s native CopyPaste function uses the Clipboard to transfer data, which is a shared resource for all applications. For the purposes of this tutorial, you do not want to share the data with other applications.

As a workaround, use a special CopyPaste function in the add-in component that avoids using the Clipboard. When you want to copy and paste cell data, use the CopyPaste add-in function instead of Excel’s CopyPaste function.

VBA Code in Module1

In the Getting Started chapter, you created two empty functions in Module1: ProcessData() and Validate().

- The Validate() function is called before the XML data is imported and it checks if the named ranges are intact. Validate checks for the following types of named ranges:
  - Required–Without the required name ranges, the template cannot work properly.
  - Optional–The template uses a default value if an optional range is missing. For example, PageBreakEvery is an optional named range that specifies how many rows of the RepeatArea to include on a single page. By default, you have three rows on a page. If you add more data to the RepeatArea and need more space for each row, you can define a named range with the cell value set to 2, so that each page will be made of two rows.
- The ProcessData() function is called after the XML data is imported into the spreadsheet.

  As a convention, we put the logic code for each sheet in a separate function, for example, ProcessSheet1Data().

You are now ready to create a special footer for the flag legend.

❖ To create a special footer

1. Click the **Developer** tab to display the Developer features.
2. Click **Visual Basic**.

   The Microsoft Visual Basic window opens.
3. In the Project - VBAProject pane, double-click **MyUtil** to open the MyHighDensitySampleReport1Long.xltm - MyUtil (Code) window.
4. Change the IUtil_SetupPageFooter() interface function in the MyUtil class module

   From:

   ```
   Private Sub IUtil_SetupPageFooter()
       mUtil.SetupPageFooter “”, 0.5
   End Sub
   ```

   To:

   ```
   Private Sub IUtil_SetupPageFooter()
       mUtil.SetupPageFooter GetPageFooterText, 0.5
   End Sub
   ```

The IUtil_SetupPageFooter() function calls the GetPageFooterText function defined in Module1.

You are now ready to add VBA code to create multiple repeating areas.

❖ To duplicate the repeating area for each compound

1. In the Project - VBAProject pane, double-click **Module1** to open the MyHighDensitySampleReport1Long.xltm - Module1 (Code) window.

2. Replace the code in the Module1 code editor with the following code:

```
Option Explicit
' Called after xml data gets imported
Public Sub ProcessData()
    On Error GoTo ErrorHandler
    mUtil.LogMsg "Processing - Start"
    ProcessSheet1Data
    mUtil.LogMsg "Processing - End"
    Exit Sub
ErrorHandler:
    MsgBox Err.Number & " - " & Err.Source & ": " & Err.Description,
vbCritical, "Process Data Error"
End Sub


Private Sub ProcessSheet1Data()
    mUtil.LogMsg "Processing - Sheet1 - Start"
    ActiveWorkbook.Sheets("Sheet1").Activate
   Dim i As Long, j As Long, lstObj As ListObject, compName As String,
offset As Long, k As Long
   Dim n As Long, h As Long, rr As Long, cc As Long, pageBreakEvery As
Long, compKey As String, nn As Long


    ' Setup page header/footer here once and it will be copied to all
sheets by Sheet.Copy.
    ' Otherwise, it's slow to set it up sheet by sheet.
   ' NOTE: To make this work, the page header/footer must meet certain
requirements.
    mUtil.SkipSetupPageHeaderFooter = True
    mIUtil.SetupPageHeader
    mIUtil.SetupPageFooter
    If
mUtil.GetXmlElement("SampleCentricExportBatch/Samples/SampleCentricEx
portSample/ApplicationSampleType") <> "Breakdown" Then
        pageBreakEvery = mUtil.GetValue("PageBreakEvery", 3)
        h = Range("RepeatArea").Rows.Count  ' row height
        ' First duplicate the RepeatArea to fill up the whole page
        For i = 2 To pageBreakEvery
          mUtil.CopyPaste "RepeatArea", Range("RepeatArea").offset((i
- 1) * h)
        Next
        Set lstObj = Worksheets("tempSheet").ListObjects("DataTable")
       mUtil.LogMsg "Processing - Start - " & lstObj.Range.Rows.Count
- 2 & " Compounds"
        For i = 2 To lstObj.Range.Rows.Count
            If lstObj.ListColumns("Active").Range(i) And
lstObj.ListColumns("IsValidCompound").Range(i) And
lstObj.ListColumns("CompoundType").Range(i) <> "Native" And
lstObj.ListColumns("CompoundType").Range(i) <> "Breakdown" Then
                n = n + 1
                compName = lstObj.Range.Cells(i, 1)
                If (n - 1) Mod 12 = 0 Then
                    If n > 1 Then
                        ' Refresh the formula
                        Application.Calculate
                      ' Turn formula into text because we will delete
the source data at the end
                        ActiveSheet.UsedRange.Value =
ActiveSheet.UsedRange.Value
                    End If
                    ' Make a new sheet by copying it from Sheet1
                    Worksheets("Sheet1").Copy
Before:=Worksheets("tempSheet")
' Set the sheet name to compound name for easier review
                    ActiveSheet.Name =
mUtil.GetValidSheetName(compName)
                    nn = 0
                End If
                nn = nn + 1 ' tracks how many compounds in a sheet
                rr = Int((nn - 1) / 4) + 1  ' row number
                cc = (n - 1) Mod 4 + 1      ' column number
                offset = (rr - 1) * h
                Range("CompoundName" & cc).offset(offset) = compName
                compKey = mUtil.VLookup(compName, Range("DataTable"),
11, False)
```

```
              mUtil.CreateImage mUtil.shiftRange("QuanChart" & cc, 0,
offset, 0, offset),
"SampleCentricExportBatch/Samples/SampleCentricExportSample/Compounds
/SampleCentricCompoundExportData[CompoundKey='" & compKey &
"']/QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/PeakI
mage"

                If n Mod 10 = 0 Then mUtil.LogMsg "Processing - " & n
& "th Compound"

            End If

        Next

        If nn > 0 Then  ' Last sheet not finalized yet

            Application.Calculate

            ' Turn formula into text

            ActiveSheet.UsedRange.Value = ActiveSheet.UsedRange.Value

        End If

        mUtil.LogMsg "Processing - End"

    End If

    If n = 0 Then

        mUtil.SetNoData "NoData"

    Else

        Sheets("Sheet1").Delete

    End If

    Sheets("tempSheet").Delete

    mUtil.LogMsg "Processing - Sheet1 - End"

End Sub


Public Function GetPageFooterText() As String

    If
mUtil.GetXmlElement("SampleCentricExportBatch/MethodHeaderData/ShowFl
agsAndLegend") Then

        GetPageFooterText = "Flag legend: LOD<J<JOQ; I=Ion ratio
failure; C=Carryover; ?=Linearity limit; D=Detection limit; Q=Quan
limit; POS=Rpt limit; b=Blank; s=Solvent blank."

    End If

End Function


' Validate the template design e.g. named ranges

Public Function Validate(Optional showWarning As Boolean = True) As
Boolean

    Dim err_msg As String, info_msg As String, msg As String

    ' Check required names
```

```
mUtil.CheckRequiredName err_msg, "RepeatArea", "The 'RepeatArea' is 
the named range for repeating area."

mUtil.CheckRequiredName err_msg, "DataTable", "The 'DataTable' is the 
named table including all compound data in TempSheet."

    mUtil.CheckRequiredName err_msg, "QuanChart1", "The 'QuanChart1' 
is the named cell for the first chart."

    mUtil.CheckRequiredName err_msg, "QuanChart2", "The 'QuanChart2' 
is the named cell for the second chart."

    mUtil.CheckRequiredName err_msg, "QuanChart3", "The 'QuanChart3' 
is the named cell for the third chart."

    mUtil.CheckRequiredName err_msg, "QuanChart4", "The 'QuanChart4' 
is the named cell for the fourth chart."

    mUtil.CheckRequiredName err_msg, "CompoundName1", "The 
'CompoundName1' is the named cell for the compound name of the first 
chart."

    mUtil.CheckRequiredName err_msg, "CompoundName2", "The 
'CompoundName2' is the named cell for the compound name of the second 
chart."

    mUtil.CheckRequiredName err_msg, "CompoundName3", "The 
'CompoundName3' is the named cell for the compound name of the third 
chart."

mUtil.CheckRequiredName err_msg, "CompoundName4", "The 'CompoundName4' 
is the named cell for the compound name of the fourth chart."

' Check optional names

    mUtil.CheckOptionalNameOutsideOf err_msg, info_msg, 
"PageBreakEvery", "RepeatArea", "Page break is set at every 3 rows as 
default."

    ' Display message if any

    mUtil.ShowErrMsg err_msg, info_msg, showWarning

    Validate = err_msg = ""

End Function
```

Where:

- **ProcessData()** calls ProcessSheet1Data and is used for error handling.
- **ProcessSheet1Data** contains the main body of this report:
  - **LogMsg** is an add-in function used for logging messages, primarily for debugging.
  - **SetupPageHeader** and **SetupPageFooter** are add-in functions that are usually called once for each page break. Because this tutorial uses the Excel's Sheet.Copy function to duplicate the pages (including the headers and footers), you do not need these functions.
  - The first **IF** statement checks if the sample type is a Breakdown type. The high density sample report is valid only for non-breakdown sample types.

  - If **PageBreakEvery** is undefined, it defaults to 3, which means that each page can have a maximum of three RepeatArea rows.

  - The **RepeatArea** is duplicated as follows:

    i. Duplicates the RepeatArea to fill the entire page using the CopyPaste add-in function (it does not use the Clipboard).

    ii. Fetches the data from DataTable and grabs each valid compound. To be valid, the compound must be active, isValidCompound must be true, and the compound type must not be Native or Breakdown.

    iii. Fetches the data from DataTable and grabs each valid compound. To be valid, the compound must be active and isValidCompound must be true.

    iv. Sets the compound name to the sheet (for example, B19). This automatically updates the formulas on the sheet. Calls **Application.Calculate** to refresh the formulas before copying the sheet.

    v. **Sheet.Copy** copies all formulas into the new sheet. **ActiveSheet.UsedRange.Value** is assigned to itself to turn the formulas into text on the new sheet and disconnect them from the original sheet.

    vi. **Sheet.Copy** creates a new sheet.

    vii. **GetValidSheetName** is an add-in function that assigns a sheet name based on the first compound name on the sheet and verifies that the name is a valid tab name (because compound names can contain invalid characters).

    viii. **CreateImage** is an add-in function that loads an image into the spreadsheet. CreateImage grabs the image byte array from the XML file, converts the byte array into a bitmap image, puts the image on the designated named range, and resizes or crops the image to make it fit.

    ix. Each sheet is copied when it is filled with 12 compounds. The last page needs special handling if it is not completely filled.

    x. The code makes sure that “No data in the report” is shown if there is no data to show.

    xi. The application deletes Sheet1 that acted as a template during this process and deletes tempSheet because it is not needed after processing.

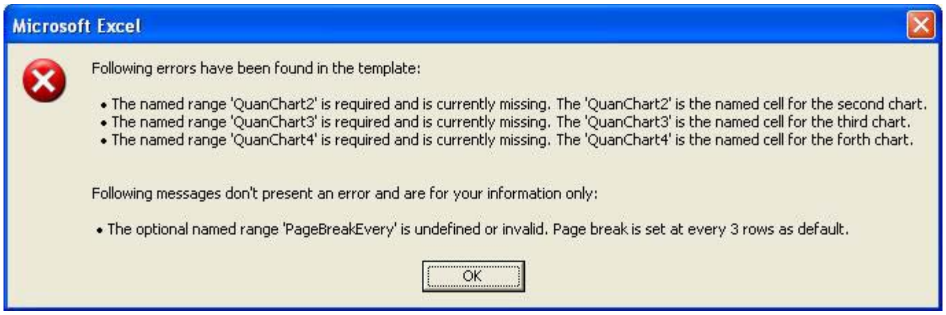
- **GetPageFooterText** returns the flag legend to be used as a page footer.

  - **SkipSetupPageHeaderFooter** is used because the footer text is the same on every page.

  - The legend visibility depends on the setting of **ShowFlagsAndLegend**, which is retrieved directly from the XML source file using the **GetXmlElement** add-in function.

- **Validate** checks these named ranges:

- **CheckRequiredName** is an add-in function that checks if the required named range exists in the workbook. You can specify the error message if the named range is missing.
- **CheckOptionalNameOutsideOf** is an add-in function that checks if a named range is defined outside another named range. If not, a warning message specifies the default value.
- The error messages are accumulated for all errors and warnings and are displayed in a single dialog box, for example:



Following errors have been found in the template:

- The named range 'QuanChart2' is required and is currently missing. The 'QuanChart2' is the named cell for the second chart.
- The named range 'QuanChart3' is required and is currently missing. The 'QuanChart3' is the named cell for the third chart.
- The named range 'QuanChart4' is required and is currently missing. The 'QuanChart4' is the named cell for the forth chart.

Following messages don't present an error and are for your information only:

- The optional named range 'PageBreakEvery' is undefined or invalid. Page break is set at every 3 rows as default.

3. To use the Visual Basic compile function to check for errors, choose **Debug > Compile VBAProject**.

Testing the Template

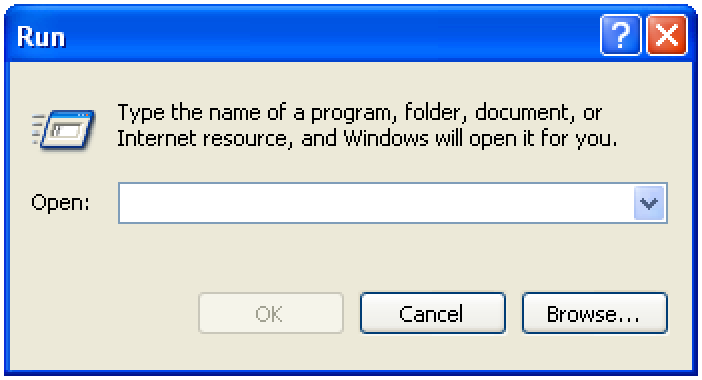
You are now ready to run a test from the Windows **Start** > **Run** box. In this test, the code that you enter does the following:

- Opens the Excel application.
- Creates a new workbook using the MyHighDensitySampleReport1Long.xltm template.
- Passes the parameters to the template.
- Generates a report based on the 4bw01_SampleCentricData.xml source file.
- Saves the report to the MyHighDensitySampleReport1Long.xlsm file.

❖ To run the test

1. Choose **Start** > **Run**.

   The Run dialog box opens.

   Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

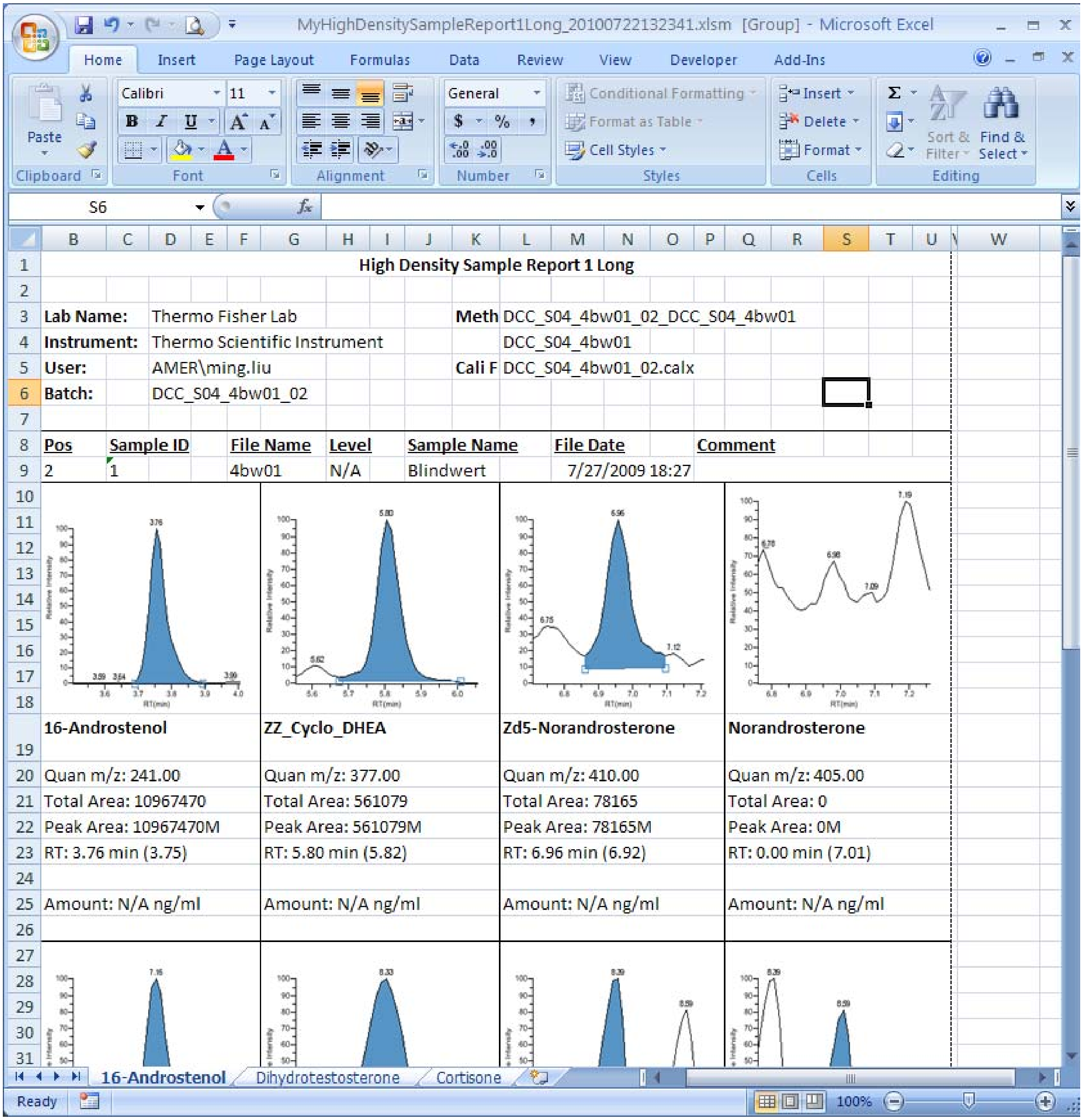
2. Enter this code in the Open box:

   ```
   Excel.exe /n “C:\Thermo\Custom Report
   Tutorials\MyHighDensitySampleReport1Long.xltm”
   /x/C:\Thermo\Custom|Report|Tutorials\Data\4bw01_SampleCentricData.xml
   /C:\Thermo\Custom|Report|Tutorials\Reports\MyHighDensitySampleReport1
   Long.xlsm/s/h
   ```

   For detailed descriptions of the parameters used in this code, see “Testing the Template” on page 25.

The generated report that appears on your screen is similar to the following figure (actual data may vary depending on the batch you used).

Figure 53. Generated report



High Density Sample Report 1 Long

Lab Name: Thermo Fisher Lab
Instrument: Thermo Scientific Instrument
User: AMER\ming.liu
Batch: DCC_S04_4bw01_02

Meth DCC_S04_4bw01_02_DCC_S04_4bw01
DCC_S04_4bw01
Cali F DCC_S04_4bw01_02.calx

Pos	Sample ID	File Name	Level	Sample Name	File Date	Comment
2	1	4bw01	N/A	Blindwert	7/27/2009 18:27	

16-Androstenol	ZZ_Cyclo_DHEA	Zd5-Norandrosterone	Norandrosterone
Quan m/z: 241.00	Quan m/z: 377.00	Quan m/z: 410.00	Quan m/z: 405.00
Total Area: 10967470	Total Area: 561079	Total Area: 78165	Total Area: 0
Peak Area: 10967470M	Peak Area: 561079M	Peak Area: 78165M	Peak Area: 0M
RT: 3.76 min (3.75)	RT: 5.80 min (5.82)	RT: 6.96 min (6.92)	RT: 0.00 min (7.01)
Amount: N/A ng/ml	Amount: N/A ng/ml	Amount: N/A ng/ml	Amount: N/A ng/ml

You are now ready to integrate the template into the TraceFinder application.

❖ To use the template in the TraceFinder application

1. Copy your new template into the dedicated folder of the application:

   C:\Thermo\Shared\Templates\Reports

2. Use this template the same way as the other delivered reports in the TraceFinder application.

   For detailed information about how to use custom reports, refer to the *TraceFinder User Guide*.

Creating a Calibration Report

In this chapter, you will create an Excel template for a calibration report similar to the Calibration Report delivered with the TraceFinder application.

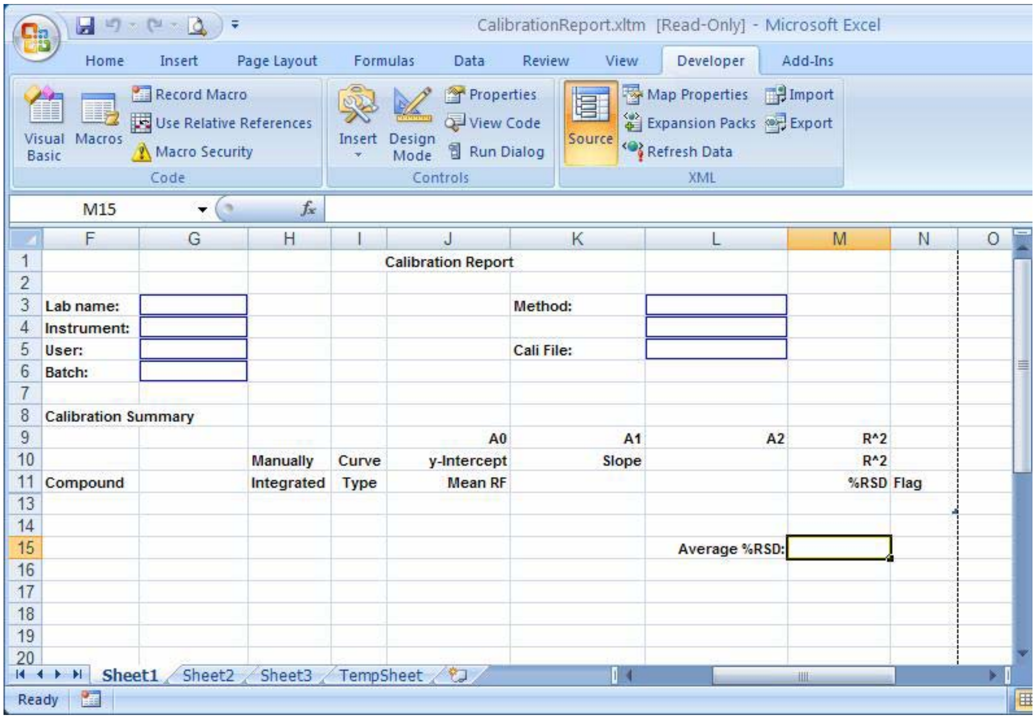
Contents

A calibration report in the TraceFinder application is a batch-level report that presents the calibration results of the batch. The calibration report includes the following three sections:

- Calibration summary data
- Calibration data points for each compound
- A list of calibration files

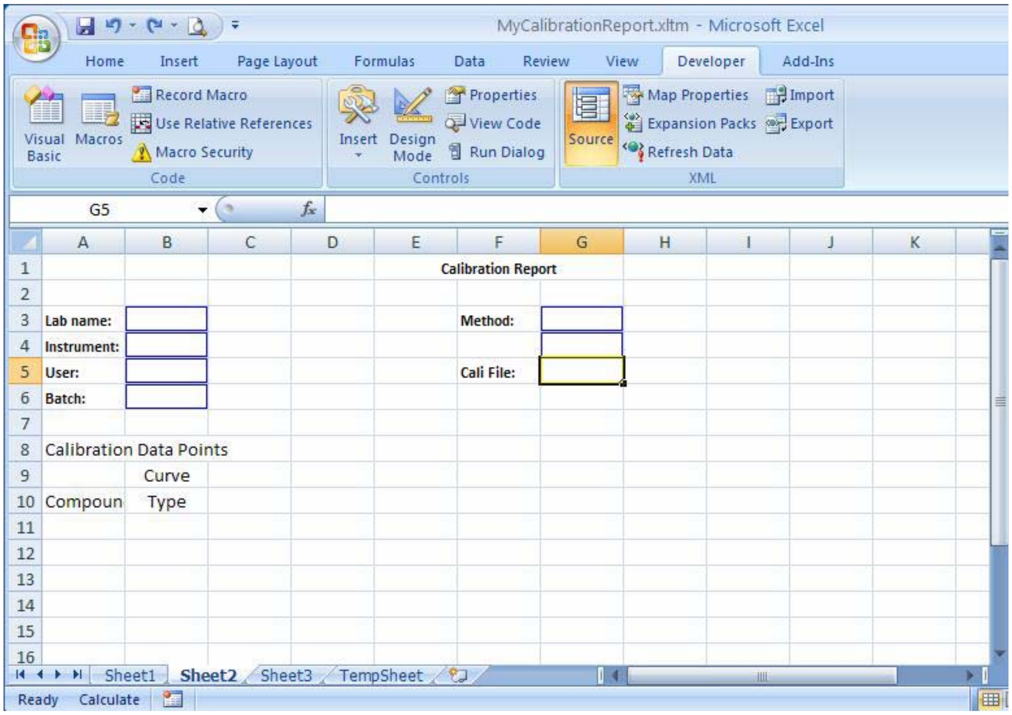
Each section of the report is presented in a separate sheet in an Excel spreadsheet.

Figure 54. Template for Sheet1 - Calibration summary data



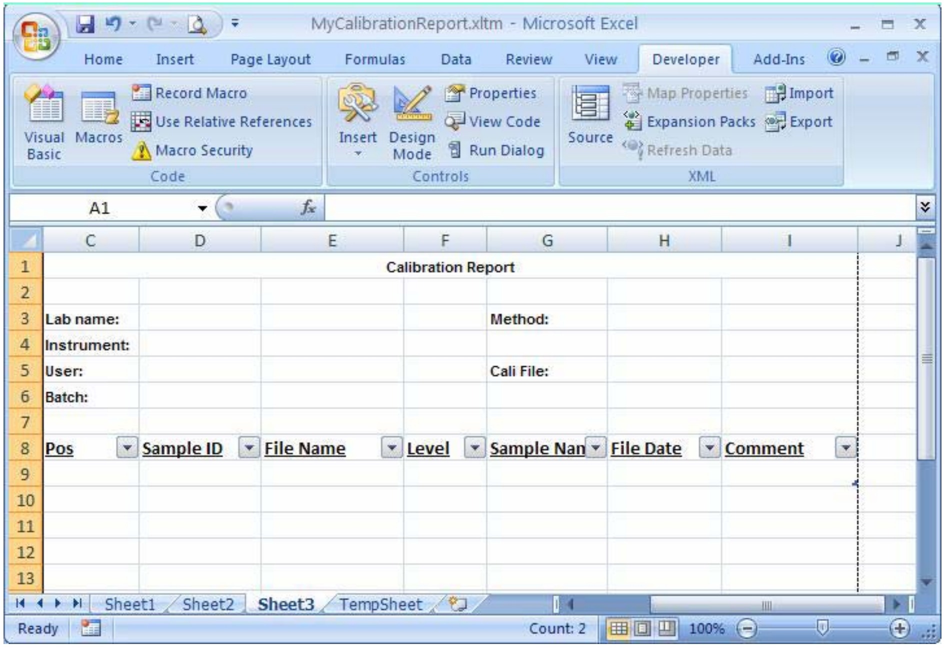
	F	G	H	I	J	K	L	M	N	O
1					Calibration Report					
2										
3	Lab name:					Method:				
4	Instrument:									
5	User:					Cali File:				
6	Batch:									
7										
8	Calibration Summary									
9					A0	A1	A2	R^2		
10			Manually	Curve	y-Intercept	Slope		R^2		
11	Compound		Integrated	Type	Mean RF			%RSD	Flag	
13										
14										
15							Average %RSD:			
16										
17										
18										
19										
20										

Figure 55. Template for Sheet2 - Calibration data points



	A	B	C	D	E	F	G	H	I	J	K
1						Calibration Report					
2											
3	Lab name:					Method:					
4	Instrument:										
5	User:					Cali File:					
6	Batch:										
7											
8	Calibration Data Points										
9		Curve									
10	Compoun	Type									
11											
12											
13											
14											
15											
16											

Figure 56. Template for Sheet3 - Calibration files



	C	D	E	F	G	H	I
1				Calibration Report			
2							
3	Lab name:				Method:		
4	Instrument:						
5	User:				Cali File:		
6	Batch:						
7							
8	Pos	Sample ID	File Name	Level	Sample Nan	File Date	Comment

As you create a calibration report template, you will learn the following:

- How to work with a data table to add, insert, remove, or move a column
- How to use the same field multiple times to work around the limitation that a data field can be referenced only once
- How to create a cross-tab (pivot table) report
- How to filter data for display
- How to use derived columns and rows

Before you begin creating a template for any report, identify the data points on the report and understand how they map to the nodes in the XML Source.

Opening the Report Template

In these step-by-step procedures, you will use the ThermoCustomReportBaseTemplate.xltm file that you created in the Getting Started section as the basic template.

❖ To begin your high density report template

In Windows, make a copy of the ThermoCustomReportBaseTemplate.xltm file and name the new copy **MyCalibrationReport.xltm**.

❖ To load the template file

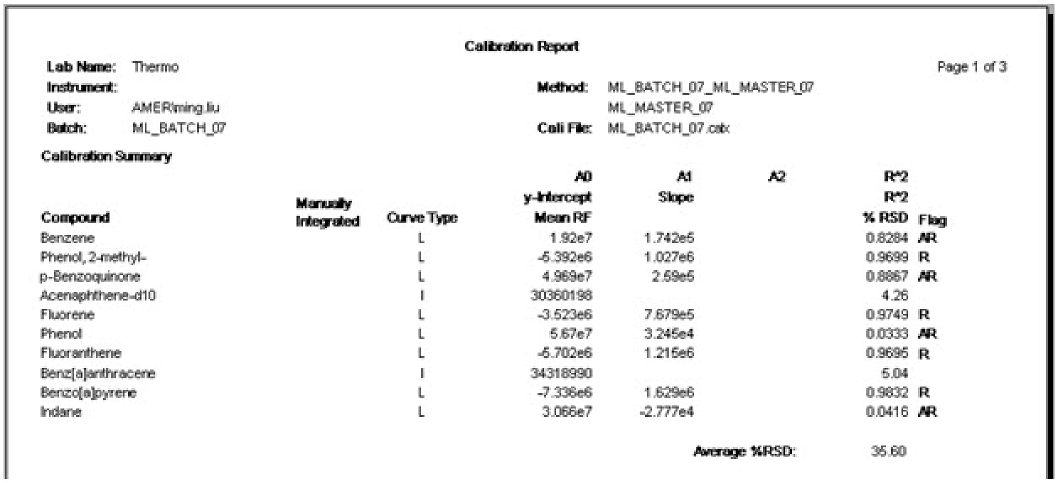
1. Open the Excel application.
2. Click the **Excel application** icon, , and choose **Open**.
3. In the Open dialog box, double-click **MyCalibrationReport.xltm**.

> **Tip** Do not open this template file from Windows Explorer because doing so creates a new spreadsheet based on this template instead of opening the template for editing.

Creating a Calibration Summary – Sheet1

The calibration summary page in a calibration report contains a list of summary data, as shown in the following figure.

Figure 57. Calibration summary



Calibration Report

Lab Name: Thermo

Page 1 of 3

Instrument:

Method: ML_BATCH_07_ML_MASTER_07
ML_MASTER_07

User: AMER\ming.liu

Batch: ML_BATCH_07

Cali File: ML_BATCH_07.calx

Calibration Summary

			A0	A1	A2	R^2	
	Manually		y-Intercept	Slope		R^2	
Compound	Integrated	Curve Type	Mean RF			% RSD	Flag
Benzene		L	1.92e7	1.742e5		0.8284	AR
Phenol, 2-methyl-		L	-5.392e6	1.027e6		0.9699	R
p-Benzoquinone		L	4.969e7	2.59e5		0.8867	AR
Acenaphthene-d10		I	30360198			4.26	
Fluorene		L	-3.523e6	7.679e5		0.9749	R
Phenol		L	5.67e7	3.245e4		0.0333	AR
Fluoranthene		L	-5.702e6	1.215e6		0.9695	R
Benz[a]anthracene		I	34318990			5.04	
Benzo[a]pyrene		L	-7.336e6	1.629e6		0.9832	R
Indane		L	3.066e7	-2.777e4		0.0416	AR

Average %RSD: 35.60

Where:

- **Compound** is the compound name.
- **Manually Integrated** is flagged as **M** when the curve is manually integrated or blank otherwise.
- **Curve Type** indicates one of the available curve types:
  - L (Linear)
  - Q (Quadratic)
  - A (Average RF)
  - I (Internal Standard, which has no curve)
- **A0/y-Intercept/Mean RF** is a multi-purpose column. Its value depends on the curve type.
- **A1/Slope** is a multi-purpose column. Its values depends on the curve type.
- **A2** is used for quadratic curve **types** only.
- **R^2/R^2/%RSD** is a multi-purpose column.
- **Flag** is a calibration flag.

About Multipurpose Columns

The content of a multi-purpose column depends on the curve type settings.

- For a Linear curve type, the curve equation is y = A0 + A1 * x, where the A0 column is the y-intercept and the A1 column is the slope. The A2 column is empty.
- For a Quadratic curve type, the curve equation is y = A0 + A1 * x + A2 * x^2. All three columns are used.
- For an Average RF type curve, the curve for each compound has a single value. Only the first column is used. The average RF value depends on the Standard Type settings.
  - For external Average RF:
    Amount = (Response)/(Mean Response Factor)
  - For internal Average RF:
    Amount = (Response * ISTD Amount)/(ISTD Response * Mean Response Factor)
- For a linear or quadratic curve, the fourth column shows the R^2 value. For an Average RF curve, the fourth column shows a %RSD value.

All the individual data points needed in this report are exported in the XML file. Putting the logic code into the spreadsheet (and the VBA code) can be difficult to maintain. For this reason, you exported extra massaged data points ready for the custom report to use.

The CurveSummaryValue1, CurveSummaryValue2, CurveSummaryValue3, and CurveSummaryValue4 columns in the SampleCentricExportBatch/CalibrationData/ CompoundCurveExportData folder represent this type of shared column.

Creating the Report Headers

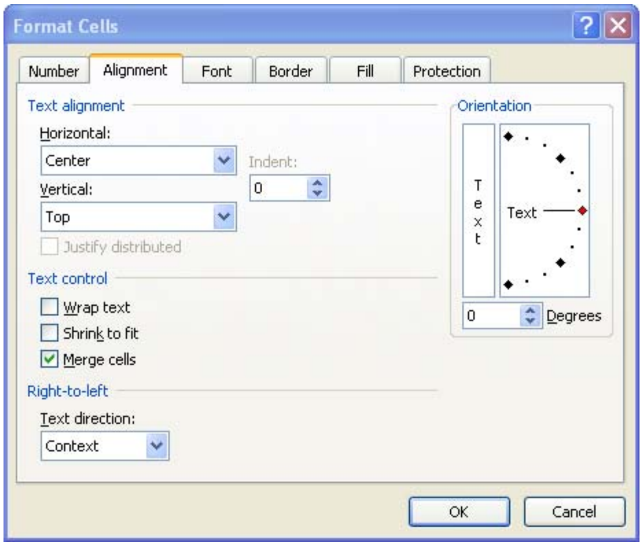
In this section, you will create static labels for the report header and then drag the data nodes from the XML Source tree.

❖ To enter static labels for the report heading

1. Type **Calibration Report** in cell B1.
2. Select the cells **B1** through **L1**.
3. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

Figure 58. Format Cells dialog box - Alignment page



Horizontal: Center

Indent: 0

Vertical: Top

□ Justify distributed

□ Wrap text

□ Shrink to fit

☑ Merge cells

Text direction: Context

0 Degrees

4. Click the **Alignment** tab.

5. In the Horizontal list, select **Center**.

6. In the Text Control area, select **Merge cells**.

7. Click **OK**.

8. Type **Lab name:** in cell A3.

9. Type **Instrument:** in cell A4.

10. Type **User:** in cell A5.

11. Type **Batch:** in cell A6.

12. Type **Method:** in cell G3.

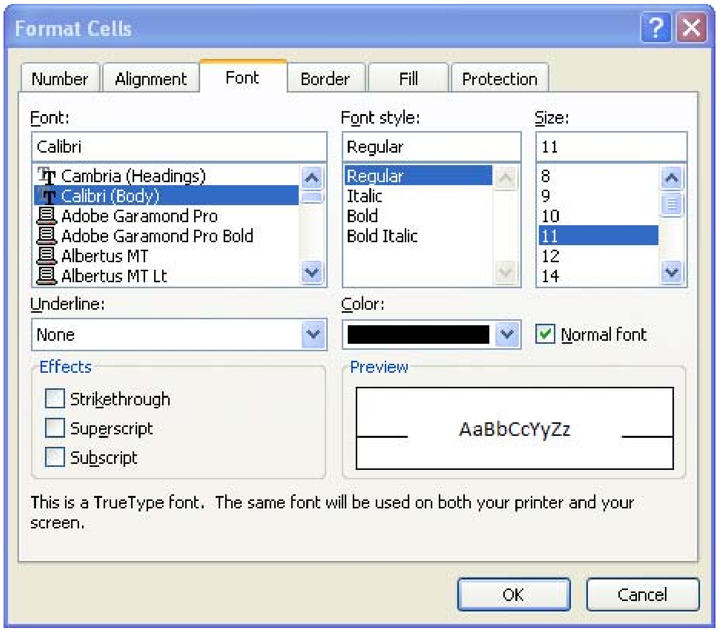
13. Type **Cali File:** in cell G5.

14. Hold down the CTRL key and select the label cells you just created.

15. Right-click and choose **Format Cells** from the shortcut menu.

    The Format Cells dialog box opens.

Figure 59. Format Cells dialog box - Font page



Format Cells

Number | Alignment | Font | Border | Fill | Protection

Font: Calibri
- Cambria (Headings)
- Calibri (Body)
- Adobe Garamond Pro
- Adobe Garamond Pro Bold
- Albertus MT
- Albertus MT Lt

Font style: Regular
- Regular
- Italic
- Bold
- Bold Italic

Size: 11
- 8
- 9
- 10
- 11
- 12
- 14

Underline: None

Color:

☑ Normal font

Effects
- ☐ Strikethrough
- ☐ Superscript
- ☐ Subscript

Preview: AaBbCcYyZz

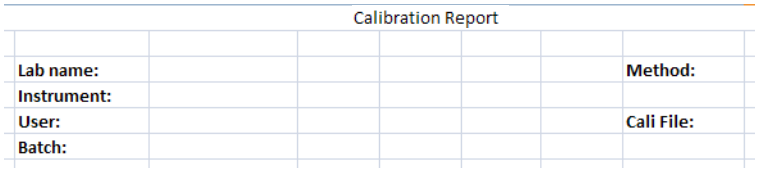
This is a TrueType font. The same font will be used on both your printer and your screen.

16. Click the **Font** tab if it is not already selected.

17. In the Font Style box, select **Bold**.

18. Click **OK** in the Format Cells dialog box.

Your template header looks like this:



Calibration Report

Lab name:						**Method:**
Instrument:						
User:						**Cali File:**
Batch:						

❖ To enter static labels for the column headers

1. Type **Calibration Summary** in cell C8.
2. Type **Compound** in cell C11.
3. Type **Manually Integrated** in cell D10.
4. Type **Curve Type** in cell E10.
5. Type **A0 y-Intercept Mean RF** in cell F9.
6. Type **A1 Slope** in cell G9.
7. Type **A2** in cell H9.
8. Type **R^2 R^2 %RSD** in cell I9.
9. Select cells **C8** through **I10**.

10. Right-click and choose **Format Cells** from the shortcut menu.

    The Format Cells dialog box opens. See "Format Cells dialog box - Font page" on page 96.

11. Click the **Font** tab if it is not already selected.

12. In the Font Style box, select **Bold**.

13. Click **OK** in the Format Cells dialog box.

> **Tip** Before proceeding, make sure the Excel window shows the Developer tab, which is hidden by default. If your Excel window does not show the Developer tab, see Chapter 3, "Getting Started."

Before you begin creating a template for any report, identify the data points on the report and understand how they map to the nodes in the XML Source tree.

> **Note** A "node" in the XML Source tree creates a "field" in the spreadsheet.

❖ To add mapped fields to the template

1. Click the **Developer** tab to display the Developer features.

2. Click the **Source** button,

.

   The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet. You are ready to create mapped fields from the nodes in the SampleCentricExportBatch folder.

3. Open the MethodHeaderData folder and drag the **LabName** node from the XML Source tree to cell B3.

4. Open the MethodHeaderData folder and drag the **InstrumentMethodName** node from the XML Source tree to cell B4.

5. Open the BatchHeaderData folder and drag the **UserName** node from the XML Source tree to cell B5.

6. Open the BatchHeaderData folder and drag the **BatchName** node from the XML Source tree to cell B6.

7. Open the MethodHeaderData folder and drag the **MethodName** node from the XML Source tree to cell G3.

8. Open the MethodHeaderData folder and drag the **MasterMethodName** node from the XML Source tree to cell G4.

9. Open the BatchHeaderData folder and drag the **CalibrationFile** node from the XML Source tree to cell G5.

You are now ready to add a mapped table to your template. Using the Excel application, you can drag the mapped table nodes from the XML Source pane.

❖ To add a mapped table to the template

To create mapped fields from the nodes in the SampleCentricExportBatch/ CalibrationData/CompoundCurveExportData folder, do the following:

a. Drag the **CompoundName** node from the XML Source tree to cell A12.

b. Drag the **CurveSummaryValue1** node from the XML Source tree to cell D12.

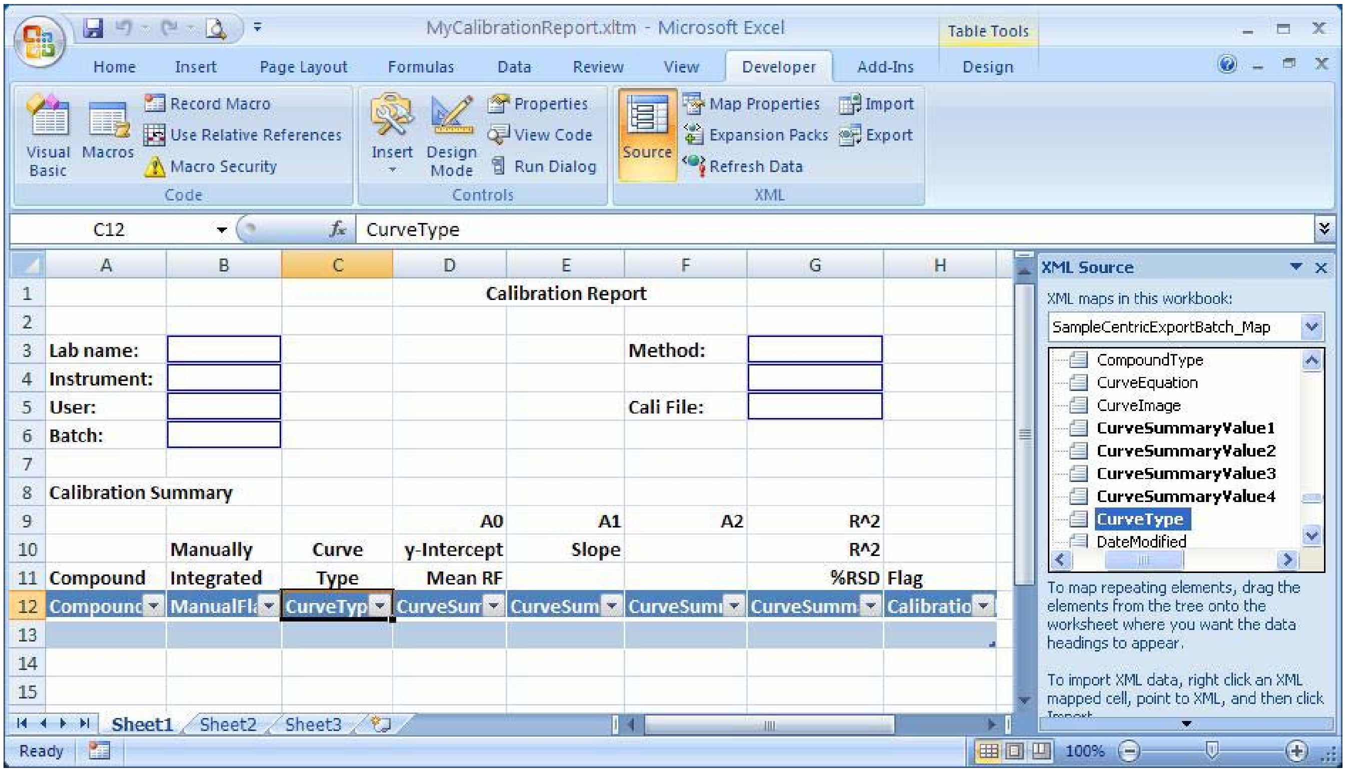
c. Drag the **CurveSummaryValue2** node from the XML Source tree to cell E12.

d. Drag the **CurveSummaryValue3** node from the XML Source tree to cell F12.

e. Drag the **CurveSummaryValue4** node from the XML Source tree to cell G12.

Ensure that your template looks like the following figure.

Figure 60. Template with mapped fields



C12 | CurveType

	A	B	C	D	E	F	G	H
1					Calibration Report			
2								
3	Lab name:					Method:		
4	Instrument:							
5	User:					Cali File:		
6	Batch:							
7								
8	Calibration Summary							
9				A0	A1	A2	R^2	
10		Manually	Curve	y-Intercept	Slope		R^2	
11	Compound	Integrated	Type	Mean RF			%RSD	Flag
12	Compound	ManualFl	CurveTyp	CurveSum	CurveSum	CurveSum	CurveSumm	Calibratio
13								
14								
15								

XML Source

XML maps in this workbook:

SampleCentricExportBatch_Map

- CompoundType
- CurveEquation
- CurveImage
- **CurveSummaryValue1**
- **CurveSummaryValue2**
- **CurveSummaryValue3**
- **CurveSummaryValue4**
- **CurveType**
- DateModified

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped cell, point to XML, and then click Import

Adding Helper Columns as Derived Columns

In this section, you will add two helper columns, CompoundType and HasActiveReplicates, as derived columns. Some columns in the data table depend on the CompoundType and HasActiveReplicates columns, which are not visible in the generated report.

❖ To add two new columns to the template

1. Right-click the column A header cell and choose **Insert** from the shortcut menu.
2. Repeat Step 1 to add another new column.

   Starting from column C, the Excel application shifts all cells to the right.
3. Open the SampleCentricExportBatch/CalibrationData/CompoundCurveExportData folder and drag the **CompoundType** node from the XML Source tree to cell B12.
4. Open the SampleCentricExportBatch/CalibrationData/CompoundCurveExportData folder and drag the **HasActiveReplicates** node from the XML Source tree to cell A12.

Later, you will hide these columns and use formulas to create massaged values for display.

You are now ready to add mapped fields to the template.

❖ To map the derived columns in the template

To add mapped fields from the SampleCentricExportBatch/CalibrationData/CompoundCurveExportData folder, do the following:

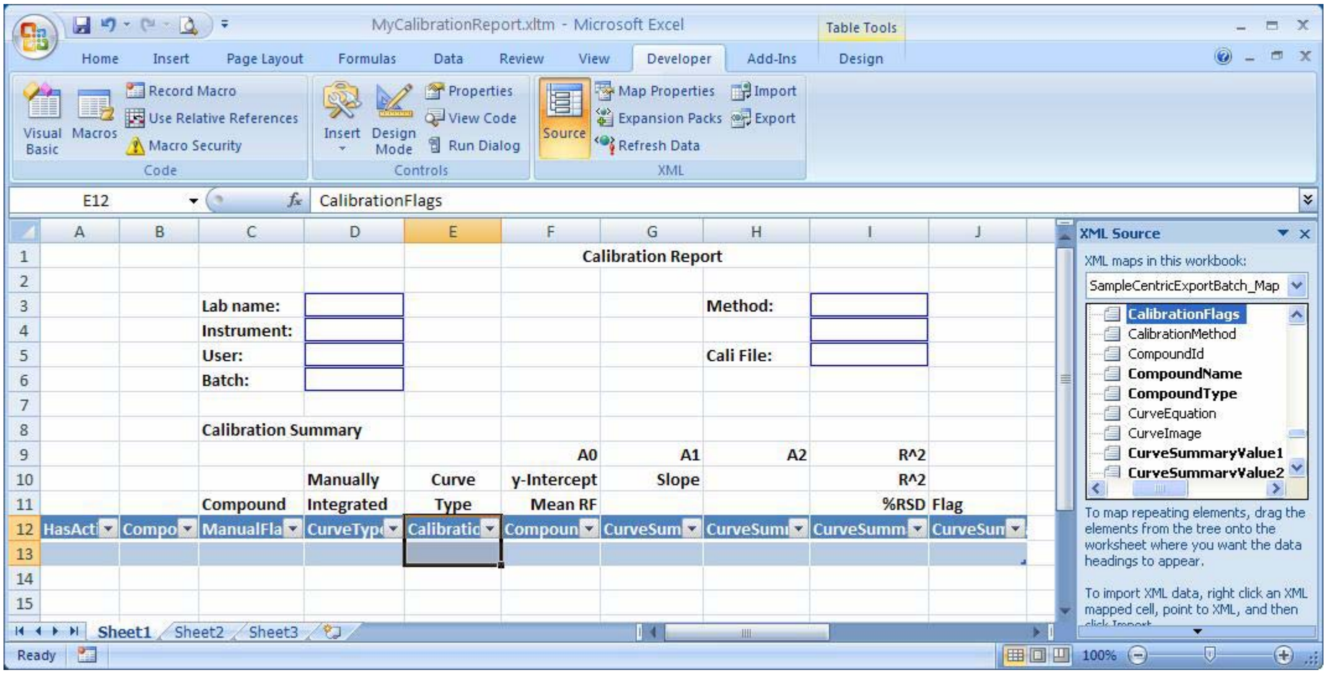
a. Drag the **ManualFlagSet** node from the XML Source tree to cell C12.

b. Drag the **CurveType** node from the XML Source tree to cell D12.

c. Drag the **CalibrationFlags** node from the XML Source tree to cell E12.

Ensure that your template looks like the following figure.

Figure 61. Template with mapped table



E12 | CalibrationFlags

	A	B	C	D	E	F	G	H	I	J
1							Calibration Report			
2										
3			Lab name:					Method:		
4			Instrument:							
5			User:					Cali File:		
6			Batch:							
7										
8			Calibration Summary							
9						A0	A1	A2	R^2	
10				Manually	Curve	y-Intercept	Slope		R^2	
11			Compound	Integrated	Type	Mean RF			%RSD	Flag
12	HasActi	Compo	ManualFla	CurveTyp	Calibratic	Compoun	CurveSum	CurveSumi	CurveSumm	CurveSun
13										
14										
15										

XML Source

XML maps in this workbook:

SampleCentricExportBatch_Map

- CalibrationFlags
- CalibrationMethod
- CompoundId
- CompoundName
- CompoundType
- CurveEquation
- CurveImage
- CurveSummaryValue1
- CurveSummaryValue2

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped cell, point to XML, and then click Import

Note Do not worry that the static headers and mapped fields are not correctly aligned. In the next section, you will move the static headers to the correct columns.

❖ To hide the helper columns in the template

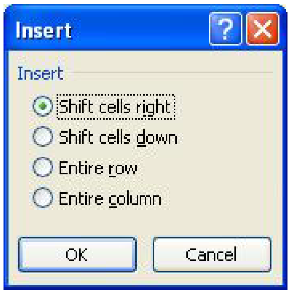
1. Select the cells **A1** through **A11**.

   Select only these cells; do not select the entire column.

2. Right-click and choose **Insert** from the shortcut menu.

   The Insert dialog box opens.

   Figure 62. Insert dialog box

   Insert

   Insert
   - ● Shift cells right
   - ○ Shift cells down
   - ○ Entire row
   - ○ Entire column

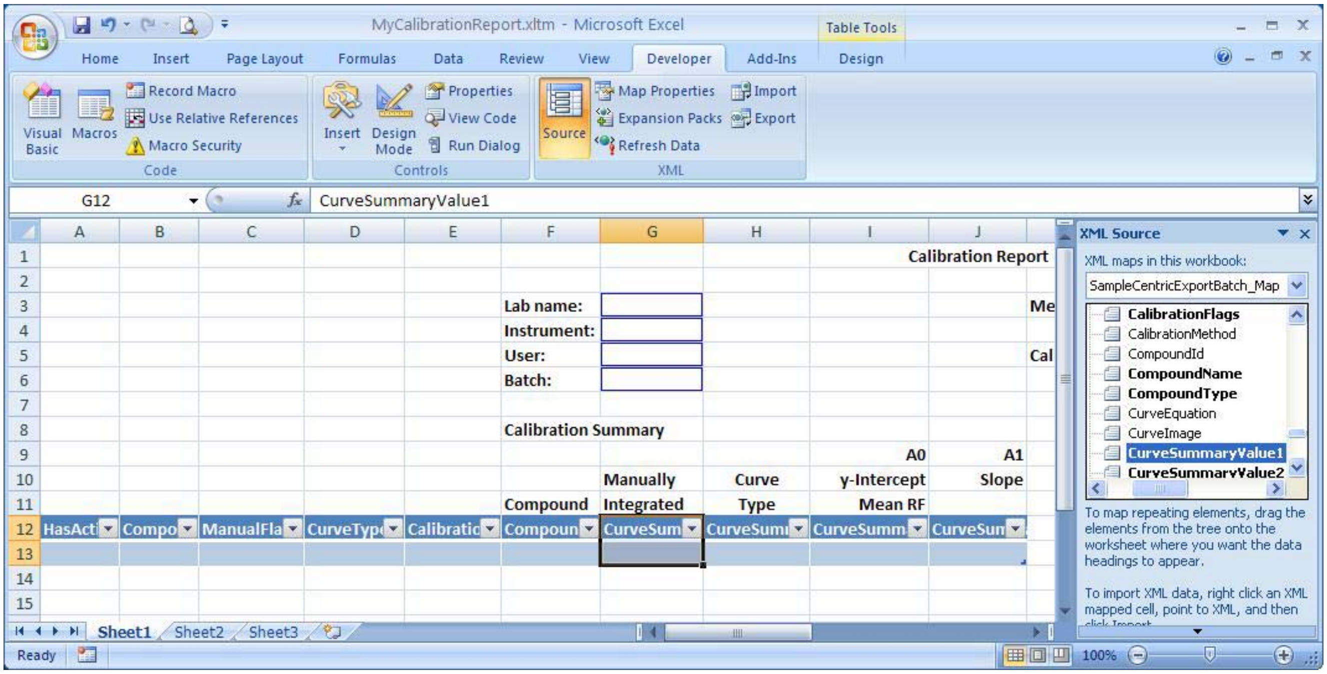
3. Select the **Shift cells right** option and click **OK**.

4. Repeat steps 1 through 3 two more times.

You now have a total of five empty columns: the three you added plus the two that were already there. The columns are empty only in cells 1 through 11.

Ensure that your template looks like the following figure.

Figure 63. Template with five empty columns



❖ To add three new columns to the template

1. Right-click the G12 cell and choose **Insert > Table Columns to the Left** from the shortcut menu.

   The Excel application adds a new column with the default name Column1 to the left of the G column.

1. Right-click the G12 cell and choose **Insert > Table Columns to the Left** from the shortcut menu.

   The Excel application adds a new column with the default name Column2 to the left of the G column.

2. Right-click the L12 cell and choose **Insert > Table Column to the Right** from the shortcut menu.

   The Excel application adds a new column with the default name CurveSummaryValue5 (based on the previous column name) to the right of the L column.

You now have three empty columns to use for Manually Integrated, Curve Type, and Flag.

When you add a new column to the data table, the cell format defaults to Text. In this format, the formula you enter into the cell appears as text. As text, the formula will not expand when data is imported into data table.

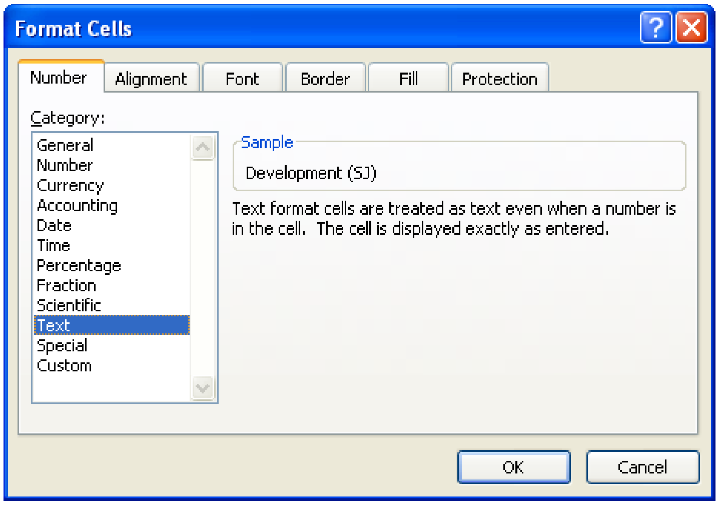
You are now ready to manually change the cell formats from Text to General.

❖ To change the cell format in the template

1. Right-click the Column1 cell and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

   Figure 64. Format Cells dialog box - Number page

   Sample
   Development (SJ)

   Text format cells are treated as text even when a number is in the cell. The cell is displayed exactly as entered.

2. Click the **Number** tab if it is not already selected.

3. Select **General** in the Category list and click **OK**.

4. While the Column1 cell is still selected, press ENTER.

   The Excel application does not automatically update the new cell format. Pressing ENTER manually updates it.

5. Repeat steps 1-5 for each of the three columns that you added.

In the next procedure, you will give the column headers meaningful names. Because later you will hide the headers, this procedure is optional.

❖ To rename the column headers in the template

1. In cell G10, change Column1 to **Manually Integrated**.

2. In cell H10, change Column2 to **Curve Type**.

3. In cell M11, change CurveSummaryValue5 to **Flag**.

You are now ready to enter the formulas for the columns.

❖ To enter the formulas in the template

1. In cell G13, enter the following formula:

   ```
   =IF(C13, "M", "")
   ```

2. In cell H13, enter the following formula:

   ```
   =IF(B13="Internal Standard", "I", LEFT(D13, 1))
   ```

3. In cell M13, enter the following formula:

   ```
   =IF(A13, IF(E13=0, "", E13), "")
   ```

The field below the R^2 column is for the Average %RSD value, which is the average of all RSD values. The result is also exported into the XML file under the SampleCentricExportBatch/CalibrationSummaryData/AverageRSD node. The Excel data table automatically pushes the Average %RSD row down when it expands the data table.

You are now ready to hide the helper columns and rows.

❖ To hide the helper columns and rows

1. Select columns **A** through **E**.

2. Right-click and choose **Hide** from the shortcut menu.

3. Right-click the number **12** in the first column of row 12 and choose **Hide** from the shortcut menu.

   The Excel application hides row 12. At any time, you can restore the contents of a hidden row.

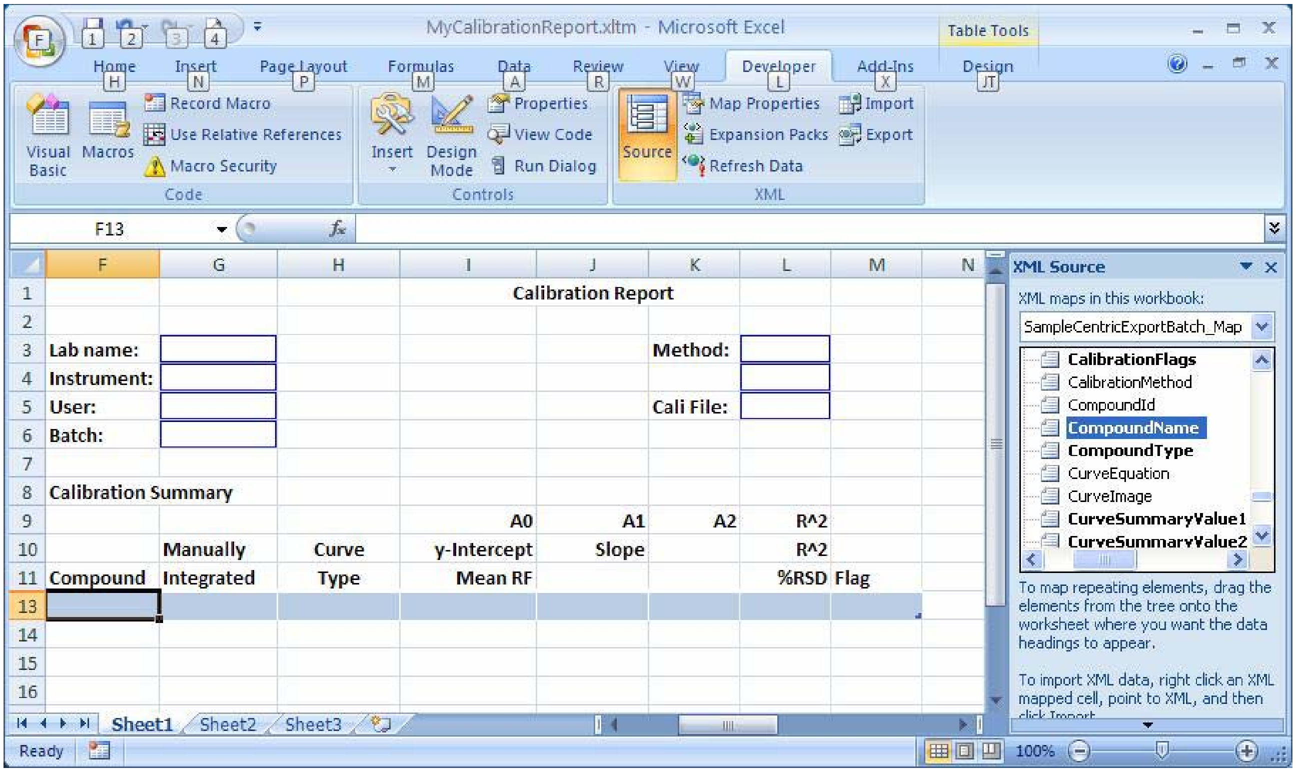
   To restore row 12, do the following:

   a. Select rows **11** through **13**.

   b. Right-click and choose **Unhide** from the shortcut menu.

Ensure that your template looks like the following figure.

Figure 65. Completed Sheet1



	F	G	H	I	J	K	L	M	N
1					Calibration Report				
2									
3	Lab name:					Method:			
4	Instrument:								
5	User:					Cali File:			
6	Batch:								
7									
8	Calibration Summary								
9				A0	A1	A2	R^2		
10		Manually	Curve	y-Intercept	Slope		R^2		
11	Compound	Integrated	Type	Mean RF			%RSD	Flag	
13									
14									
15									
16									

XML maps in this workbook:

SampleCentricExportBatch_Map

- CalibrationFlags
- CalibrationMethod
- CompoundId
- CompoundName
- CompoundType
- CurveEquation
- CurveImage
- CurveSummaryValue1
- CurveSummaryValue2

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped cell, point to XML, and then click Import.

Sheet1 of the template is ready to run.

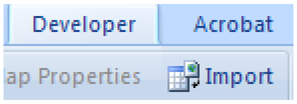
You are now ready to directly import an XML file to test the data table. You will not save this data; this is a test to see how the imported data will look in your report.

❖ To import the XML file into the template

1. Save your template.

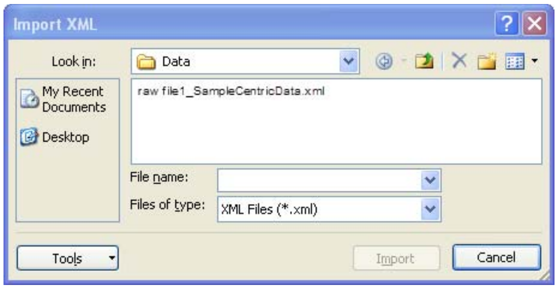
   There is no undo for the Import function, so it is best to save your template in case you make a mistake, for example, importing the wrong .xml file.

2. Click the **Developer** tab and then click the **Import** button.

   The Import XML dialog box opens.

Figure 66. Import XML dialog box



Look in: Data

raw file1_SampleCentricData.xml

File name:

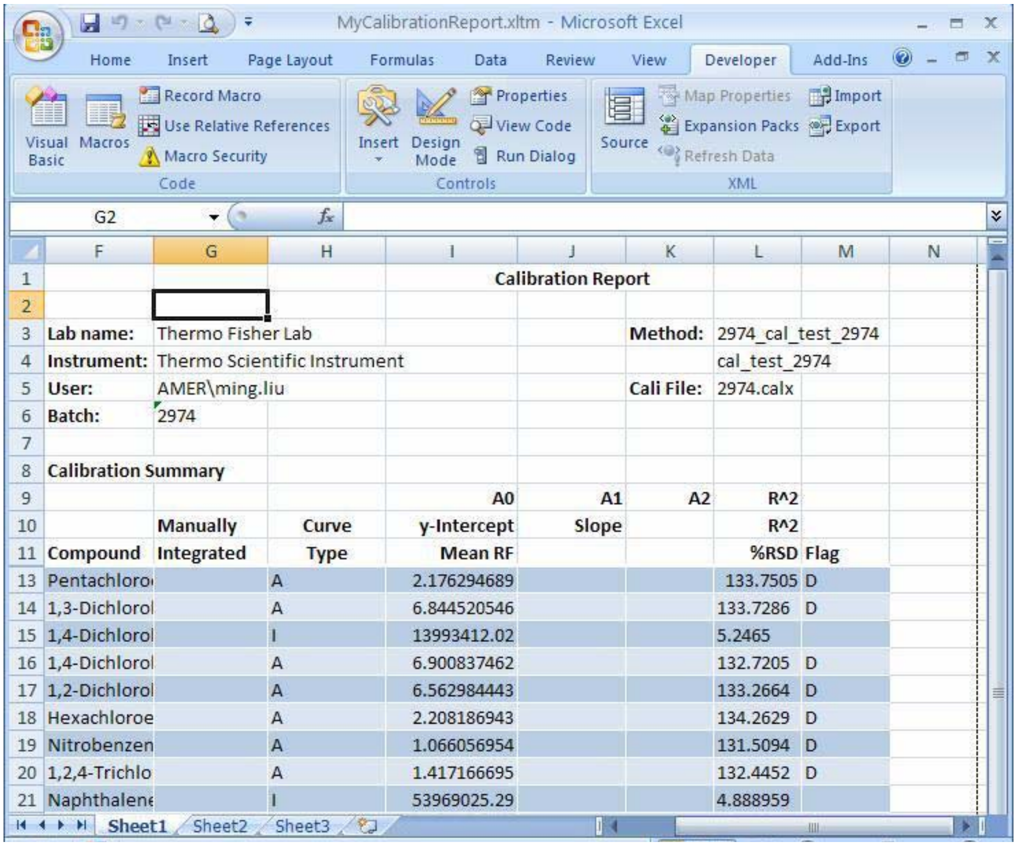
Files of type: XML Files (*.xml)

3. Select your sample-centric data file, and click **Import**.

   The Excel application imports the .xml file into your spreadsheet.

   Ensure that your template looks like the following figure.

Figure 67. Template with imported data



	F	G	H	I	J	K	L	M	N
1				Calibration Report					
2									
3	Lab name:	Thermo Fisher Lab				Method:	2974_cal_test_2974		
4	Instrument:	Thermo Scientific Instrument					cal_test_2974		
5	User:	AMER\ming.liu				Cali File:	2974.calx		
6	Batch:	2974							
7									
8	Calibration Summary								
9				A0	A1	A2	R^2		
10		Manually	Curve	y-Intercept	Slope		R^2		
11	Compound	Integrated	Type	Mean RF			%RSD	Flag	
13	Pentachloro		A	2.176294689			133.7505	D	
14	1,3-Dichlorol		A	6.844520546			133.7286	D	
15	1,4-Dichlorol		I	13993412.02			5.2465		
16	1,4-Dichlorol		A	6.900837462			132.7205	D	
17	1,2-Dichlorol		A	6.562984443			133.2664	D	
18	Hexachloroe		A	2.208186943			134.2629	D	
19	Nitrobenzen		A	1.066056954			131.5094	D	
20	1,2,4-Trichlo		A	1.417166695			132.4452	D	
21	Naphthalene		I	53969025.29			4.888959		

4. Discard this version of the template and open the one you saved before the Import function.

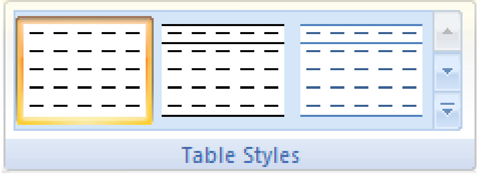
   Leaving imported data in the template does not cause any harm to generated reports because the data is overwritten at run time.

   If you have a problem, see "How Do I Remove Data from the Template After Importing It?" on page 186.

❖ To change the background in the template

1. Click the header of the data table to display the Design tab.
2. Click the **Design** tab.

   The Design tab displays the Table Styles functions.

3. Expand the Table Styles gallery and click **Clear**, Clear , in the lower left corner of the gallery.

   The background changes to a plain background.

The Compound Name column is so narrow that the compound names are not entirely visible. Cell text can run into the next cell as long as the next cell is empty. You are now ready to introduce an empty column after the Compound Name column.

> **Note** Making the entire column wider is not a solution because it affects the alignment of the cells above the data table.

❖ To enlarge the Compound Name column

1. Insert an empty column by right-clicking the Compound Name column header and choosing **Insert > Table Columns to the Left** from the shortcut menu.

   The Excel application inserts an empty, unmapped column in the table. By default, the new column is named Column1.

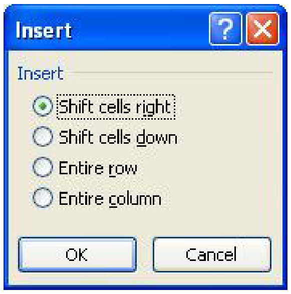
2. Select the cells **G1** through **G11**.

   Select only these cells; do not select the entire column.

3. Right-click and choose **Insert** from the shortcut menu.

   The Insert dialog box opens.

   Figure 68. Insert dialog box

   Insert

   Insert
   - ● Shift cells right
   - ○ Shift cells down
   - ○ Entire row
   - ○ Entire column

4. Select the **Shift cells right** option and click **OK**.

5. Select cells **G3** through **G6**.

6. Right-click and choose **Cut** from the shortcut menu.

7. Right-click in cell F3 and choose **Paste** from the shortcut menu.

   Moving the text in these header columns to the left prevents them from overrunning the page.

❖ To align the header text in the template

1. Select the **Manually Integrated** and **Curve Type** header cells.

2. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See "Format Cells dialog box - Alignment page" on page 95.

3. Click the **Alignment** tab.

4. In the Horizontal list, select **Center**.

5. Click **OK**.

The Mean RF and %RSD cells required a number format of three decimal places. You are now ready to change the data type of these cells.

❖ To change the data type of the Mean RF and %RSD cells

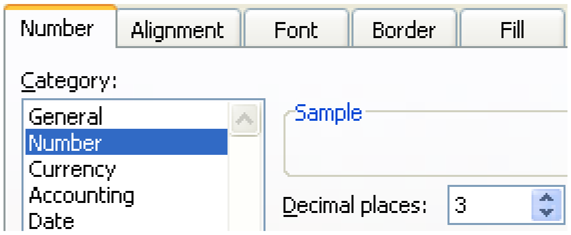
1. Select the cells from **J13** through **M13**.

   Select only these cells; do not select the entire column.

2. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See "Format Cells dialog box - Number page" on page 102.

3. Click the **Number** tab.



Decimal places: 3

4. In the Category list, select **Number**.

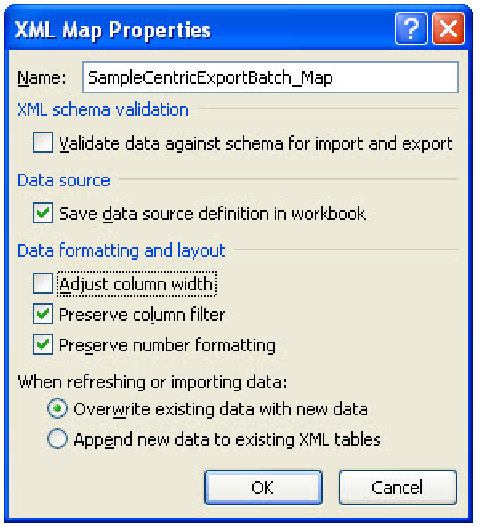
5. In the Decimal Places box, change the value to **3**.

❖ To save your layout design

1. Click the **Developer** tab, then click **Map Properties**.

The XML Map Properties dialog box opens with the first map automatically selected.

Figure 69. XML Map Properties dialog box



XML Map Properties

Name: SampleCentricExportBatch_Map

XML schema validation

□ Validate data against schema for import and export

Data source

☑ Save data source definition in workbook

Data formatting and layout

□ Adjust column width

☑ Preserve column filter

☑ Preserve number formatting

When refreshing or importing data:

● Overwrite existing data with new data

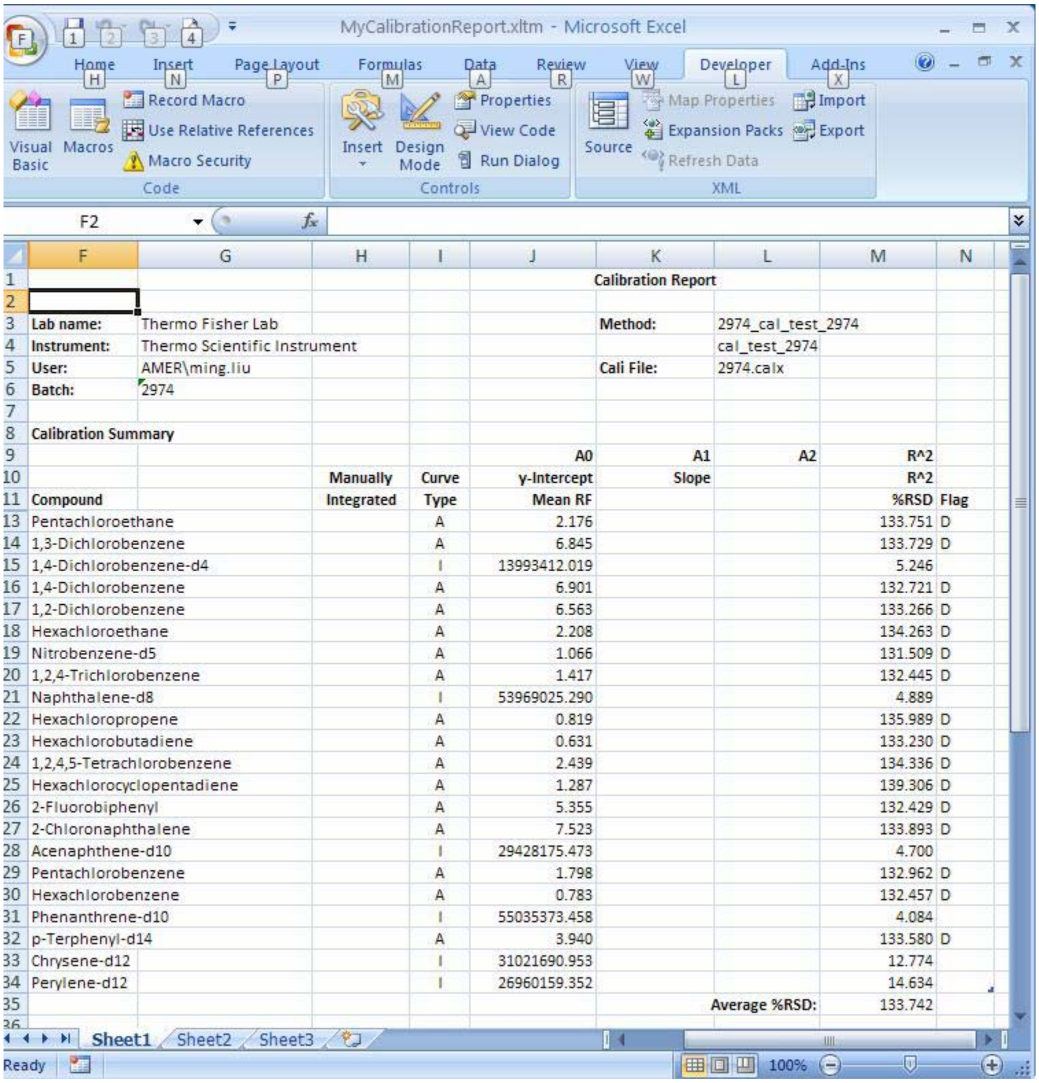
○ Append new data to existing XML tables

2. Clear the **Adjust Column Width** check box.

   When this option is selected, the field expands to fit imported data, which changes your layout.

You can optionally resize the columns to make them look best on the page. If you reimport the data, you template should look like the following figure.

Figure 70. Template with finalized layout



	F	G	H	I	J	K	L	M	N
1						Calibration Report			
2									
3	Lab name:	Thermo Fisher Lab				Method:	2974_cal_test_2974		
4	Instrument:	Thermo Scientific Instrument					cal_test_2974		
5	User:	AMER\ming.liu				Cali File:	2974.calx		
6	Batch:	2974							
7									
8	Calibration Summary								
9					A0	A1	A2	R^2	
10			Manually	Curve	y-Intercept	Slope		R^2	
11	Compound		Integrated	Type	Mean RF			%RSD	Flag
13	Pentachloroethane			A	2.176			133.751	D
14	1,3-Dichlorobenzene			A	6.845			133.729	D
15	1,4-Dichlorobenzene-d4			I	13993412.019			5.246	
16	1,4-Dichlorobenzene			A	6.901			132.721	D
17	1,2-Dichlorobenzene			A	6.563			133.266	D
18	Hexachloroethane			A	2.208			134.263	D
19	Nitrobenzene-d5			A	1.066			131.509	D
20	1,2,4-Trichlorobenzene			A	1.417			132.445	D
21	Naphthalene-d8			I	53969025.290			4.889	
22	Hexachloropropene			A	0.819			135.989	D
23	Hexachlorobutadiene			A	0.631			133.230	D
24	1,2,4,5-Tetrachlorobenzene			A	2.439			134.336	D
25	Hexachlorocyclopentadiene			A	1.287			139.306	D
26	2-Fluorobiphenyl			A	5.355			132.429	D
27	2-Chloronaphthalene			A	7.523			133.893	D
28	Acenaphthene-d10			I	29428175.473			4.700	
29	Pentachlorobenzene			A	1.798			132.962	D
30	Hexachlorobenzene			A	0.783			132.457	D
31	Phenanthrene-d10			I	55035373.458			4.084	
32	p-Terphenyl-d14			A	3.940			133.580	D
33	Chrysene-d12			I	31021690.953			12.774	
34	Perylene-d12			I	26960159.352			14.634	
35							Average %RSD:	133.742	
36									

The template design is complete. No VBA code is required for this sheet.

Creating a Calibration Data Point – Sheet2

Sheet2 of the calibration report contains a cross-tab (pivot table) list of data that displays the calculated amount for each compound against each calibrator.

To create this sheet in your custom report, use an add-in function for creating cross-tab type reports to expose the parameters you need.

For additional information about creating a pivot table, refer to your Excel documentation.

Creating a Pivot Table

The calculated amount (the pivot data point) for each compound and calibrator depends on these factors:

- Compound type
- Curve type
- Manual flag (manual integration affects how the amount is calculated)
- Sample level (as column header)
- Acquisition date (for sorting purposes)
- Expected retention time (for sorting purposes)
- Exclusion reason (to flag the values)

Calibration reports require a pivot table for indicating flags such as the following:

- "Manually Integrated" for a curve that was manually integrated
- "X" for an excluded point
- "X(ISNF)" for exclusion because the internal standard was not found

The Excel pivot table works only with numbers as pivot data points. In this tutorial, you will reformat the pivot data for those data points to a special value so that the template can treat them differently after the pivot table is created.

- For the Manually Integrated flag, change the data point value to "negative" for manually integrated points. Because your data points are normally positive, you can easily identify those values after the pivoting.
- For the X flag, change the data point value to a large number (for example, **88 987 654 321**). This number must be a valid value, but unlikely to be a real data point.
- For X(ISNF) flag, change the data point value to a different large number (for example, **9 987 654 321**). This number must be a valid value, but unlikely to be a real data point.

Tip These large numbers are based solely on observations. If these suggested numbers happen to be the actual result data of your report, use different numbers.

Using the CreatePivotTable Function

You will use the following CreatePivotTable add-in function to create the pivot table for your report.

```
CreatePivotTable(dataTableName, targetRange, rowFields, columnFields,
valueFields, [pivotTableName])
```

Where:

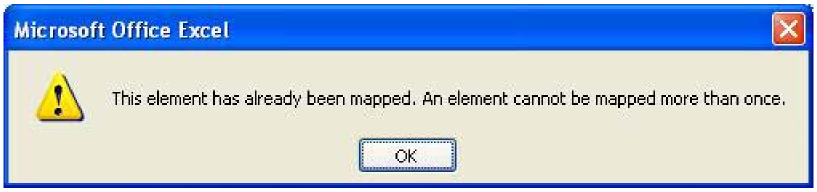
- **dataTableName** is the named range for the data table on which the pivot table is based.
- **targetRange** is the named range for where the pivot table is located on the sheet.
- **rowFields** is a semicolon-delimited string from the Compound Name and Curve Type columns that will become rows in the resulting pivot table.
- **columnFields** is a semicolon-delimited string from the Acquisition Date and Sample Level columns that will become columns in the resulting pivot table.
- **valueFields** is a semicolon-delimited string from the Point Value column that will become pivot data values.
- **pivotTableName** is an optional argument that specifies the name for the resulting pivot table.

This CreatePivotTable add-in function performs the following two tasks:

- Creates the pivot table using the ActiveWorkbook.PivotCaches.Create and ActiveSheet.PivotTables.AddDataField Excel functions.
- Clears the default styles and borders of the pivot table.

You cannot directly use the resulting pivot table in your report because of the format requirements. After running this function, the Excel application creates the pivot table on a temporary sheet. You will use the CopyPaste add-in function to copy the data to the reporting sheet.

In an Excel spreadsheet, you can use each tree node in the XML Source only once. If you try to drag a tree node that is already mapped onto your spreadsheet, you get the following error message:



Microsoft Office Excel

This element has already been mapped. An element cannot be mapped more than once.

To avoid this problem, you can add as many XML maps to the same XML Source file as you need. This generates sequentially numbered XML maps, for example, SampleCentricExportBatch_Map1.

Creating the Sheet2 Template

Create another map in the XML Source pane and use that map to create the report heading.

❖ To add an XML mapping file

1. In the XML Source pane in your spreadsheet, click **XML Maps**, XML Maps... .

   The XML Maps dialog box opens.

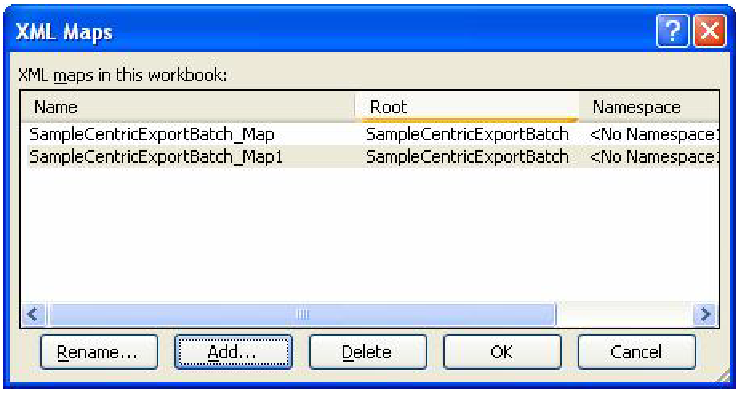
2. Click **Add**.

3. Navigate to your XML mapping file (the same one you used before) and click **Open**.

   The Excel application adds the mapping file to the XML Maps in This Workbook list. Because this is a duplicate of your original mapping file, the number **1** is appended to the file name.

4. Click **OK**.

Figure 71. Multiple XML maps



XML maps in this workbook:

Name	Root	Namespace
SampleCentricExportBatch_Map	SampleCentricExportBatch	<No Namespace
SampleCentricExportBatch_Map1	SampleCentricExportBatch	<No Namespace

Because you have only a single .xml file to import, the add-in component populates all maps when you import the .xml file.

❖ To enter static labels for the report heading

1. Type the report header **Calibration Report** in cell B1.

2. Type **Lab name:** in cell A3.

3. Type **Instrument:** in cell A4.

4. Type **User:** in cell A5.

5. Type **Batch:** in cell A6.

6. Type **Method:** in cell F3.

7. Type **Cali File:** in cell F5.

8. Hold down the CTRL key and select the label cells you just created.

9. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See "Format Cells dialog box - Font page" on page 96.

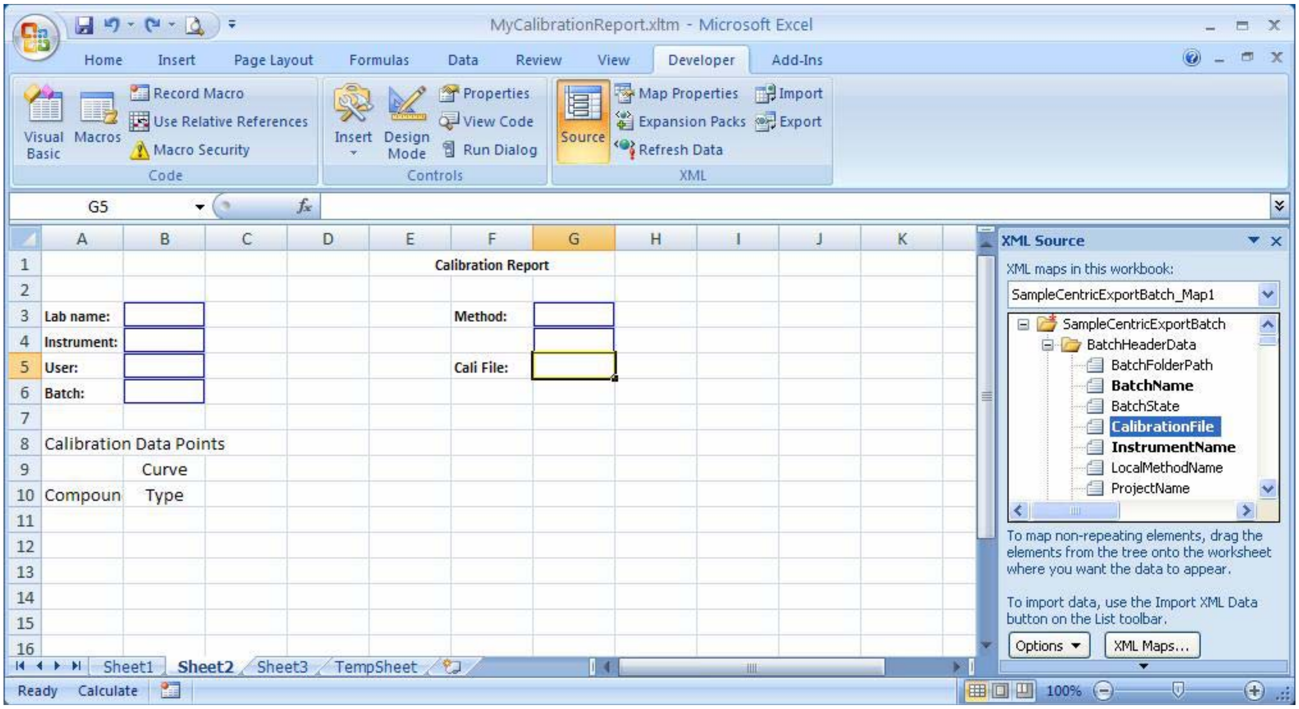
10. Click the **Font** tab.

11. In the Font Style box, select **Bold**.

12. Click **OK** in the Format Cells dialog box.

Ensure that your template looks like the following figure.

Figure 72. Template with static labels



	A	B	C	D	E	F	G	H	I	J	K
1						Calibration Report					
2											
3	Lab name:					Method:					
4	Instrument:										
5	User:					Cali File:					
6	Batch:										
7											
8	Calibration Data Points										
9		Curve									
10	Compoun	Type									

XML maps in this workbook:

SampleCentricExportBatch_Map1

- SampleCentricExportBatch
  - BatchHeaderData
    - BatchFolderPath
    - **BatchName**
    - BatchState
    - **CalibrationFile**
    - **InstrumentName**
    - LocalMethodName
    - ProjectName

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

To import data, use the Import XML Data button on the List toolbar.

❖ To create the cross-tab table in the template

1. Open the SampleCentricExportBatch/CalibrationData/CompoundCurveExportData folder, and do the following:

   a. Drag the **CompoundName** node from the XML Source tree to cell D1.

   b. Drag the **CurveType** node from the XML Source tree to cell E1.

   c. Drag the **ManualFlagSet** node from the XML Source tree to cell F1.

   d. Drag the **SampleLevel** node from the XML Source tree to cell G1.

2. Open the SampleCentricExportBatch/CalibrationData/CalibrationReplicates/SampleCompoundReplicateExportData folder, and do the following:

   a. Drag the **AcquisitionDate** node from the XML Source tree to cell H1.

b. Drag the **ExpectedRetentionTime** node from the XML Source tree to cell I1.

c. Drag the **ExclusionReason** node from the XML Source tree to cell J1.

d. Drag the **CurveDataPointValue** node from the XML Source tree to cell K1.

You are ready to add two helper columns: Curve Type and Point Value.

- The Curve Type column needs reformatting so that only the first character of the type is shown in the report:
  - L (Linear)
  - Q (Quadratic)
  - A (Average RF)
  - I (Internal Standard, which has no curve)
- The Point Value column contains the massaged data ready to be used in the pivot table.

❖ To add two helper columns to the template

1. Right-click the column A header cell and choose **Insert** from the shortcut menu.
2. Repeat Step 1 to add another new column.

   Starting from column C, the Excel application shifts all cells to the right.
3. Change the default header in column B to **Curve Type**.
4. Change the default header in column C to **Point Value**.
5. In cell B2, enter the following formula:

   ```
   =IF(A2="Internal Standard", "I", LEFT(E2, 1))
   ```
6. In cell C2, enter the following formula:

   ```
   =IF(OR(J2="eISTDNotFound",J2= "eISTDManuallyExcluded"), 99987654321,
   IF(J2="eManual", 88987654321, IF(F2, -K2, K2)))
   ```

You are now ready to add a function to process the pivot table.

❖ To create the ProcessSheet2Data add-in function

1. Click the **Developer** tab to display the Developer features.
2. Click the **Visual Basic** button.

   The Microsoft Visual Basic window opens.
3. Enter the following code into the Module1 code editor:

   ```
   Public Sub ProcessData()
       ProcessSheet2Data
   End Sub
   Private Sub ProcessSheet2Data()
   ```

```
        Dim rng As Range
        Set rng = mUtil.CreatePivotTable("DataTable", "TempSheet!O1",
"CompoundName;Curve Type", "AcquisitionDate;SampleLevel", "Point
Value")
End Sub
```

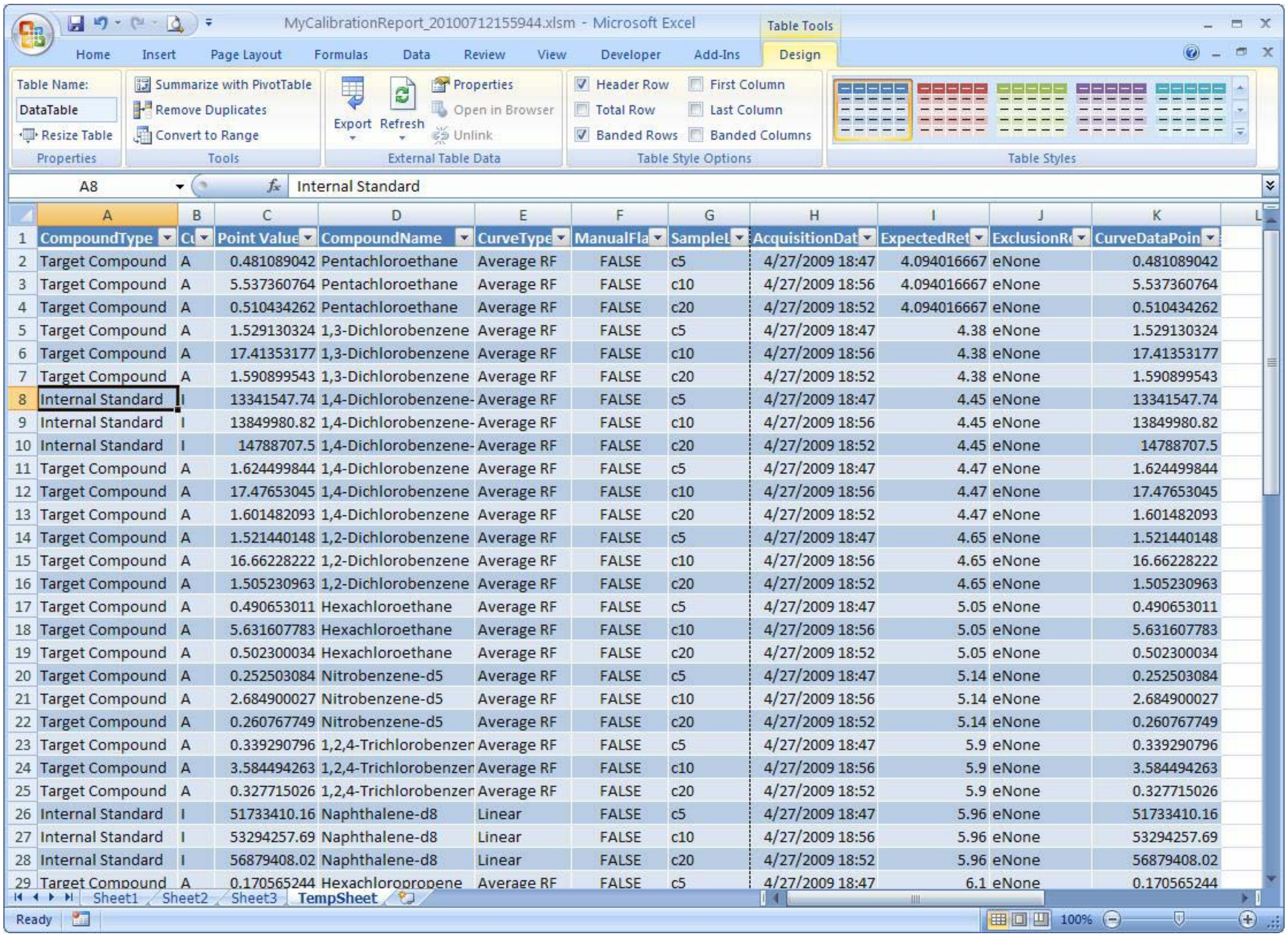
Where:

CreatePivotTable creates the pivot table based on the DataTable (a named range of your data on TempSheet) and puts the resulting pivot table in the cell starting from TempSheet!O1.

The pivot table uses the **CompoundName** and **Curve Type** column names as row headers, **AcquisitionDate** and **SampleLevel** as column headers, and the value from the **Point Value** column.

When you create your custom report, the original data table in the Excel spreadsheet is translated into a pivot table, as shown in the following examples.

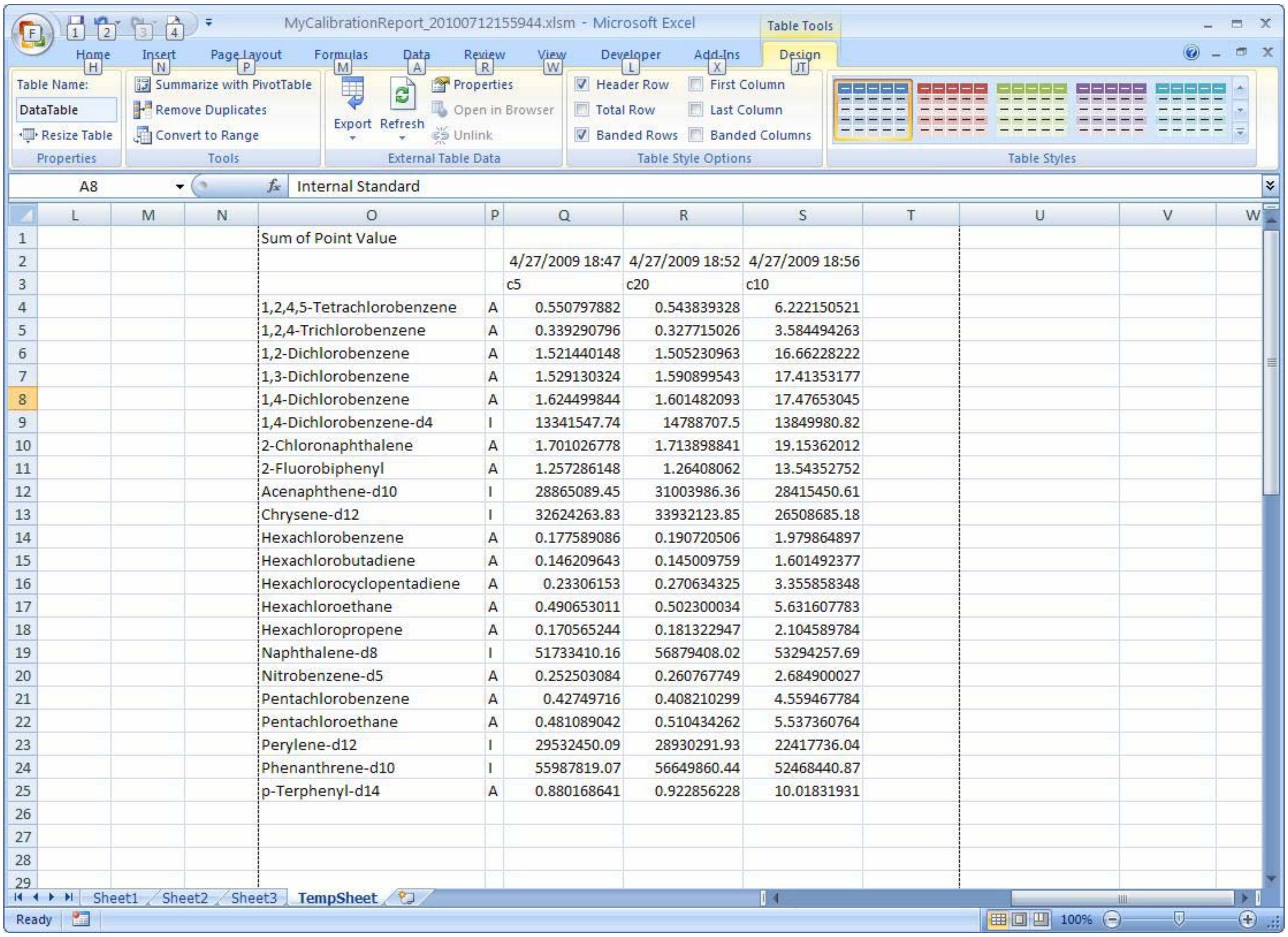
Figure 73. Original data table



A8 Internal Standard

	A	B	C	D	E	F	G	H	I	J	K
1	CompoundType	Cu	Point Value	CompoundName	CurveType	ManualFla	SampleL	AcquisitionDat	ExpectedRet	ExclusionR	CurveDataPoin
2	Target Compound	A	0.481089042	Pentachloroethane	Average RF	FALSE	c5	4/27/2009 18:47	4.094016667	eNone	0.481089042
3	Target Compound	A	5.537360764	Pentachloroethane	Average RF	FALSE	c10	4/27/2009 18:56	4.094016667	eNone	5.537360764
4	Target Compound	A	0.510434262	Pentachloroethane	Average RF	FALSE	c20	4/27/2009 18:52	4.094016667	eNone	0.510434262
5	Target Compound	A	1.529130324	1,3-Dichlorobenzene	Average RF	FALSE	c5	4/27/2009 18:47	4.38	eNone	1.529130324
6	Target Compound	A	17.41353177	1,3-Dichlorobenzene	Average RF	FALSE	c10	4/27/2009 18:56	4.38	eNone	17.41353177
7	Target Compound	A	1.590899543	1,3-Dichlorobenzene	Average RF	FALSE	c20	4/27/2009 18:52	4.38	eNone	1.590899543
8	Internal Standard	I	13341547.74	1,4-Dichlorobenzene-	Average RF	FALSE	c5	4/27/2009 18:47	4.45	eNone	13341547.74
9	Internal Standard	I	13849980.82	1,4-Dichlorobenzene-	Average RF	FALSE	c10	4/27/2009 18:56	4.45	eNone	13849980.82
10	Internal Standard	I	14788707.5	1,4-Dichlorobenzene-	Average RF	FALSE	c20	4/27/2009 18:52	4.45	eNone	14788707.5
11	Target Compound	A	1.624499844	1,4-Dichlorobenzene	Average RF	FALSE	c5	4/27/2009 18:47	4.47	eNone	1.624499844
12	Target Compound	A	17.47653045	1,4-Dichlorobenzene	Average RF	FALSE	c10	4/27/2009 18:56	4.47	eNone	17.47653045
13	Target Compound	A	1.601482093	1,4-Dichlorobenzene	Average RF	FALSE	c20	4/27/2009 18:52	4.47	eNone	1.601482093
14	Target Compound	A	1.521440148	1,2-Dichlorobenzene	Average RF	FALSE	c5	4/27/2009 18:47	4.65	eNone	1.521440148
15	Target Compound	A	16.66228222	1,2-Dichlorobenzene	Average RF	FALSE	c10	4/27/2009 18:56	4.65	eNone	16.66228222
16	Target Compound	A	1.505230963	1,2-Dichlorobenzene	Average RF	FALSE	c20	4/27/2009 18:52	4.65	eNone	1.505230963
17	Target Compound	A	0.490653011	Hexachloroethane	Average RF	FALSE	c5	4/27/2009 18:47	5.05	eNone	0.490653011
18	Target Compound	A	5.631607783	Hexachloroethane	Average RF	FALSE	c10	4/27/2009 18:56	5.05	eNone	5.631607783
19	Target Compound	A	0.502300034	Hexachloroethane	Average RF	FALSE	c20	4/27/2009 18:52	5.05	eNone	0.502300034
20	Target Compound	A	0.252503084	Nitrobenzene-d5	Average RF	FALSE	c5	4/27/2009 18:47	5.14	eNone	0.252503084
21	Target Compound	A	2.684900027	Nitrobenzene-d5	Average RF	FALSE	c10	4/27/2009 18:56	5.14	eNone	2.684900027
22	Target Compound	A	0.260767749	Nitrobenzene-d5	Average RF	FALSE	c20	4/27/2009 18:52	5.14	eNone	0.260767749
23	Target Compound	A	0.339290796	1,2,4-Trichlorobenzer	Average RF	FALSE	c5	4/27/2009 18:47	5.9	eNone	0.339290796
24	Target Compound	A	3.584494263	1,2,4-Trichlorobenzer	Average RF	FALSE	c10	4/27/2009 18:56	5.9	eNone	3.584494263
25	Target Compound	A	0.327715026	1,2,4-Trichlorobenzer	Average RF	FALSE	c20	4/27/2009 18:52	5.9	eNone	0.327715026
26	Internal Standard	I	51733410.16	Naphthalene-d8	Linear	FALSE	c5	4/27/2009 18:47	5.96	eNone	51733410.16
27	Internal Standard	I	53294257.69	Naphthalene-d8	Linear	FALSE	c10	4/27/2009 18:56	5.96	eNone	53294257.69
28	Internal Standard	I	56879408.02	Naphthalene-d8	Linear	FALSE	c20	4/27/2009 18:52	5.96	eNone	56879408.02
29	Target Compound	A	0.170565244	Hexachloropropene	Average RF	FALSE	c5	4/27/2009 18:47	6.1	eNone	0.170565244

Figure 74. Data table translated into a pivot table



A8 | Internal Standard

	O	P	Q	R	S
1	Sum of Point Value				
2			4/27/2009 18:47	4/27/2009 18:52	4/27/2009 18:56
3			c5	c20	c10
4	1,2,4,5-Tetrachlorobenzene	A	0.550797882	0.543839328	6.222150521
5	1,2,4-Trichlorobenzene	A	0.339290796	0.327715026	3.584494263
6	1,2-Dichlorobenzene	A	1.521440148	1.505230963	16.66228222
7	1,3-Dichlorobenzene	A	1.529130324	1.590899543	17.41353177
8	1,4-Dichlorobenzene	A	1.624499844	1.601482093	17.47653045
9	1,4-Dichlorobenzene-d4	I	13341547.74	14788707.5	13849980.82
10	2-Chloronaphthalene	A	1.701026778	1.713898841	19.15362012
11	2-Fluorobiphenyl	A	1.257286148	1.26408062	13.54352752
12	Acenaphthene-d10	I	28865089.45	31003986.36	28415450.61
13	Chrysene-d12	I	32624263.83	33932123.85	26508685.18
14	Hexachlorobenzene	A	0.177589086	0.190720506	1.979864897
15	Hexachlorobutadiene	A	0.146209643	0.145009759	1.601492377
16	Hexachlorocyclopentadiene	A	0.23306153	0.270634325	3.355858348
17	Hexachloroethane	A	0.490653011	0.502300034	5.631607783
18	Hexachloropropene	A	0.170565244	0.181322947	2.104589784
19	Naphthalene-d8	I	51733410.16	56879408.02	53294257.69
20	Nitrobenzene-d5	A	0.252503084	0.260767749	2.684900027
21	Pentachlorobenzene	A	0.42749716	0.408210299	4.559467784
22	Pentachloroethane	A	0.481089042	0.510434262	5.537360764
23	Perylene-d12	I	29532450.09	28930291.93	22417736.04
24	Phenanthrene-d10	I	55987819.07	56649860.44	52468440.87
25	p-Terphenyl-d14	A	0.880168641	0.922856228	10.01831931

Note The AcquisitionDate in row 2 is for sorting purposes. It is not required on the report.

The pivot table on TempSheet contains the data you want to include on the second page of your Calibration report.

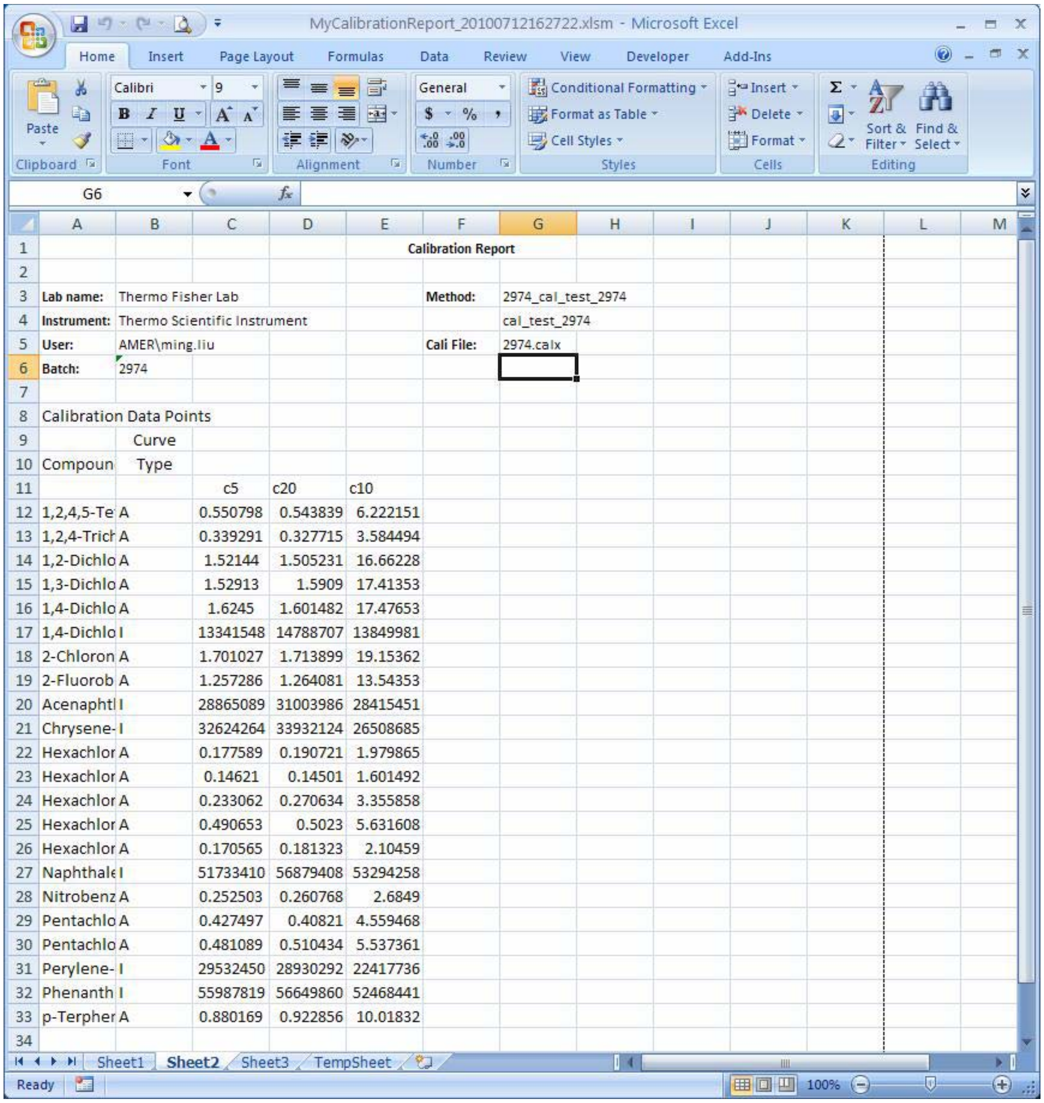
❖ To copy the pivot table to Sheet2

Use the CopyPaste add-in function.

```
Private Sub ProcessSheet2Data()

    Dim rng As Range

    Set rng = mUtil.CreatePivotTable("DataTable", "TempSheet!O1",
"CompoundName;Curve Type", "AcquisitionDate;SampleLevel", "Point
Value")

mUtil.CopyPaste mUtil.ShiftRange(rng, 0, 2), "Sheet2!A11", False, True

End Sub
```

When you generate your report, Sheet2 looks like the following figure:

Figure 75. Sheet2 in a generated report



	A	B	C	D	E	F	G
1						**Calibration Report**	
2							
3	**Lab name:**	Thermo Fisher Lab				**Method:**	2974_cal_test_2974
4	**Instrument:**	Thermo Scientific Instrument					cal_test_2974
5	**User:**	AMER\ming.liu				**Cali File:**	2974.calx
6	**Batch:**	2974					
7							
8	Calibration Data Points						
9		Curve					
10	Compoun	Type					
11			c5	c20	c10		
12	1,2,4,5-Te	A	0.550798	0.543839	6.222151		
13	1,2,4-Trich	A	0.339291	0.327715	3.584494		
14	1,2-Dichlo	A	1.52144	1.505231	16.66228		
15	1,3-Dichlo	A	1.52913	1.5909	17.41353		
16	1,4-Dichlo	A	1.6245	1.601482	17.47653		
17	1,4-Dichlo	I	13341548	14788707	13849981		
18	2-Chloron	A	1.701027	1.713899	19.15362		
19	2-Fluorob	A	1.257286	1.264081	13.54353		
20	Acenaphtl	I	28865089	31003986	28415451		
21	Chrysene-	I	32624264	33932124	26508685		
22	Hexachlor	A	0.177589	0.190721	1.979865		
23	Hexachlor	A	0.14621	0.14501	1.601492		
24	Hexachlor	A	0.233062	0.270634	3.355858		
25	Hexachlor	A	0.490653	0.5023	5.631608		
26	Hexachlor	A	0.170565	0.181323	2.10459		
27	Naphthale	I	51733410	56879408	53294258		
28	Nitrobenz	A	0.252503	0.260768	2.6849		
29	Pentachlo	A	0.427497	0.40821	4.559468		
30	Pentachlo	A	0.481089	0.510434	5.537361		
31	Perylene-	I	29532450	28930292	22417736		
32	Phenanth	I	55987819	56649860	52468441		
33	p-Terpher	A	0.880169	0.922856	10.01832		
34							

❖ To complete the layout

(Optional) Resize the columns to make them look best on the page.

Creating a Calibration File List – Sheet3

The last sheet of the Calibration Report contains a list of all calibration sample types. The information you need for this list is embedded in each compound node; therefore, you must filter out the unwanted data. You will use the Excel filtering function to create this filtered list by adding a CompoundID column as a derived helper column. This sample list must also include a derived acquisition date column to order the samples by their acquisition date.

❖ To enter static labels for the report heading

1. Type the report header **Calibration Report** in cell B1.
2. Type **Lab name:** in cell B3.
3. Type **Instrument:** in cell B4.
4. Type **User:** in cell B5.
5. Type **Batch:** in cell B6.
6. Type **Method:** in cell F3.
7. Type **Cali File:** in cell F5.
8. Hold down the CTRL key and select the label cells you just created.
9. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See "Format Cells dialog box - Font page" on page 96.
10. Click the **Font** tab.
11. In the Font Style box, select **Bold**.
12. Click **OK** in the Format Cells dialog box.

You are now ready to add another XML map to use for Sheet3.

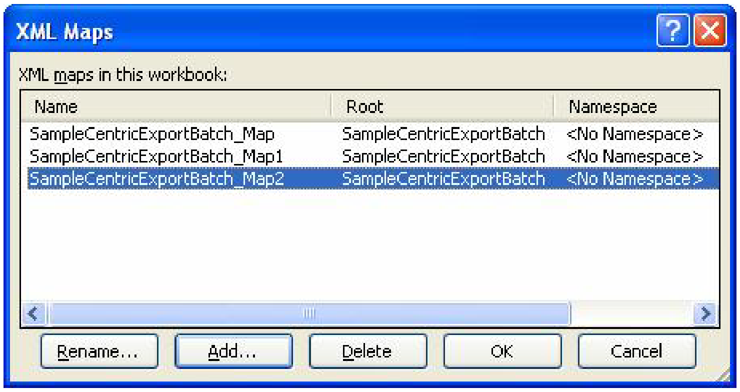
❖ To add an XML mapping file to the template

1. In the XML Source pane in your spreadsheet, click **XML Maps**, XML Maps... .

   The XML Maps dialog box opens.
2. Click **Add**.
3. Navigate to your XML mapping file (the same one you used before) and click **Open**.

   The Excel application adds the mapping file to the XML Maps in This Workbook list. Because this is the second duplicate of your original mapping file, the number **2** is appended to the file name.
4. Click **OK**.

Figure 76. XML Maps dialog box with three mapping files



Name	Root	Namespace
SampleCentricExportBatch_Map	SampleCentricExportBatch	<No Namespace>
SampleCentricExportBatch_Map1	SampleCentricExportBatch	<No Namespace>
SampleCentricExportBatch_Map2	SampleCentricExportBatch	<No Namespace>

Because you have only a single .xml file to import, the add-in component populates all maps when you import the .xml file.

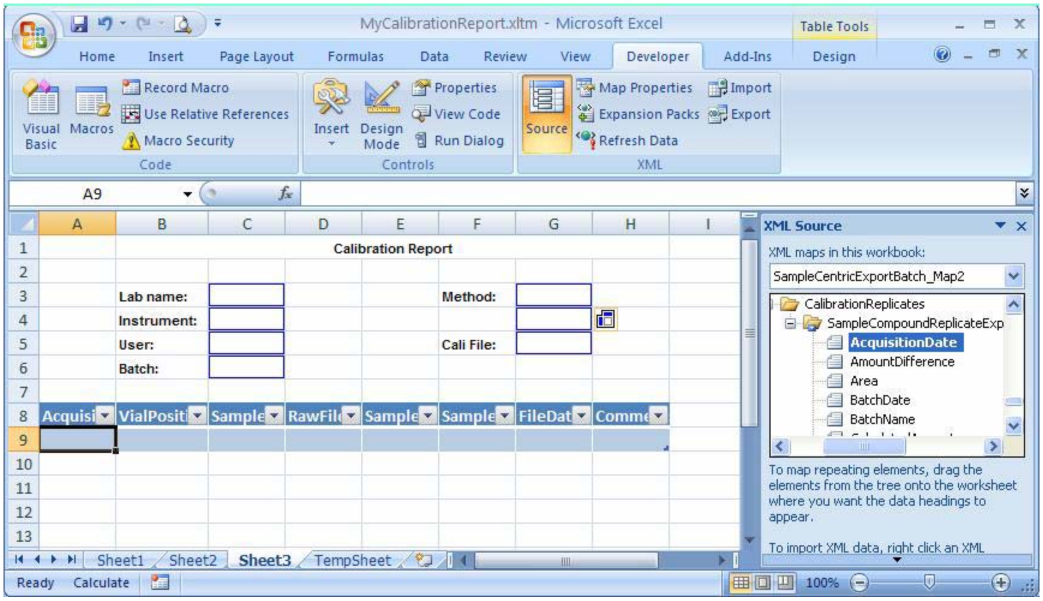
You are now ready to add mapped nodes from the SampleCentricExportBatch_Map2 file.

❖ To add mapped fields to the template

1. Open the SampleCentricExportBatch/CalibrationReplicates/SampleCompound ReplicateExportData folder.
2. Drag the **AcquisitionDate** node from the XML Source tree to cell A8.
3. Drag the **VialPosition** node from the XML Source tree to cell B8.
4. Drag the **SampleId** node from the XML Source tree to cell C8.
5. Drag the **RawFileName** node from the XML Source tree to cell D8.
6. Drag the **SampleLevel** node from the XML Source tree to cell E8.
7. Drag the **SampleName** node from the XML Source tree to cell F8.
8. Drag the **FileDate** node from the XML Source tree to cell G8.
9. Drag the **Comments** node from the XML Source tree to cell H8.

Ensure that your template looks like the following figure.

Figure 77. Template with headers and mapped fields



Calibration Report

Lab name:

Instrument:

User:

Batch:

Method:

Cali File:

A	B	C	D	E	F	G	H
Acquisi	VialPosit	Sample	RawFil	Sample	Sample	FileDat	Comme

XML Source

XML maps in this workbook:

SampleCentricExportBatch_Map2

- CalibrationReplicates
  - SampleCompoundReplicateExp
    - AcquisitionDate
    - AmountDifference
    - Area
    - BatchDate
    - BatchName

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML

Completing the Layout

In the following procedures, you will rename the default table header names, name the mapped table, and finish the layout.

❖ To rename mapped table header names

1. Change B8 from VialPosition to **Pos**.
2. Change C8 from SampleId to **Sample ID**.
3. Change D8 from RawFileName to **File Name**.
4. Change E8 from SampleLevel to **Level**.
5. Change F8 from SampleName to **Sample Name**.
6. Change G8 from FileDate to **File Date**.
7. Change H8 from Comments to **Comment**.
8. Select cells **B8** through **H8**.
9. Right-click the Vial Pos cell and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See “Format Cells dialog box - Font page” on page 96.
10. Click the **Font** tab.
11. In the Font Style box, select **Bold**.

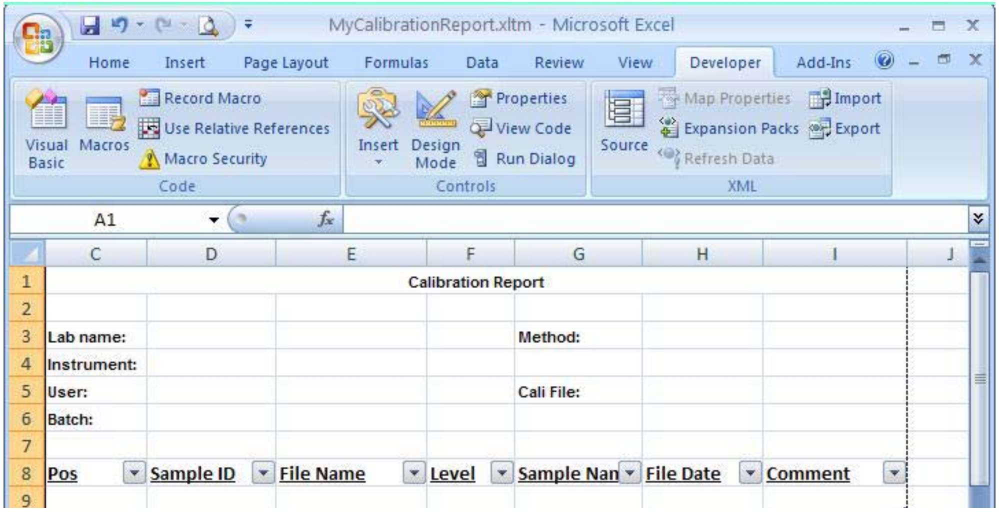
12. In the Underline list, select **Single**.

13. Click **OK**.

14. Resize the columns to have the appropriate width and still fit into a single printed page.

Ensure that your template looks like the following figure.

Figure 78. Template with custom header text



	C	D	E	F	G	H	I	J
1	Calibration Report							
2								
3	Lab name:				Method:			
4	Instrument:							
5	User:				Cali File:			
6	Batch:							
7								
8	Pos	Sample ID	File Name	Level	Sample Nan	File Date	Comment	
9								

You are now ready to use the Excel named range feature to name the table in the template. You can give a name to a range of cells and then use the name to refer to this area.

❖ To name the table in the template

1. Click the **Formulas** tab and then click **Name Manager**.

   Note The Excel application automatically creates a name for all tables, such as Table1 or Table2. Each time you create a new table, it increments the table name, even if you deleted a previous one.

2. In the Name Manager dialog box, double-click **Table1**.

   The table name might vary. Use the sheet, row, and column numbers in the Refers To column to verify the table.

   The Edit Name dialog box opens.

3. Change the default name to **DataTableFileList**.

4. Click **OK** in the Edit Name dialog box.

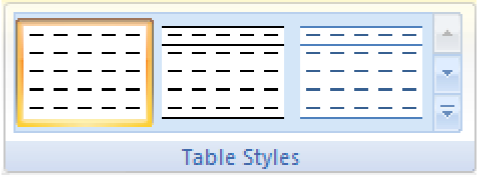
5. Click **Close** in the Name Manager dialog box.

You are now ready to polish the layout of the template.

❖ To change the background in the template

1. Click the header of the data table to display the Design tab.
2. Click the **Design** tab.

   The Design tab displays the Table Styles functions.

3. Expand the Table Styles gallery and click **Clear**, Clear , in the lower left corner of the gallery.

   The background changes to a plain background.

4. Right-click the column A header (AcquisitionDate) and choose **Hide** from the shortcut menu.

   Note You can hide any column in an Excel spreadsheet. However, if you need derived columns for the TraceFinder custom reports, always use the first columns. A derived column in the middle of the table can cause trouble when copying blocks of data for the repeating area.

Adding VBA Code to the Template

You are now ready to use VBA code to add a filtering function to the template.

❖ To add the Acquisition Date filtering function to the template

1. Click the **Developer** tab to display the Developer features.
2. Click the **Visual Basic** button.

   The Microsoft Visual Basic window opens.

3. Right-click the VBA project and choose **Insert** > **Module** from the shortcut menu.

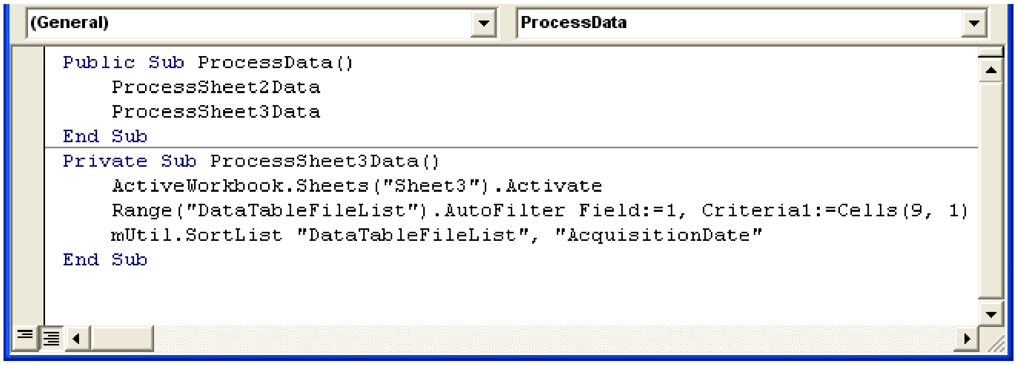
   The application adds a module to the Modules folder in the project and opens a code editor.

4. Enter following code into the module code editor:

```
Public Sub ProcessData()
    ProcessSheet2Data
    ProcessSheet3Data
End Sub
Private Sub ProcessSheet3Data()
    ActiveWorkbook.Sheets("Sheet3").Activate
    Range("DataTableFileList").AutoFilter Field:=1,
Criteria1:=Cells(9, 1)
    mUtil.SortList "DataTableFileList", "AcquisitionDate"
End Sub
```

Your Visual Basic code should look like this:

Figure 79. VBA data



```
Public Sub ProcessData()
    ProcessSheet2Data
    ProcessSheet3Data
End Sub
Private Sub ProcessSheet3Data()
    ActiveWorkbook.Sheets("Sheet3").Activate
    Range("DataTableFileList").AutoFilter Field:=1, Criteria1:=Cells(9, 1)
    mUtil.SortList "DataTableFileList", "AcquisitionDate"
End Sub
```

With this code and a *RawFileName*_SampleCentricData.xml file, the template is ready to run.

Testing the Template

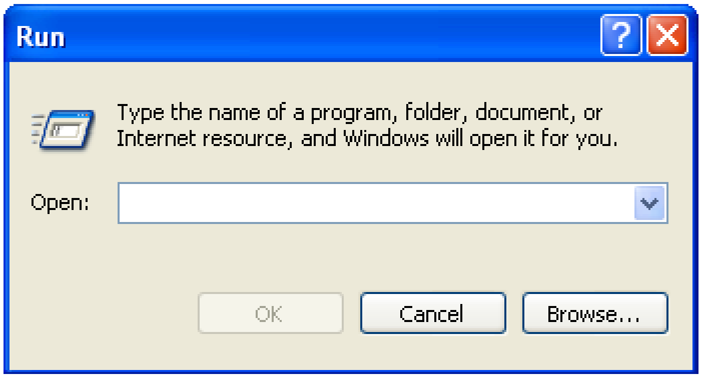
You are now ready to run a test from the Windows **Start > Run** box. In this test, the code you enter does the following:

- Opens the Excel application.
- Creates a new workbook using the MyCalibrationReport.xltm template.
- Passes the parameters to the template.
- Generates a report based on the level5_SampleCentricData.xml source file.
- Saves the report to the MyCalibrationReport.xlsm file.

❖ To run the test

1. Choose **Start > Run**.

   The Run dialog box opens.



Run
?
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
OK
Cancel
Browse...

Tip By default, the TraceFinder application file structure results in a long file path. The Windows Run dialog box limits you to 256 characters. To work around this limitation, either copy files to a folder close to the root directory or put the command line into a batch file.

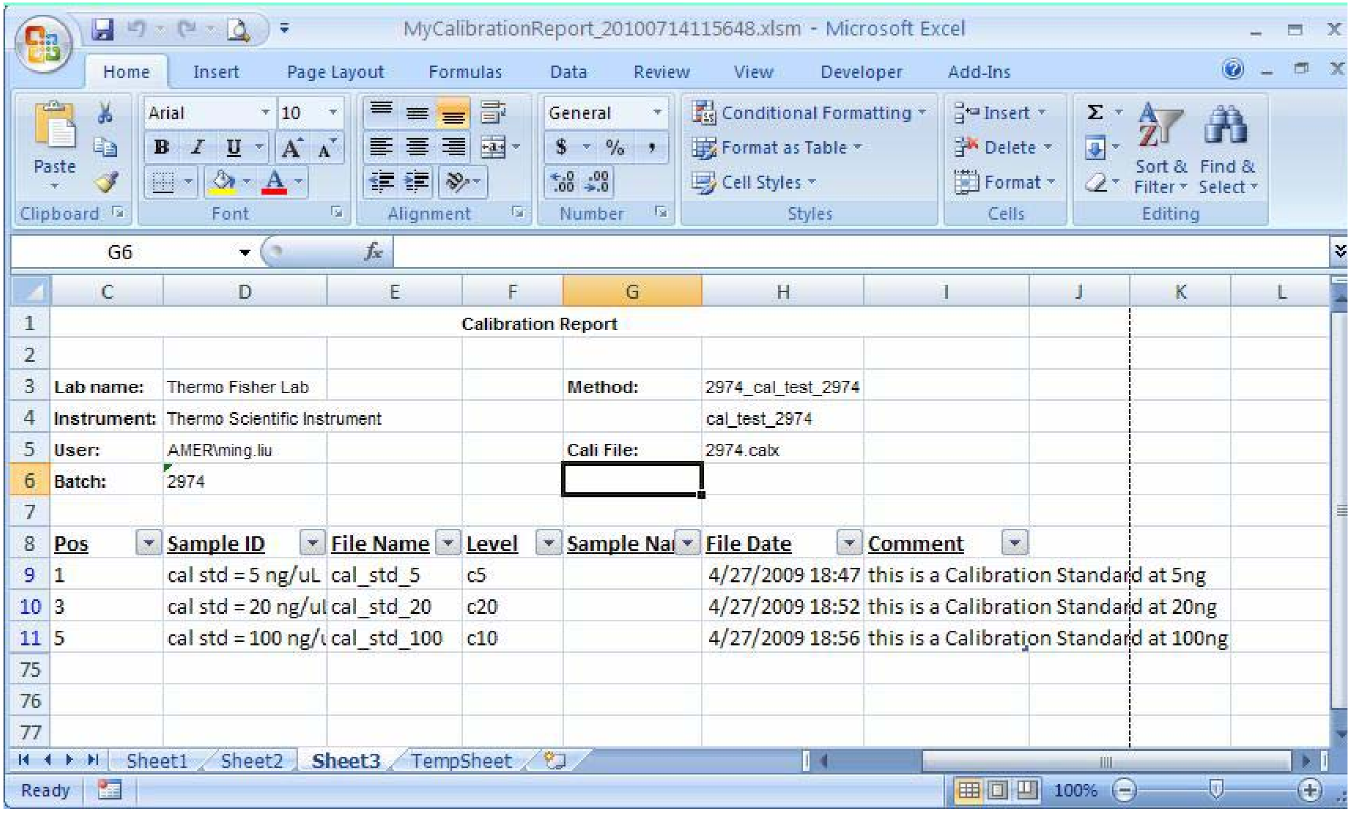
2. Enter this code in the Open box:

```
Excel.exe /n "C:\Thermo\Custom Report
Tutorials\MyCalibrationReport.xltm" /x/C:\Thermo\
Custom|Report|Tutorials\Data\level5_SampleCentricData.xml/C:\Thermo\
Custom|Report|Tutorials\Reports\MyCalibrationReport.xlsm/s/h
```

   For detailed descriptions of the parameters used in this code, see "Testing the Template" on page 25.

You should see the generated report on your screen similar to the following figure (actual data will vary depending on the batch you used):

Figure 80. Generated report



	C	D	E	F	G	H	I
1				Calibration Report			
2							
3	Lab name:	Thermo Fisher Lab			Method:	2974_cal_test_2974	
4	Instrument:	Thermo Scientific Instrument				cal_test_2974	
5	User:	AMER\ming.liu			Cali File:	2974.calx	
6	Batch:	2974					
7							
8	Pos	Sample ID	File Name	Level	Sample Na	File Date	Comment
9	1	cal std = 5 ng/uL	cal_std_5	c5		4/27/2009 18:47	this is a Calibration Standard at 5ng
10	3	cal std = 20 ng/ul	cal_std_20	c20		4/27/2009 18:52	this is a Calibration Standard at 20ng
11	5	cal std = 100 ng/u	cal_std_100	c10		4/27/2009 18:56	this is a Calibration Standard at 100ng
75							
76							
77							

You are now ready to integrate the template into the TraceFinder application.

❖ To use the template in the TraceFinder application

1. Copy your new template into the dedicated folder of the application:

   C:\Thermo\Shared\Templates\Reports

2. Use this template the same way as the other delivered reports in the TraceFinder application.

   For detailed information about how to use custom reports, refer to the *TraceFinder User Guide*.

Creating a DCC Report

In this chapter, you will create an Excel template for a doping control center report similar to the Steroid Analysis Report delivered with the TraceFinder application.

Contents

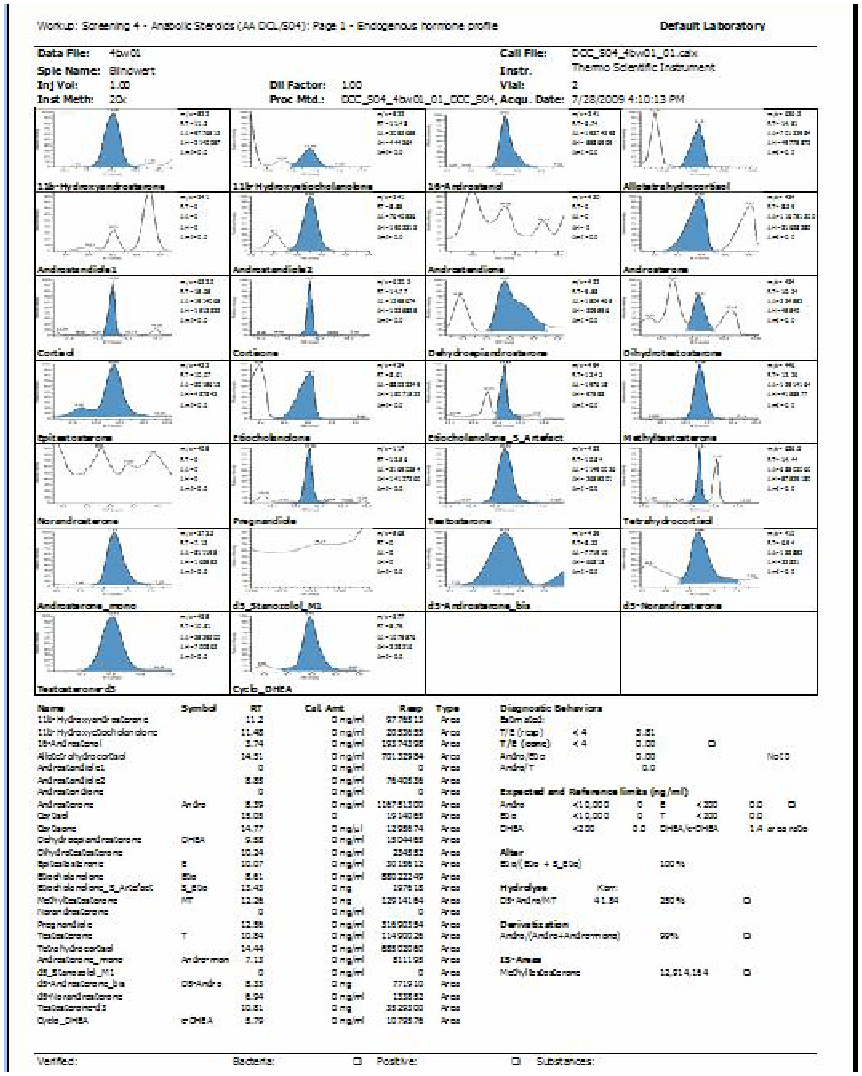
A DCC report is a special custom report that ties to a specific method. This is a common report used in doping control centers.

A DCC report is different from other reports including not using standard headers and footers. To create this report template, you will use our add-in framework and the Excel formula capability to adapt the formulas for business needs.

Features of a DCC report include the following:

- A highly condensed, single page, sample-level summary report
- A specific method with about 26 compounds (up to 28 in our case)
- The chromatograms showing on the top of the report; the compounds and related data a showing at the bottom
- The compounds listed in a specific order (In the method, compound names are prefixed with Z or ZZ to adapt to the special sorting needs. You will use an extra mapping table to remove these prefixes on the actual report.)
- Compound-related data and calculations based on this data displayed at the bottom of the report

Figure 81. DCC Report example



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile | Default Laboratory

Data File:	4bw01			**Cal File:**	DCC_S04_4bw01_01.calx
Sple Name:	Blindwert			**Instr.**	Thermo Scientific Instrument
Inj Vol:	1.00	**Dil Factor:**	1.00	**Vial:**	2
Inst Meth:	20x	**Proc Mtd.:**	DCC_S04_4bw01_01_DCC_S04	**Acqu. Date:**	7/28/2009 4:10:13 PM

Name	Symbol	RT	Cal. Amt	Resp	Type
11b-Hydroxyandrosterone		11.2	0 ng/ml	97 785 13	Area
11b-Hydroxyetiocholanolone		11.48	0 ng/ml	20 538 55	Area
16-Androstenol		3.74	0 ng/ml	193 743 98	Area
Allotetrahydrocortisol		14.51	0 ng/ml	701 329 84	Area
Androstandiole1		0	0 ng/ml	0	Area
Androstandiole2		8.85	0 ng/ml	78 405 38	Area
Androstendione		0	0 ng/ml	0	Area
Androsterone	Andro	8.39	0 ng/ml	1187 513 00	Area
Cortisol		15.05	0	19 140 85	Area
Cortisone		14.77	0 ng/µl	12 958 74	Area
Dehydroepiandrosterone	DHEA	9.58	0 ng/ml	15 044 85	Area
Dihydrotestosterone		10.24	0 ng/ml	2548 52	Area
Epitestosterone	E	10.07	0 ng/ml	30 158 12	Area
Etiocholanolone	Etio	8.81	0 ng/ml	880 222 49	Area
Etiocholanolone_5_Artefact	5_Etio	13.43	0 ng	1976 18	Area
Methyltestosterone	MT	12.28	0 ng	129 141 84	Area
Norandrosterone		0	0 ng/ml	0	Area
Pregnandiole		12.58	0 ng/ml	318 903 94	Area
Testosterone	T	10.84	0 ng/ml	114 900 28	Area
Tetrahydrocortisol		14.44	0 ng/ml	885 020 80	Area
Androsterone_mono	Andro mon	7.13	0 ng/ml	8111.98	Area
d3_Stanozolol_M1		0	0 ng/ml	0	Area
d5-Androsterone_bis	D5-Andro	8.33	0 ng	7719 10	Area
d3-Norandrosterone		8.94	0 ng/ml	1558 52	Area
Testosterone d3		10.81	0 ng	33 293 00	Area
Cyclo_DHEA	c-DHEA	5.79	0 ng/ml	10 793 78	Area

Diagnostic Behaviors
Estimated:

T/E (resp)	< 4	3.81				
T/E (conc)	< 4	0.00		□		
Andro/Etio		0.00				NoCO
Andro/T		0.0				

Expected and Reference limits (ng/ml)

Andro	<10,000	0	E	< 200	0.0	□
Etio	<10,000	0	T	< 200	0.0	
DHEA	<200	0.0	DHEA/c-DHEA		1.4 area ratio	

Alter
Etio/(Etio + 5_Etio) 100 %

Hydrolyse Korr:
D5-Andro/MT 41.84 250 % □

Derivatization
Andro/(Andro+Andro mono) 99% □

IS-Areas
Methyltestosterone 12,914,184 □

Verified: Bacteria: □ Positive: □ Substances:

As you create a DCC Report template, you will learn the following:

- How to create headers, footers, and other static objects.
- How to create specific headers and footers not using the add-in framework
- How to create a compound mapping table that maps the compound names in a method to displayed text.
- How to use reference symbols in formulas
- How to use a TempSheet to hold a data table for compound-related information
- How to sort the data table by compound name after you import the data
- How to create a template that loops through each compound, finds its display name through the compound mapping table, copies it to a display page, and creates the image area for the compound

Before you begin creating a template for any report, identify the data points on the report and understand how they map to the nodes in the XML Source.

Opening the Report Template

In these step-by-step procedures, you will use the ThermoCustomReportBaseTemplate.xltm file that you created in the Getting Started section as the basic template.

❖ To begin your DCC report template

In Windows, make a copy of the ThermoCustomReportBaseTemplate.xltm file and name the new copy **MyDCCReport.xltm**.

❖ To load the template file

1. Open the Excel application.
2. Click the **Excel application** icon, , and choose **Open**.
3. In the Open dialog box, double-click **MyDCCReport.xltm**.

Tip Do not open this template file from Windows Explorer, because instead of opening the template for editing, you create a new spreadsheet based on this template.

Creating the Report Header

This report has many sections with different layouts on the same page, so having many smaller columns makes the adjustment easier. You can use the Excel application to run text from one cell over to the next cell as long as the cell is empty.

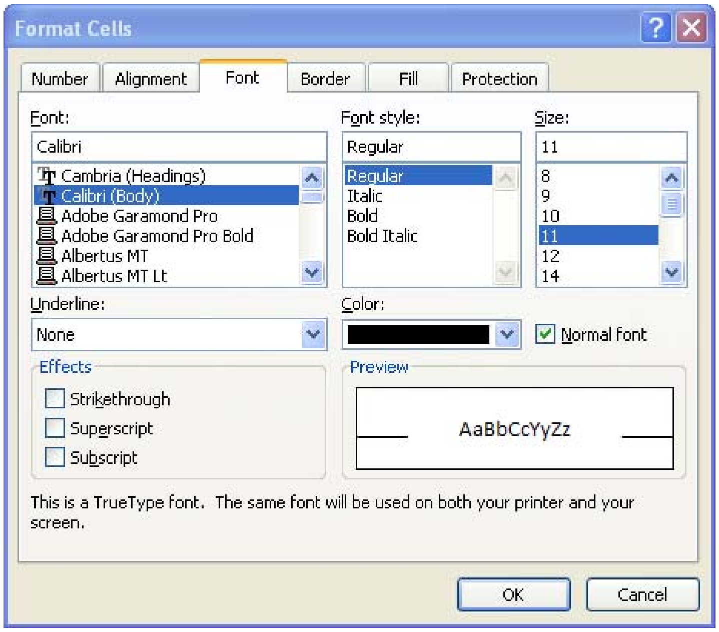
In this section, you will create static labels for the report header and then drag the data nodes from the XML Source tree.

❖ To enter static labels for the report heading

1. Type **Data File:** in cell A3.
2. Type **Sple Name:** in cell A4.
3. Type **Inj Vol:** in cell A5.
4. Type **Inst Meth:** in cell A6.
5. Type **Dil Factor:** in cell G5.
6. Type **Proc Mtd.:** in cell G6.
7. Type **Cali File:** in cell M3.
8. Type **Instr.:** in cell M4.
9. Type **Vial:** in cell M5.
10. Type **Acqu. Date:** in cell M6.
11. Hold down the CTRL key and select the label cells you just created.
12. Right-click and choose **Format Cells** from the shortcut menu.

    The Format Cells dialog box opens.

Figure 82. Format Cells dialog box - Font page



Font: Calibri

Font style: Regular

Size: 11

Underline: None

Color:

☑ Normal font

☐ Strikethrough

☐ Superscript

☐ Subscript

This is a TrueType font. The same font will be used on both your printer and your screen.

13. Click the **Font** tab.

14. In the Font Style box, select **Bold**.

15. Click **OK** in the Format Cells dialog box.

You are now ready to add mapped nodes from the SampleCentricExportBatch folder.

Note A “node” in the XML Source tree creates a “field” in the spreadsheet.

❖ To add mapped fields to the template

1. Click the **Developer** tab to display the Developer features.

   Tip For instructions about how to enable the Developer tab, which is hidden by default, see Chapter 3, “Getting Started.”

2. Click the **Source** button, Source .

   The spreadsheet displays the XML Source pane. You can drag mapped nodes from the XML Source pane to the spreadsheet.

3. Open the SampleCentricExportBatch folder and do the following:

   a. Open the MethodHeaderData folder and drag the **LabName** node from the XML Source tree to cell R1.

   b. Open the Samples/SampleCentricExportSample folder and drag the **RawFileName** node from the XML Source tree to cell C3.

c. Open the Samples/SampleCentricExportSample folder and drag the **SampleName** node from the XML Source tree to cell C4.

d. Open the Samples/SampleCentricExportSample folder and drag the **InjectionVolume** node from the XML Source tree to cell C5.

e. Open the MethodHeaderData folder and drag the **InstrumentMethodName** node from the XML Source tree to cell C6.

f. Open the Samples/SampleCentricExportSample folder and drag the **ConversionFactor** node from the XML Source tree to cell I5.

g. Open the MethodHeaderData folder and drag the **MethodName** node from the XML Source tree to cell I6.

h. Open the MethodHeaderData folder and drag the **CalibrationFile** node from the XML Source tree to cell O3.

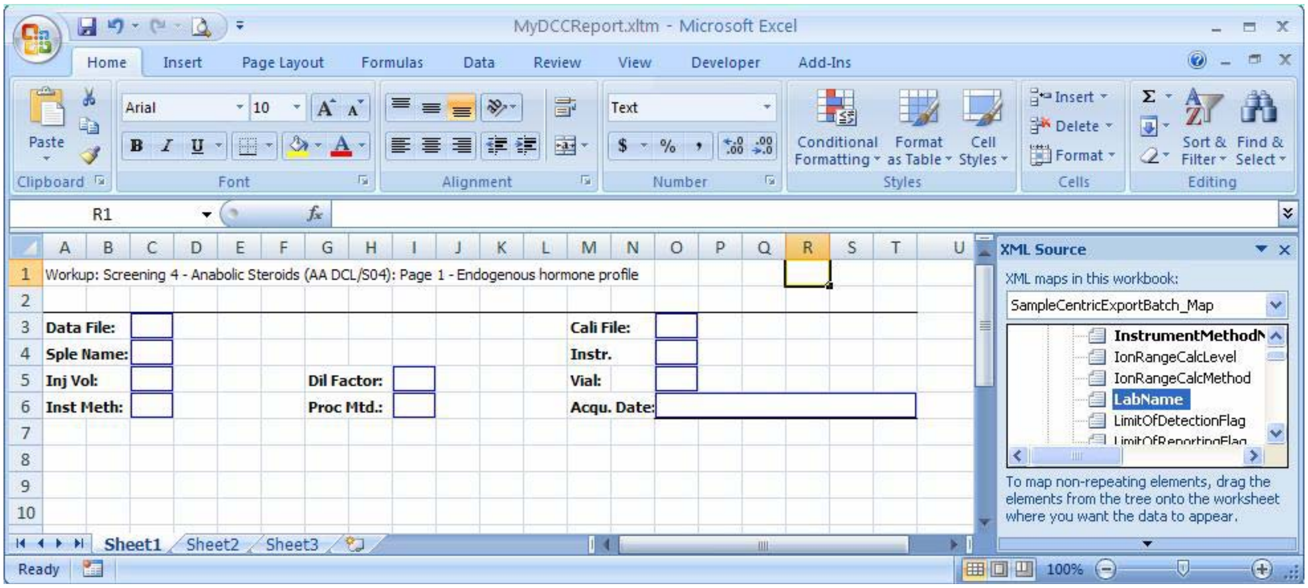
i. Open the MethodHeaderData folder and drag the **InstrumentName** node from the XML Source tree to cell O4.

j. Open the Samples/SampleCentricExportSample folder and drag the **VialPosition** node from the XML Source tree to cell O5.

k. Open the Samples/SampleCentricExportSample folder and drag the **AcquisitionDate** node from the XML Source tree to cell O6.

Ensure that your template looks like the following figure.

Figure 83. Template with header cells



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile

Data File:		Cali File:	
Sple Name:		Instr.	
Inj Vol:	Dil Factor:	Vial:	
Inst Meth:	Proc Mtd.:	Acqu. Date:	

XML maps in this workbook:

SampleCentricExportBatch_Map

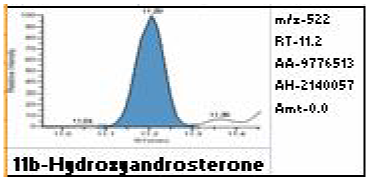
- InstrumentMethodN
- IonRangeCalcLevel
- IonRangeCalcMethod
- LabName
- LimitOfDetectionFlag
- LimitOfReportingFlag

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

Creating an Image Area

For each compound, you must show a quantitation peak chromatogram, the compound name below the chromatogram, and the legend on the right, as shown in the following figure.

Figure 84. Compound data as shown in a DCC report



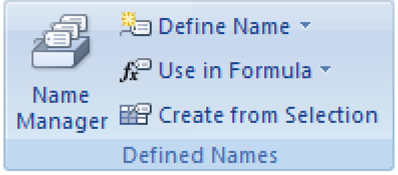
To create this image area, you will define three named ranges:

- QuanChart at cells A7 through D11 for holding the image
- CompoundName at cell A12 for holding the compound name
- ChartLegend at cells E7 through E12 for holding the legend

To create this DCC report template, you will duplicate this area 28 times. Use the ShiftRange function to dynamically calculate the ranges for each compound at run time and use VBA code to merge the ChartLegend cells after their value is populated.

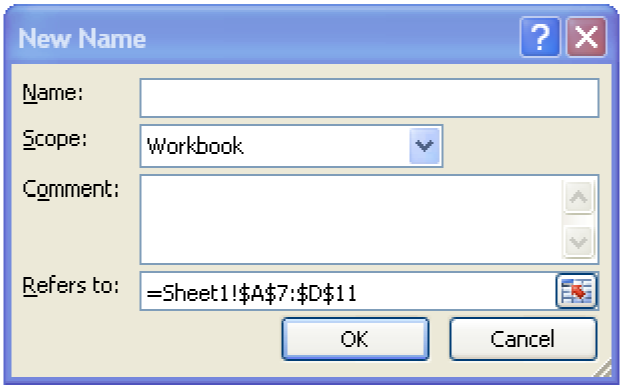
❖ To create the QuanChart named range in the template

1. Select cells **A7** through **D11**.
2. Click the **Formulas** tab, and then click **Define Name**.



The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.

Figure 85. New Name dialog box


New Name

Name:

Scope: Workbook

Comment:

Refers to: =Sheet1!$A$7:$D$11

3. In the Name box, type **QuanChart**.

4. Click **OK**.

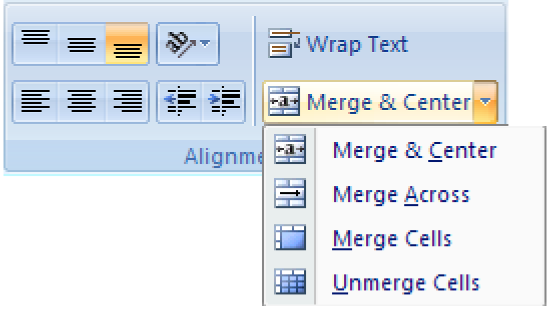
You are now ready to merge the cells.

❖ To merge the cells in the template

1. Select cells **A7** through **D11**.

2. Click the **Home** tab to display the Home functions.

3. Click the **Merge & Center** down arrow in the Alignment area.



4. Choose **Merge Cells** from the menu.

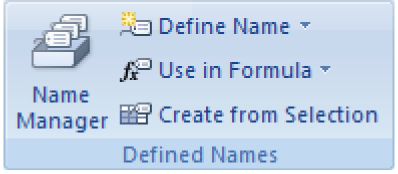
You are now ready to create a named range for the compound.

❖ To create the CompoundName named range in the template

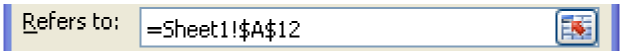
1. Select cell **A12**.

2. Click the **Formulas** tab to display the Formulas functions.

3. Click **Define Name** in the Defined Names area.



The selected cell automatically appears in the **Refers To** box of the New Name dialog box.


Refers to: =Sheet1!$A$12

4. In the Name box, type **CompoundName**.

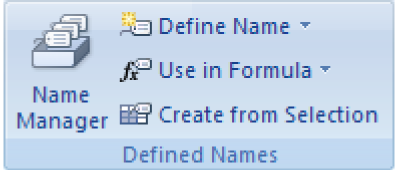
5. Click **OK**.

You are now ready to create a named range for the legend and align the text from the bottom to the top of the range.

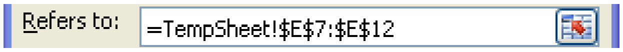
❖ To create the ChartLegend named range in the template

1. Select cells **E7** through **E12**.

2. Click the **Formulas** tab, and then click **Define Name**.



The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.



Refers to: =TempSheet!$E$7:$E$12

3. In the Name box, type **ChartLegend**.

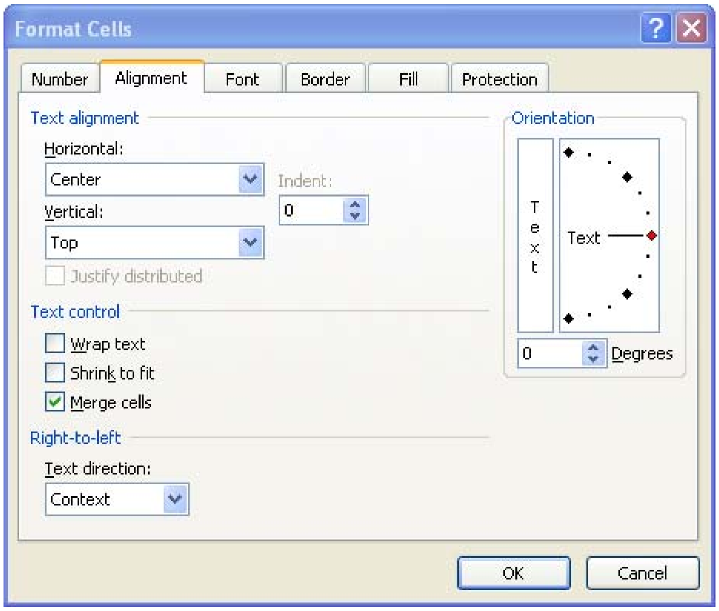
4. Click **OK**.

5. Select the cells in the ChartLegend cell range: cells **A7** through **D11**.

6. Right-click and choose **Format Cells** from the shortcut menu.

The Format Cells dialog box opens.

Figure 86. Format Cells dialog box - Alignment page



Format Cells

Number | Alignment | Font | Border | Fill | Protection

Text alignment

Horizontal: Center

Indent: 0

Vertical: Top

□ Justify distributed

Text control

□ Wrap text

□ Shrink to fit

☑ Merge cells

Right-to-left

Text direction: Context

Orientation

0 Degrees

7. Click the **Alignment** tab.

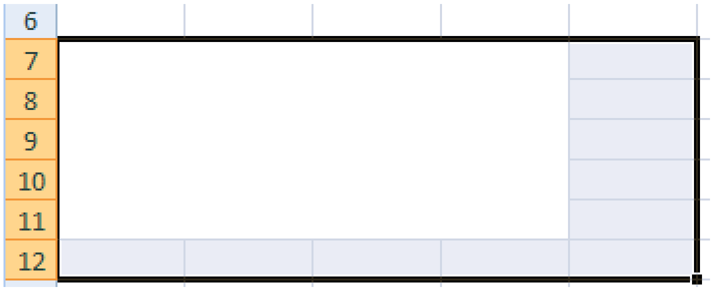
8. In the Vertical list, select **Top**.

9. Click **OK**.

❖ To add a border to the cell range

1. Select cells **A7** through **E12**.

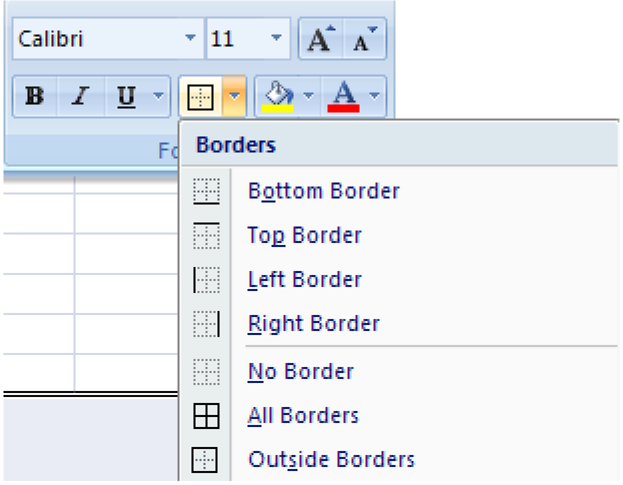
   The selected cells look like this:



   These selected cells include the area for the image, the column on the right for the legend, and the row at the bottom for the compound name.

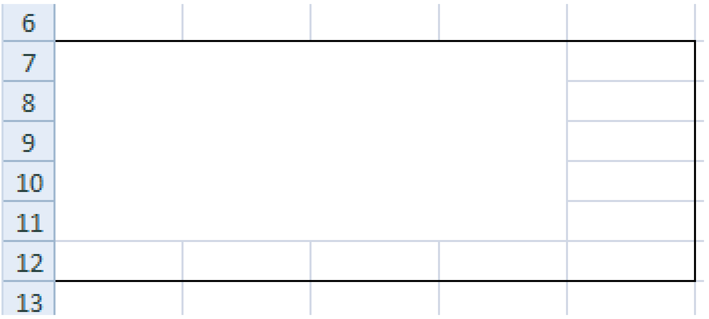
2. Click the **Home** tab to display the Home functions.

3. Click the down arrow next to the Borders icon in the Font area.



4. Choose **Outside Borders** from the menu.

   The bordered cells look like this:



In the next procedures, you will make 28 copies of the bordered area.

❖ To horizontally copy the QuanChart cell range

1. Select the cells **A7** through **E12**.

   You cannot simply click the bordered area; you must drag your cursor across all the cells to select them.

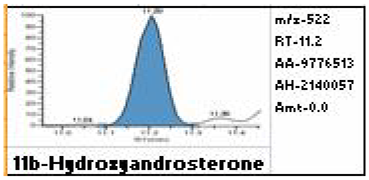
2. Right-click and choose **Copy** from the shortcut menu.
3. Right-click cell F7 and choose **Paste** from the shortcut menu.
4. Repeat Step 3 for cells K7 and P7.

The copy/paste function copies the cells and the border, but it does not copy the named ranges.

❖ To vertically copy the QuanChart cell range

1. Select rows **7** through **12**.
2. Right-click and choose **Copy** from the shortcut menu.
3. Right-click cell A13 and choose **Paste** from the shortcut menu.
4. Repeat Step 3 for cells A19, A25, A31, A37, and A43.

You now have 28 image areas for holding up to 28 compounds. Each of the areas displays data that looks like this:



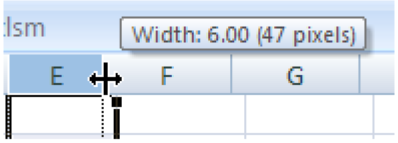
m/z-522
RT-11.2
AA-9776513
AH-2140057
Amt-0.0
11b-Hydroxyandrosterone

In the next procedures, you will adjust the columns in your spreadsheet.

❖ To adjust the ChartLegend columns

1. Click the header for column E (the first legend column).
2. Drag the separator between column E and column F until column E is 6.00 points wide.

   The Excel application displays the column width as you drag the separator.



3. Repeat Step 1 for each of the other legend columns (J, O, and T).

❖ To adjust the QuanChart columns

1. Click the header for column A (the first chart column).
2. Drag the separator between column A and column B until column A is 4.00 points wide.

   The Excel application displays the column width as you drag the separator.
3. Repeat Steps 1 and 2 for each of the other chart columns (B, C, and D), (F, G, H, and I), (K, L, M, and N), and (P, Q, R, and S).

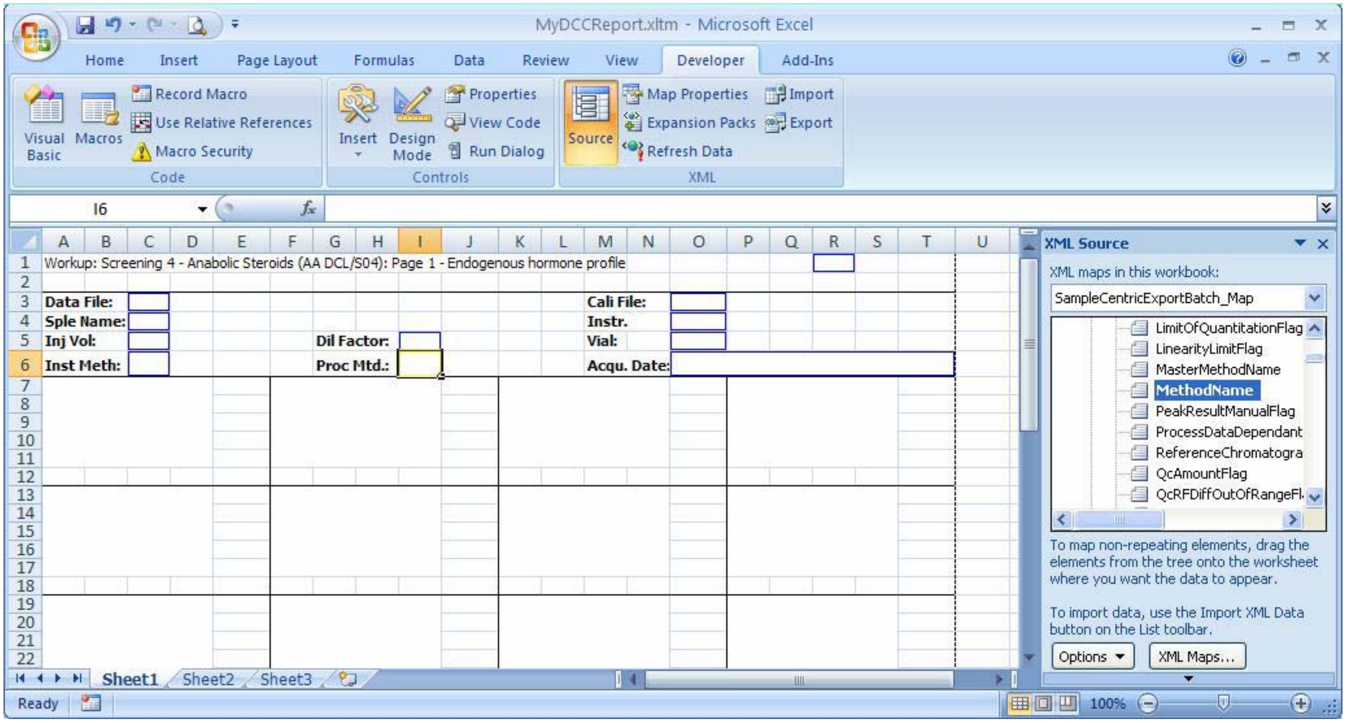
❖ To adjust the row height

1. Click the number for row 7.
2. Drag the separator between row 7 and row 8 until row 7 is 4.00 points in height.

   > **Tip** Column width is controlled by the right separator; row height is controlled by the bottom separator.
3. Repeat Steps 1 and 2 for each of the other rows (through row 48).

Ensure that your template looks like the following figure.

Figure 87. Template with 28 image areas



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile

Data File:
Sple Name:
Inj Vol:
Inst Meth:
Dil Factor:
Proc Mtd.:
Cali File:
Instr.
Vial:
Acqu. Date:

XML maps in this workbook:
SampleCentricExportBatch_Map

LimitOfQuantitationFlag
LinearityLimitFlag
MasterMethodName
MethodName
PeakResultManualFlag
ProcessDataDependant
ReferenceChromatogra
QcAmountFlag
QcRFDiffOutOfRangeFl

To map non-repeating elements, drag the elements from the tree onto the worksheet where you want the data to appear.

To import data, use the Import XML Data button on the List toolbar.

Creating a Data Area

At the bottom of the Sheet1, you will reserve up to 28 rows for displaying data for up to 28 compounds. See the "DCC Report example" on page 128.

In the following procedure, you will create the column headers and populate the data at run time using VBA code.

❖ To enter static labels for the column headers in the template

1. Type **Name** in cell A50.
2. Type **Symbol** in cell E50.

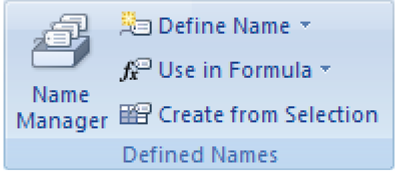
   This column will later act as a key for the VBA Vlookup function.

3. Type **RT** in cell F50.
4. Type **Cal. Amt** in cell H50.
5. Type **Resp** in cell J50.
6. Type **Type** in cell K50.
7. Type **Diagnostic Behaviors** in cell M50.

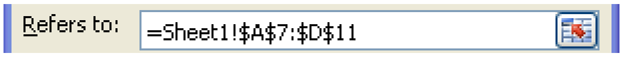
You are now ready to group the data and create a named range.

❖ To create a named range for the data in the template

1. Select cells **E51** through **K78**.
2. Click the **Formulas** tab, and then click **Define Name**.

   The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.

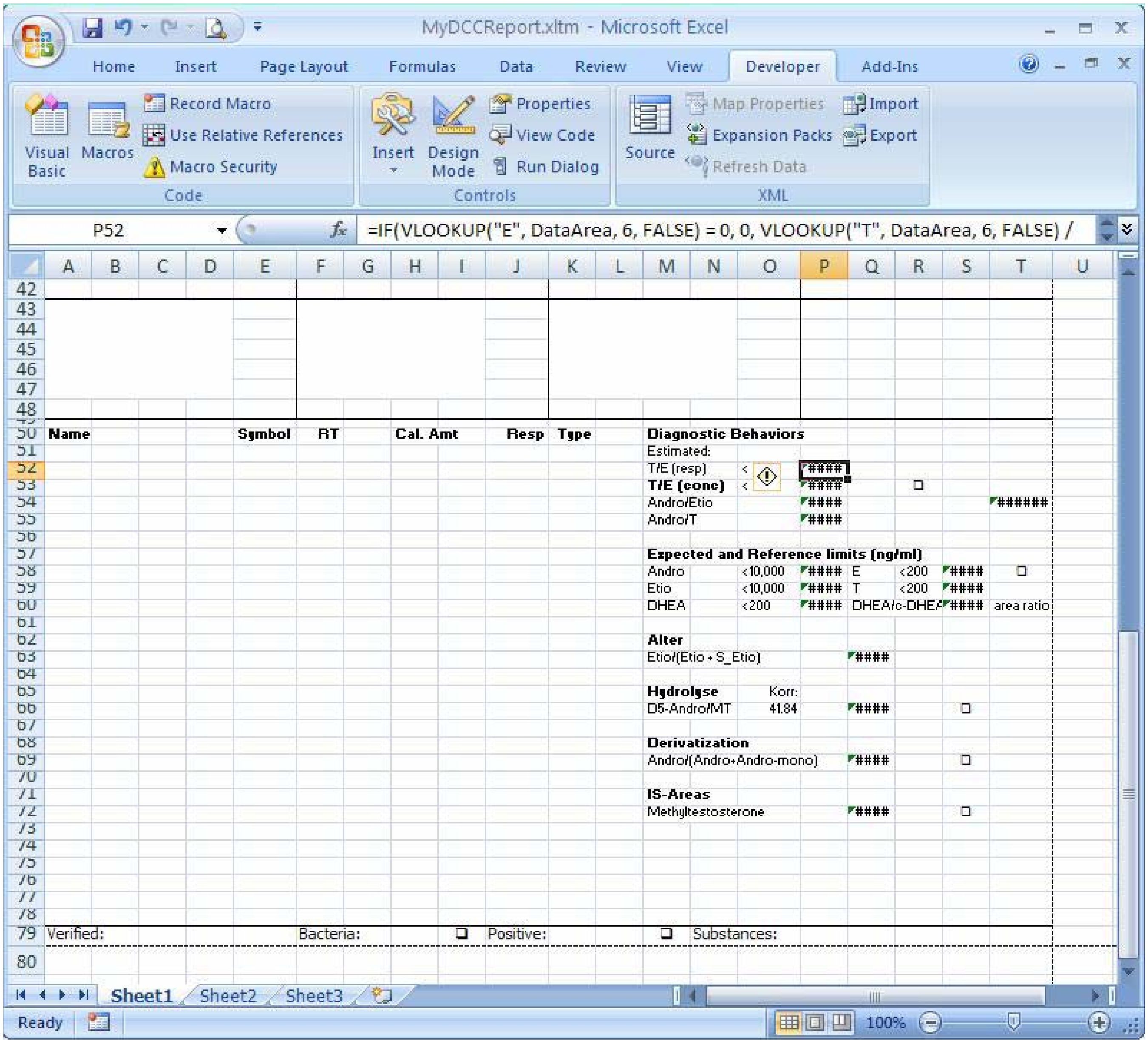
   Refers to: =Sheet1!$A$7:$D$11

3. In the Name box, type **DataArea**.
4. Click **OK**.
5. To make everything fit onto a page, make the data rows smaller. Change the row height to 9 points for rows 50 through 78.

Adding Derived Data

In this section, you will add derived data to Sheet1. The finished sheet will look like this:

Figure 88. Completed template with derived data



P52 =IF(VLOOKUP("E", DataArea, 6, FALSE) = 0, 0, VLOOKUP("T", DataArea, 6, FALSE) /

Row	A	E	F	H	I	J	K	M	N	O	P	Q	R	S	T
50	Name	Symbol	RT	Cal. Amt		Resp	Type	Diagnostic Behaviors							
51								Estimated:							
52								T/E (resp)	<		####				
53								**T/E (conc)**	<		####		□		
54								Andro/Etio			####				######
55								Andro/T			####				
57								**Expected and Reference limits (ng/ml)**							
58								Andro		<10,000	####	E	<200	####	□
59								Etio		<10,000	####	T	<200	####	
60								DHEA		<200	####	DHEA/c-DHEA		####	area ratio
62								**Alter**							
63								Etio/(Etio + S_Etio)				####			
65								**Hydrolyse**		Korr:					
66								D5-Andro/MT		41.84		####		□	
68								**Derivatization**							
69								Andro/(Andro+Andro-mono)				####		□	
71								**IS-Areas**							
72								Methyltestosterone				####		□	
79	Verified:		Bacteria:		□	Positive:		□	Substances:						

In the first procedure, you will create static labels for the derived data areas.

❖ To enter static labels for the derived data in the template

1. Type **Diagnostic Behaviors** in cell M50.
2. Type **T/E (conc)** in cell M53.

3. Type **Expected and Reference limits (ng/ml)** in cell M57.

4. Type **Alter** in cell M62.

5. Type **Hydrolyze** in cell M65.

6. Type **Derivatization** in cell M68.

7. Type **IS-Areas** in cell M71.

You are now ready to format these headings with bold font.

❖ To format the heading font in the template

1. Hold down the CTRL key and select the label cells you just created.

2. Right-click and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens. See “Format Cells dialog box - Font page” on page 131.

3. Click the **Font** tab.

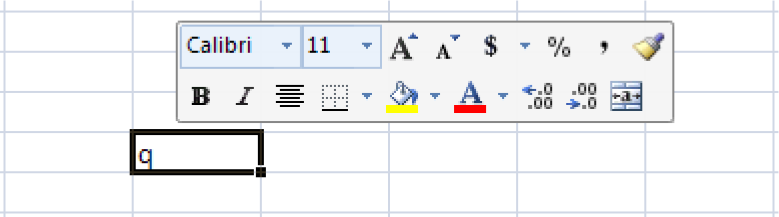
4. In the Font Style box, select **Bold**.

You are now ready to add check boxes to specific cells.

❖ To add the check boxes to the template

1. Type **q** in cell R53.

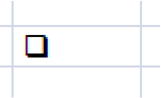
2. Right-click cell R53 to display the following pop up box.



3. Click the down arrow next to Calibri (Calibri is the spreadsheet default font).

4. From the list of fonts, choose **Wingdings**.

   The q in Wingdings font looks like this:



5. Repeat steps 1 through 4 for cells T58, S66, S69, and S72.

You are now ready to enter the formulas for the derived data.

❖ To enter the formulas in the template

1. In cell P52, enter the following formula:

   ```
   =IF(VLOOKUP("E", DataArea, 6, FALSE) = 0, 0, VLOOKUP("T", DataArea, 6,
   FALSE) / VLOOKUP("E", DataArea, 6, FALSE))
   ```

2. In cell P53, enter the following formula:

   ```
   =IF(VLOOKUP("E", DataArea, 4, FALSE) = 0, 0, VLOOKUP("T", DataArea, 4,
   FALSE) / VLOOKUP("E", DataArea, 4, FALSE))
   ```

3. In cell P54, enter the following formula:

   ```
   =IF(VLOOKUP("Etio", DataArea, 4, FALSE) = 0, 0, VLOOKUP("Andro",
   DataArea, 4, FALSE) / VLOOKUP("Etio", DataArea, 4, FALSE))
   ```

4. In cell P55, enter the following formula:

   ```
   =IF(VLOOKUP("T", DataArea, 4, FALSE) = 0, 0, VLOOKUP("Andro",
   DataArea, 4, FALSE) / VLOOKUP("T", DataArea, 4, FALSE))
   ```

5. In cell T54, enter the following formula:

   ```
   =IF(VLOOKUP("T", DataArea, 4, FALSE) = "0", "Im 0", "Not 0")
   ```

6. In cell P58, enter the following formula:

   ```
   =VLOOKUP("Andro", DataArea, 4, FALSE)
   ```

7. In cell P59, enter the following formula:

   ```
   =VLOOKUP("Etio", DataArea, 4, FALSE)
   ```

8. In cell P60, enter the following formula:

   ```
   =VLOOKUP("DHEA", DataArea, 4, FALSE)
   ```

9. In cell S58, enter the following formula:

   ```
   =VLOOKUP("E", DataArea, 4, FALSE)
   ```

10. In cell S59, enter the following formula:

    ```
    =VLOOKUP("T", DataArea, 4, FALSE)
    ```

11. In cell S60, enter the following formula:

    ```
    =IF(VLOOKUP("c-DHEA", DataArea, 6, FALSE)=0, "N/A", VLOOKUP("DHEA",
    DataArea, 6, FALSE) / VLOOKUP("c-DHEA", DataArea, 6, FALSE))
    ```

12. In cell Q63, enter the following formula:

    ```
    =IF(VLOOKUP("Etio", DataArea, 6, FALSE) = 0, 0, IF(VLOOKUP("S_Etio",
    DataArea, 6, FALSE) = 0, 0, VLOOKUP("Etio", DataArea, 6, FALSE) /
    (VLOOKUP("Etio", DataArea, 6, FALSE) + VLOOKUP("S_Etio", DataArea, 6,
    FALSE))))
    ```

13. In cell Q66, enter the following formula:

    ```
    = IF(VLOOKUP("MT", DataArea, 6, FALSE) = 0, 0,  O66 *
    VLOOKUP("D5-Andro", DataArea, 6, FALSE)/VLOOKUP("MT", DataArea, 6,
    FALSE))
    ```

14. In cell Q69, enter the following formula:

```
=IF(VLOOKUP("Andro",DataArea,6,FALSE)=0,0,IF(VLOOKUP("Andro-mono",Dat
aArea,6,FALSE)
=0,0,VLOOKUP("Andro",DataArea,6,FALSE)/(VLOOKUP("Andro",DataArea,6,FA
LSE)+VLOOKUP("Andro-mono",DataArea,6,FALSE))))
```

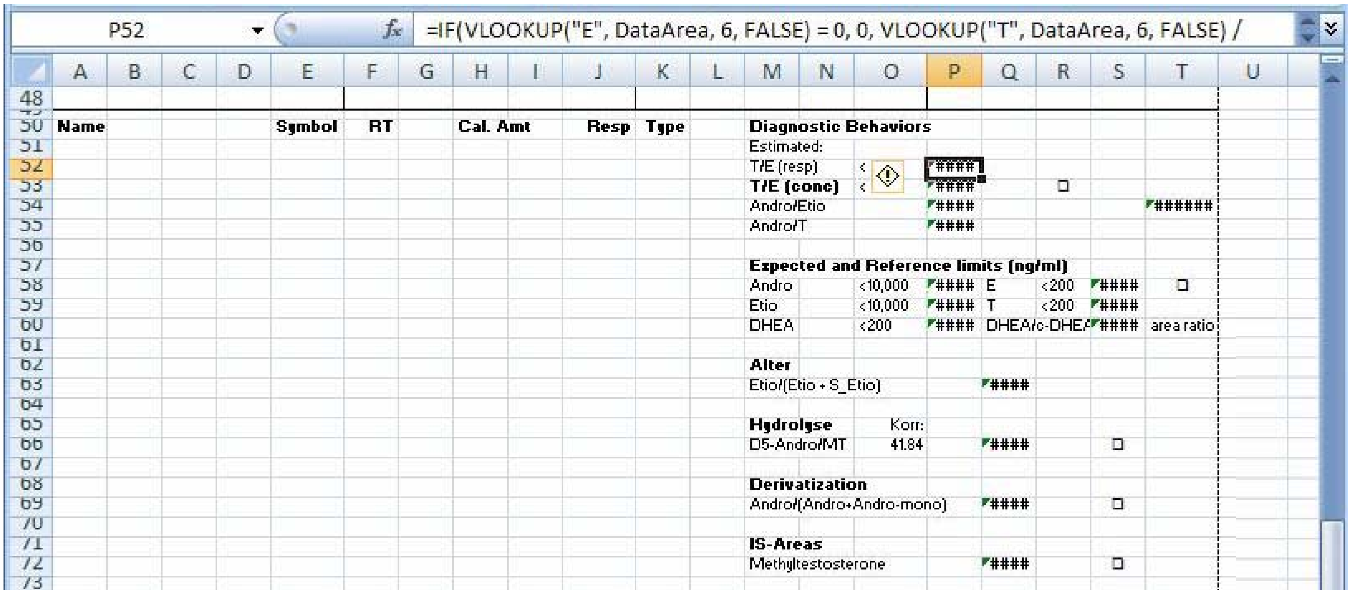
15. In cell Q72, enter the following formula:

```
=TEXT(VLOOKUP("MT", DataArea, 6, FALSE), "0,000")
```

All the formulas are included in the Excel look-up function based on the symbols defined in the named range DataArea. These formulas are evaluated and filled after data is imported and copied to the DataArea range.

The derived data area in your template looks like this:

Figure 89. Template with derived data



P52 | =IF(VLOOKUP("E", DataArea, 6, FALSE) = 0, 0, VLOOKUP("T", DataArea, 6, FALSE) /

	A	E	F	H	J	K	M	O	P	Q	R	S	T
48													
50	Name	Symbol	RT	Cal. Amt	Resp	Type	Diagnostic Behaviors						
51							Estimated:						
52							T/E (resp)	<	####				
53							T/E (conc)	<	####		□		
54							Andro/Etio		####				#######
55							Andro/T		####				
56													
57							Expected and Reference limits (ng/ml)						
58							Andro	<10,000	####	E	<200	####	□
59							Etio	<10,000	####	T	<200	####	
60							DHEA	<200	####	DHEA/c-DHEA		####	area ratio
61													
62							Alter						
63							Etio/(Etio + S_Etio)			####			
64													
65							Hydrolyse	Korr:					
66							D5-Andro/MT	41.84		####		□	
67													
68							Derivatization						
69							Andro/(Andro+Andro-mono)			####		□	
70													
71							IS-Areas						
72							Methyltestosterone			####		□	
73													

Creating the Page Footer

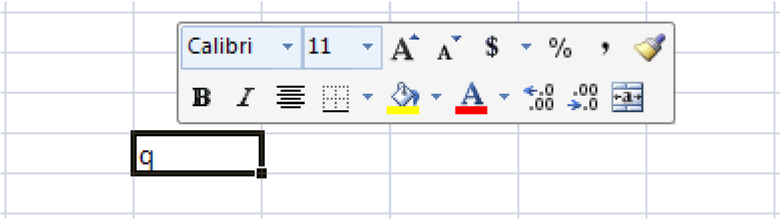
In this section, you will create a footer for the data table.

❖ To enter static labels for the report footer

1. Type **Verified:** in cell A79.
2. Type **Bacteria:** in cell A79.
3. Type **Positive:** in cell A79.
4. Type **Substances:** in cell A79.

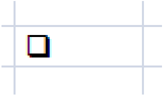
❖ To add the check boxes

1. Type **q** in cell I79.
2. Right-click cell I79 to display the following pop up box.



3. Click the down arrow next to Calibri (Calibri is the spreadsheet default font).
4. From the list of fonts, choose **Wingdings**.

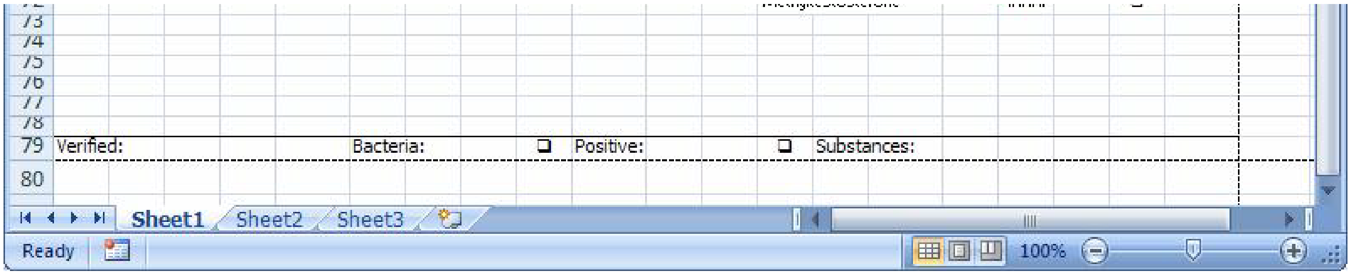
   The q in Wingdings font looks like this:



5. Repeat steps 1 through 4 for cell M79.

The footer of your template looks like this:

Figure 90. Template with footer



Creating a Compound Mapping Table

The DCC report requires listing the compounds in a specific order. You will create a mapping table that serves two purposes:

- Sorting: This report need a special sorting order using prefixes in the compound names in the method. Later, you will remove the prefixes from the report.
- Reference: In the area M51 through T72, there are many formulas that use the Excel application Vlookup function. Instead of using actual compound names, you will use a symbol for some of the compounds.

The first column of the mapping table contains the compound names that must match the names defined in the method.

The second column of the mapping table contains any names you want to display on the report.

The third column of the mapping table contains symbols for the compounds that you will reference in the formulas. Using symbols instead of compound names makes the formula easier to read and maintain (for example, you can change the display name without changing the formula).

You will use the values in the following table to create a mapping table in your template.

Table 1. Mapping table column values (Sheet 1 of 2)

Name in method	Name to be displayed	Symbol
11b-Hydroxyandrosterone	11b-Hydroxyandrosterone	
11b-Hydroxyetiocholanolone	11b-Hydroxyetiocholanolone	
16-Androstenol	16-Androstenol	
Allotetrahydrocortisol	Allotetrahydrocortisol	
Androstandiole1	Androstandiole1	
Androstandiole2	Androstandiole2	
Androstendione	Androstendione	
Androsterone	Androsterone	Andro
Cortisol	Cortisol	
Cortisone	Cortisone	
Dehydroepiandrosterone	Dehydroepiandrosterone	DHEA
Dihydrotestosterone	Dihydrotestosterone	
Epitestosterone	Epitestosterone	E
Etiocholanolone	Etiocholanolone	Etio
Etiocholanolone_S_Artefact	Etiocholanolone_S_Artefact	S_Etio

Table 1. Mapping table column values (Sheet 2 of 2)

Name in method	Name to be displayed	Symbol
Methyltestosterone	Methyltestosterone	MT
Norandrosterone	Norandrosterone	
Pregnandiole	Pregnandiole	
Testosterone	Testosterone	T
Tetrahydrocortisol	Tetrahydrocortisol	
Z-Androsterone_mono	Androsterone_mono	Andro-mono
Zd5_Stanozolol_M1	d5_Stanozolol_M1	
Zd5-Androsterone_bis	d5-Androsterone_bis	D5-Andro
Zd5-Norandrosterone	d5-Norandrosterone	
ZTestosterone-d3	Testosterone-d3	
ZZ_Cyclo_DHEA	Cyclo_DHEA	c-DHEA

❖ To enter static labels for columns in the template

1. Type **Name in Method** in cell V50.
2. Type **Name to be displayed** in cell W50.
3. Type **Symbol** in cell X50.

You are now ready to add the compounds, display names, and symbols to the mapping table.

❖ To add the compounds in the template

1. In cells V51 through V76, type the compound names.
2. In cells W51 through W76, type the display names.

   In most cases these names are the same as the compound names, but for the last few compounds, just remove the Z or ZZ from the name. These leading Zs were used to sort those compounds to the bottom of the list.

3. In the Y column, type the symbols for the specified compounds.

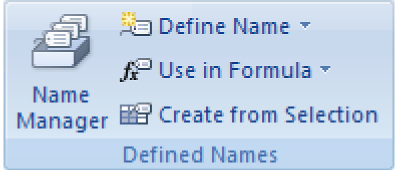
   You will use the symbols indicated here in the formulas for this report.

You are now ready to create a named range for this mapping table. You will reference this named range in the VBA code.

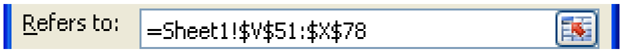
❖ To create a named range for the data in the template

1. Select cells **V51** through **X78**.

2. Click the **Formulas** tab, and then click **Define Name**.



The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.



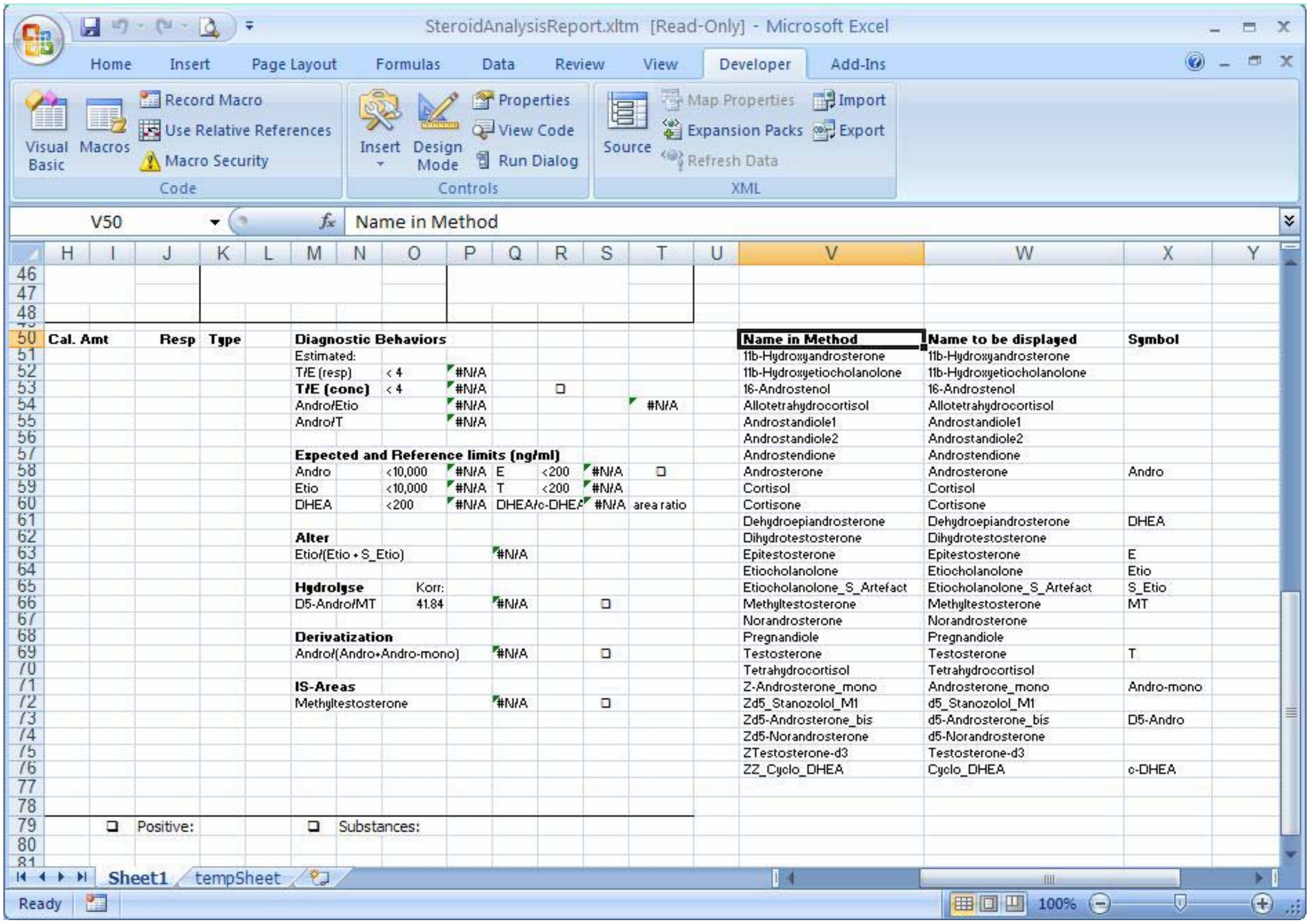
Refers to: =Sheet1!$V$51:$X$78

3. In the Name box, type **CompoundMapTable**.

4. Click **OK**.

Ensure that your template looks like the following figure.

Figure 91. Template with compound mapping table



V50 Name in Method

	H	I	J	K	M	N	O	P	Q	R	S	T
46												
47												
48												
50	**Cal. Amt**		**Resp**	**Type**	**Diagnostic Behaviors**							
51					Estimated:							
52					T/E (resp)		< 4	#N/A				
53					**T/E (conc)**		< 4	#N/A		□		
54					Andro/Etio			#N/A				#N/A
55					Andro/T			#N/A				
56												
57					**Expected and Reference limits (ng/ml)**							
58					Andro		<10,000	#N/A	E	<200	#N/A	□
59					Etio		<10,000	#N/A	T	<200	#N/A	
60					DHEA		<200	#N/A	DHEA/c-DHEA		#N/A	area ratio
61												
62					**Alter**							
63					Etio/(Etio + S_Etio)				#N/A			
64												
65					**Hydrolyse**		Korr:					
66					D5-Andro/MT		41.84		#N/A		□	
67												
68					**Derivatization**							
69					Andro/(Andro+Andro-mono)				#N/A		□	
70												
71					**IS-Areas**							
72					Methyltestosterone				#N/A		□	
79		□	Positive:		□	Substances:						

	V	W	X
50	**Name in Method**	**Name to be displayed**	**Symbol**
51	11b-Hydroxyandrosterone	11b-Hydroxyandrosterone	
52	11b-Hydroxyetiocholanolone	11b-Hydroxyetiocholanolone	
53	16-Androstenol	16-Androstenol	
54	Allotetrahydrocortisol	Allotetrahydrocortisol	
55	Androstandiole1	Androstandiole1	
56	Androstandiole2	Androstandiole2	
57	Androstendione	Androstendione	
58	Androsterone	Androsterone	Andro
59	Cortisol	Cortisol	
60	Cortisone	Cortisone	
61	Dehydroepiandrosterone	Dehydroepiandrosterone	DHEA
62	Dihydrotestosterone	Dihydrotestosterone	
63	Epitestosterone	Epitestosterone	E
64	Etiocholanolone	Etiocholanolone	Etio
65	Etiocholanolone_S_Artefact	Etiocholanolone_S_Artefact	S_Etio
66	Methyltestosterone	Methyltestosterone	MT
67	Norandrosterone	Norandrosterone	
68	Pregnandiole	Pregnandiole	
69	Testosterone	Testosterone	T
70	Tetrahydrocortisol	Tetrahydrocortisol	
71	Z-Androsterone_mono	Androsterone_mono	Andro-mono
72	Zd5_Stanozolol_M1	d5_Stanozolol_M1	
73	Zd5-Androsterone_bis	d5-Androsterone_bis	D5-Andro
74	Zd5-Norandrosterone	d5-Norandrosterone	
75	ZTestosterone-d3	Testosterone-d3	
76	ZZ_Cyclo_DHEA	Cyclo_DHEA	c-DHEA

You are now ready to hide the mapping table. The mapping table will not be visible in the generated report.

❖ To hide the mapping table columns in the template

1. Select columns **V**, **W**, and **X**.
2. Right-click and choose **Hide** from the shortcut menu.

   The Excel application hides columns V, W, and X. At any time, you can restore the contents of the hidden columns.

   To restore columns V, W, and X, do the following:

   a. Select columns **U** and **Z**.

   b. Right-click and choose **Unhide** from the shortcut menu.

Creating a Data Table

The source data reported in the cells A5 through T78 comes from the *_SampleCentricData.xml file, but because of the special sorting needs, you cannot use the data directly. In this section, you will put that imported data in a temporary sheet that you will use to fill in the report at run time.

❖ To remove Sheet3 in the template

Right-click the Sheet3 tab and choose **Delete** from the shortcut menu.

❖ To rename Sheet2 in the template

1. Right-click the Sheet2 tab and choose **Rename** from the shortcut menu.
2. Type **TempSheet** and press ENTER.

You are now ready to add mapped nodes from the SampleCentricExportBatch folder.

❖ To add mapped nodes to the template

1. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData folder and drag the **CompoundName** node from the XML Source tree to cell A1.
2. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData/QuanResult folder and drag the **ActualRTDisplayValue** node from the XML Source tree to cell B1.
3. Open the CalibrationData/CompoundCurveExportData/CalibrationReplicates/ SampleCompoundReplicateExportData folder and drag the **CalculatedAmount** node from the XML Source tree to cell C1.
4. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData/QuanResult folder and drag the **TotalResponse** node from the XML Source tree to cell D1.
5. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData folder and drag the **ResponseValue** node from the XML Source tree to cell E1.
6. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData folder and drag the **CompoundType** node from the XML Source tree to cell F1.
7. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData folder and drag the **CompoundKey** node from the XML Source tree to cell G1.

8. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData folder and drag the **QuanMassRange** node from the XML Source tree to cell H1.

9. Open the SampleCentricExportBatch/Samples/SampleCentricExportSample/ Compounds/SampleCentricCompoundExportData/QuanPeakIdentifiers/ QuanPeakIdentifierExportData/PeakDetectionParameters folder and drag the **FilterString** node from the XML Source tree to cell I1.

10. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData/QuanResult folder and drag the **TotalIonArea** node from the XML Source tree to cell J1.

11. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData/QuanResult folder and drag the **TotalIonHeight** node from the XML Source tree to cell K1.

12. Open the Samples/SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData/QuanResult folder and drag the **ManualIntegrationFlagSet** node from the XML Source tree to cell L1.

13. Open the CalibrationData/CompoundCurveExportData/CalibrationReplicates/ SampleCompoundReplicateExportData folder and drag the **Units** node from the XML Source tree to cell M1.

You are now ready to create helper columns to massage the data for display.

❖ To add seven new helper columns to the template

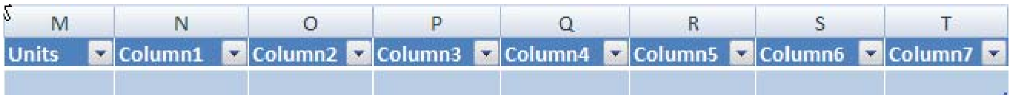
1. Right-click the Units header cell and choose **Insert > Table Column to the Right** from the shortcut menu.

   The Excel application adds a new column with the default name Column1 to the right of the M column.

2. Right-click the Column1 header cell and choose **Insert > Table Column to the Right** from the shortcut menu.

3. Continue to right-click the last column and add a column to the right until you have seven new columns named Column1–Column7.

Your new column headers should look like this:



M	N	O	P	Q	R	S	T
Units	Column1	Column2	Column3	Column4	Column5	Column6	Column7

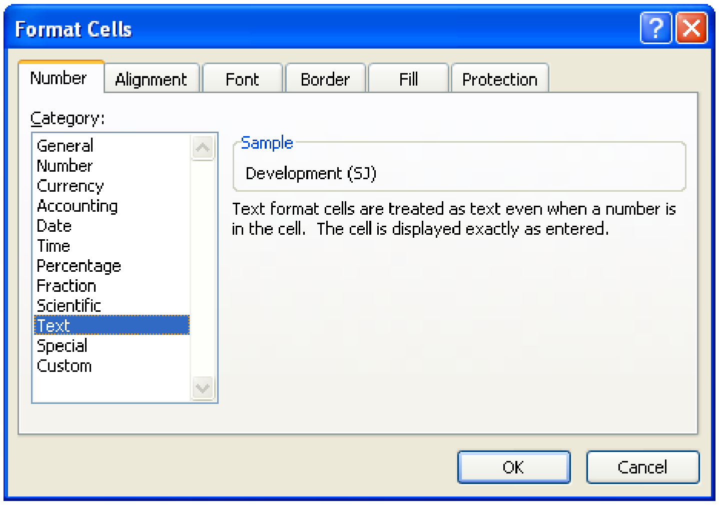
You are now ready to change the cell formats from Text to General.

❖ To change the cell format in the template

1. Right-click the Column1 cell and choose **Format Cells** from the shortcut menu.

   The Format Cells dialog box opens.

   Figure 92. Format Cells dialog box - Number page

   Category: Text

   Sample

   Development (SJ)

   Text format cells are treated as text even when a number is in the cell. The cell is displayed exactly as entered.

2. Click the **Number** tab.
3. Select **General** in the Category list and click **OK**.
4. While the Column1 cell is still selected, press ENTER.

   The Excel application does not automatically update the new cell format. Pressing ENTER manually updates it.

5. Repeat steps 1 through 5 for each of the seven columns that you added.

You are now ready to give the column headers meaningful names.

❖ To rename the column headers in the template

1. In cell N1, change Column1 to **Response**.
2. In cell O1, change Column2 to **MassCharge**.
3. In cell P1, change Column3 to **RT**.
4. In cell Q1, change Column4 to **AreaIntensity**.
5. In cell R1, change Column5 to **HeightIntensity**.
6. In cell S1, change Column6 to **Amt**.
7. In cell T1, change Column7 to **ChartLegend**.

You are now ready to enter formulas for these seven columns.

❖ To enter the formulas in the template

1. In cell N2, enter the following formula:

   ```
   =TEXT(D2, "0")
   ```

2. In cell O2, enter the following formula:

   ```
   ="m/z=" & H2
   ```

3. In cell P2, enter the following formula:

   ```
   ="RT=" & B22
   ```

4. In cell Q2, enter the following formula:

   ```
   =IF(L2, "M", "A") & "A=" & TEXT(J2, "0")
   ```

5. In cell R2, enter the following formula:

   ```
   =IF(L2,"M","A")&"H="&TEXT(K2,"0")
   ```

6. In cell S2, enter the following formula:

   ```
   ="Amt=" & TEXT(C2, "0.0")
   ```

7. In cell T2, enter the following formula:

   ```
   = O2 & CHAR(10) & P2 & CHAR(10) & Q2 & CHAR(10) & R2 & CHAR(10) & S2
   ```

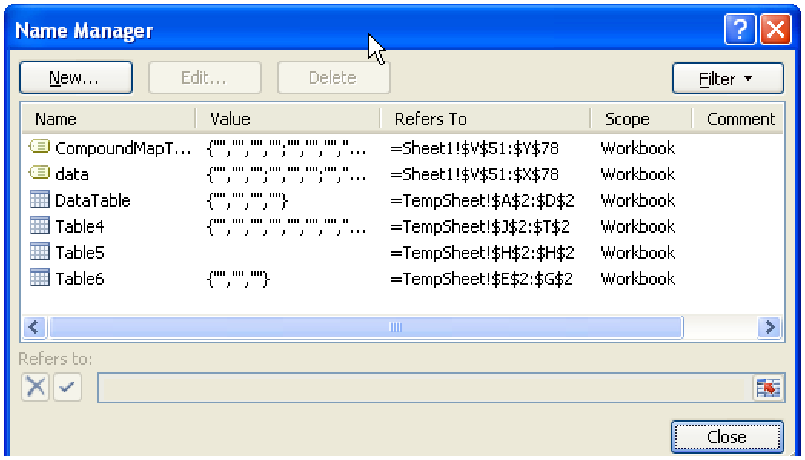
   This formula concatenates all legend fields and makes them ready to use on the report.

You are now ready to name the table you just created.

❖ To name the table in the template

1. Click the **Formulas** tab and then click **Name Manager**.

   The Name Manager dialog box opens.



Name	Value	Refers To	Scope	Comment
CompoundMapT...	{"","","","";"","","","...	=Sheet1!$V$51:$Y$78	Workbook	
data	{"","","";"","","";"","...	=Sheet1!$V$51:$X$78	Workbook	
DataTable	{"","","",""}	=TempSheet!$A$2:$D$2	Workbook	
Table4	{"","","","","","","","...	=TempSheet!$J$2:$T$2	Workbook	
Table5		=TempSheet!$H$2:$H$2	Workbook	
Table6	{"","",""}	=TempSheet!$E$2:$G$2	Workbook	

2. In the Name Manager dialog box, double-click the name of the table you created.

   The name will vary depending on how many tables you created.

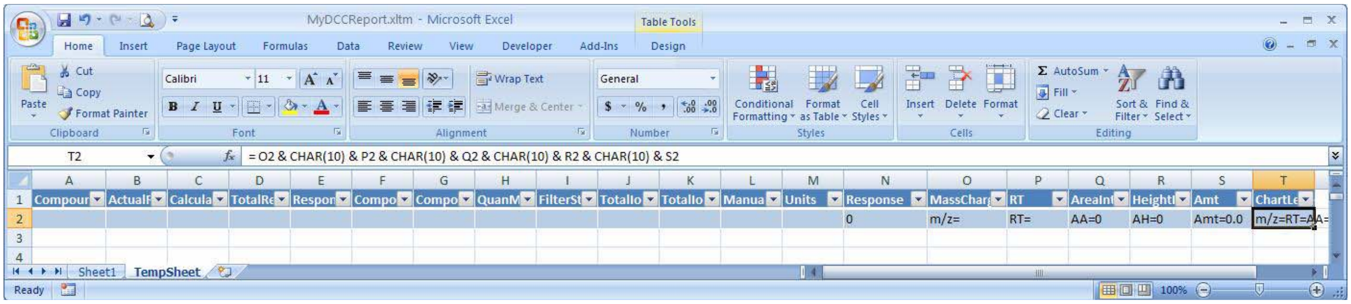
   The Edit Name dialog box opens.

3. Change the name to **DataTable** and click **OK.**

4. Click **Close** in the Name Manager dialog box.

Ensure that your template looks like the following figure.

Figure 93. Completed data table layout



T2 = O2 & CHAR(10) & P2 & CHAR(10) & Q2 & CHAR(10) & R2 & CHAR(10) & S2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Compour	ActualF	Calcula	TotalRe	Respon	Compo	Compo	QuanM	FilterSt	TotalIo	TotalIo	Manua	Units	Response	MassChar	RT	AreaInt	HeightI	Amt	ChartLe
2														0	m/z=	RT=	AA=0	AH=0	Amt=0.0	m/z=RT=AA=
3																				
4																				

The template interface design is completed. In the next section, you will add VBA code to populate the data from the DataTable on TempSheet to the columns on Sheet1.

Adding VBA Code

You are now ready to use VBA code to populate the columns on Sheet1 of the template.

❖ To connect the VBA data to the template

1. Click the **Developer** tab to display the Developer features.
2. Click the **Visual Basic** button.

   The Microsoft Visual Basic window opens.

Because this report does not use the standard header and footer, in the next procedure, you will remove the header and footer code.

❖ To remove code from the MyUtil class module

1. To remove the code that specifies the footer, type a single quote (tick mark) to comment out the footer code:

   ```
   Private Sub IUtil_SetupPageFooter()
   ' mUtil.SetupPageFooter "", 0.5
   End Sub
   ```

2. To remove the code that specifies the header, type a single quote (tick mark) to comment out the header code:

   ```
   Private Sub IUtil_SetupPageHeader()
   ' mUtil.SetupPageHeader
   End Sub
   ```

Your MyUtil code looks like this:

```
Option Explicit
Implements IUtil
Private Sub IUtil_ProcessData()
    ProcessData
End Sub
Private Sub IUtil_SetupPageFooter()
'    mUtil.SetupPageFooter "", 0.5
End Sub
Private Sub IUtil_SetupPageHeader()
'    mUtil.SetupPageHeader
End Sub
Private Function IUtil_Validate(Optional showWarning As Boolean =
True) As Boolean
    IUtil_Validate = Validate(showWarning)
End Function
```

❖ To edit the Module1 module

1. Click **Module1**.

   The Module1 window contains the code you inherited from the ThermoCustomReportBaseTemplate.xml template.

2. Replace the base template code with the following code:

```
Option Explicit
' Called after xml data gets imported
Public Sub ProcessData()
    ProcessSheet1Data
End Sub
Private Sub ProcessSheet1Data()
    Dim i As Long, r As Long, c As Long, compName As String, compId As
String, arrCompounds() As String
    Dim rr As Long, cc As Long
    ActiveWorkbook.Sheets("Sheet1").Activate
    ' Sort the data table by compound name
    mUtil.SortList "DataTable", "CompoundName", Sheets("TempSheet")
    ' Get a unique list of compound names
    arrCompounds = mUtil.GetUniqueList("DataTable", 1)
    For i = 1 To UBound(arrCompounds)
        r = Int((i - 1) / 4) + 1    ' row number
        c = (i - 1) Mod 4 + 1               ' col number
        rr = (r - 1) * 6                    ' row offset
        cc = (c - 1) * 5                    ' col offset
        ' Get the compound display name from the CompoundMapTable
        compName = mUtil.VLookup(arrCompounds(i),
Range("CompoundMapTable"), 2, False)
        If compName <> "" Then
           ' Copy compound display name to column A starting from row
51
            Range("A" & 50 + i) = compName
            ' Copy compound display name to the image area under the
image
            Range("CompoundName").Offset(rr, cc) = compName
            ' Copy symbol to column E starting from row 51
            Range("E" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("CompoundMapTable"), 3, False)
            ' Copy RT to column F starting from row 51
            Range("F" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("DataTable"), 2, False)
            ' Copy Calc Amt to column H starting from row 51
            Range("H" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("DataTable"), 3, False)
            ' Copy Units to column I starting from row 51
            Range("I" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("DataTable"), 13, False)
            ' Copy Response to column J starting from row 51
            Range("J" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("DataTable"), 14, False)
            ' Copy RespValue to column K starting from row 51
            Range("K" & 50 + i) = mUtil.VLookup(arrCompounds(i),
Range("DataTable"), 5, False)
            ' Set the legend next to the image
            mUtil.ShiftRange("ChartLegend", cc, rr, cc, rr) =
mUtil.VLookup(arrCompounds(i), Range("DataTable"), 20, False)
            ' Merge the legend cells and make the text wrap
            With mUtil.ShiftRange("ChartLegend", cc, rr, cc, rr)
            .MergeCells = True
            .WrapText = True
            End With
            Application.Calculate   ' Refresh formula
            ' Load images in place
```

```
          compId = mUtil.VLookup(arrCompounds(i), Range("DataTable"),
7, False)

          mUtil.CreateImage mUtil.ShiftRange("QuanChart", cc, rr, cc,
rr),
"SampleCentricExportBatch/Samples/SampleCentricExportSample/Compounds
/SampleCentricCompoundExportData[CompoundKey='" & compId &
"']/QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/PeakI
mage", 20, 5, 12, 2

        End If

    Next

    mUtil.HideColumns "V:AA"

    ActiveWorkbook.Sheets("TempSheet").Delete

'    mUtil.AddTextAsWatermark "Thermo Custom Report",
Range("Sheet1!A1:T78")

End Sub


' Validate the template design e.g. named ranges

Public Function Validate(Optional showWarning As Boolean = True) As
Boolean

    Validate = True

End Function
```

Where:

- **SortList** sorts the data table by the compound name.
- **GetUniqueList** retrieves a unique list of compound names that will be used for looping into each compound.
- **CopyPaste** copies the following columns from the data table on TempSheet to the data area at the bottom of Sheet1:
  - DisplayRetentionTime
  - CalculatedAmount
  - Units
  - Response
  - ResponseValue
- To loop through for each compound, the VBA functions do the following:
  - Calculate the row and column offset that will be used to place the compound name, image and legend for each compound.
  - **VLookup** gets the compound display from the CompoundMapTable.
  - **Range("A" & 50 + i)…** copies the compound display name to the data area starting at A1 below each quantitation chart.

  - **Range("CompoundName").Offset(rr, cc)…** copies the compound display name to the data area below each quantitation chart.
  - **Range("E" & 50 + i)…** copies the symbol to column E starting from E51.
  - **ShiftRange** shifts the legend for the image by the specified offset and copies it to the ChartLegend named range.
  - Merge the legend cells and set the WrapText property to true so that no text runs over the next cell.
  - Load the image from the XML file based on the CompoundKey.

- **ActiveWorkbook.Sheets("TempSheet").Delete** deletes the TempSheet.

3. To use the Visual Basic compile function to check for errors, choose **Debug > Compile VBAProject**.

To use this template to create an Excel report, you must have specific compounds defined in your method or you must modify the CompoundMapTable to fit your method.

Testing the Template

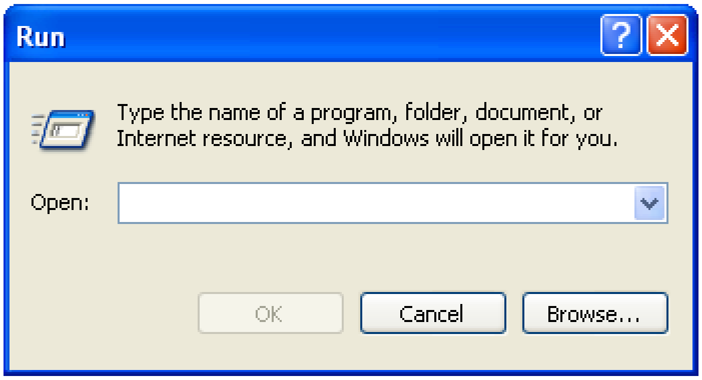
You are now ready to run a test from the Windows **Start** > **Run** box. In this test, the code you enter does the following:

- Opens the Excel application.
- Creates a new workbook using the MyDCCReport.xltm template.
- Passes the parameters to the template.
- Generates a report based on the 4bw01_SampleCentricData.xml source file.
- Saves the report to the MyDCCReport.xlsm file.

❖ To run the test

1. Choose **Start** > **Run**.

   The Run dialog box opens.



Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open:

Tip By default, the TraceFinder application file structure results in a long file path. The Windows Run dialog box limits you to 256 characters. To work around this limitation, either copy files to a folder close to the root directory or put the command line into a batch file.

2. Enter this code in the Open box:

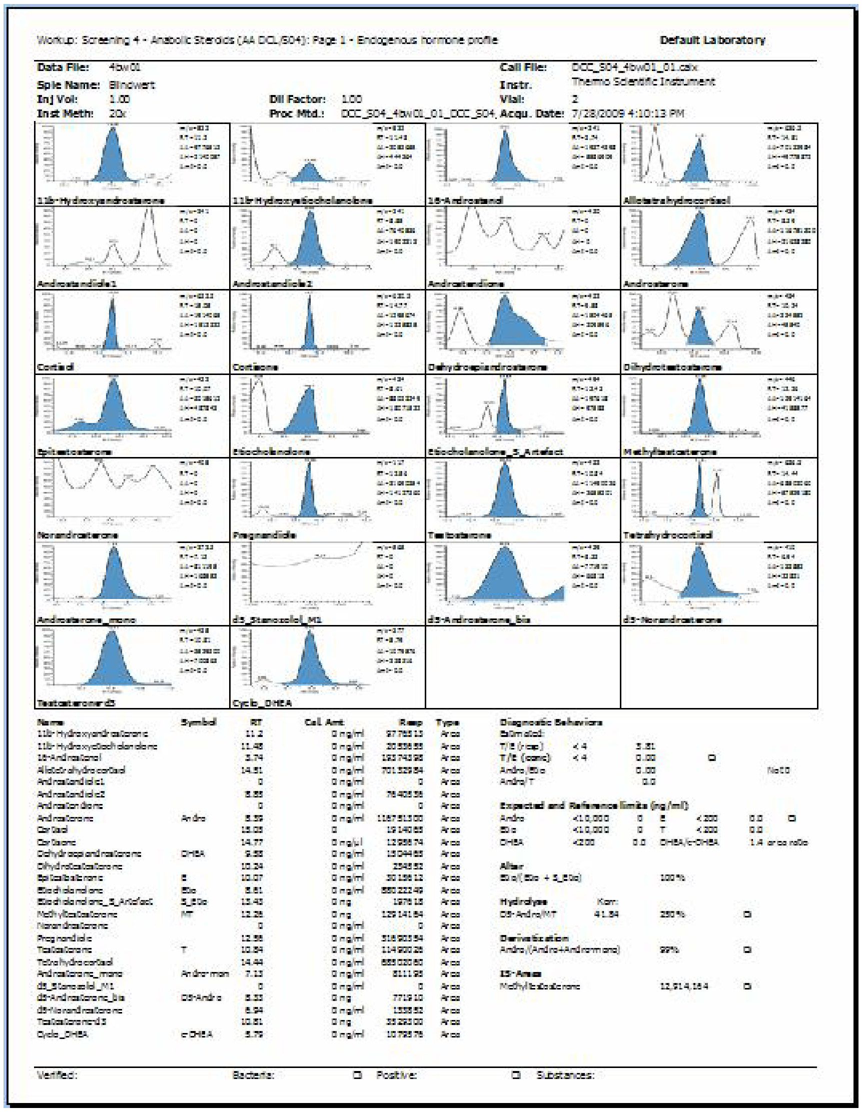
```
Excel.exe /n "C:\Thermo\Custom Report Tutorials\MyDCCReport.xltm"
/x/C:\Thermo\Custom|Report|Tutorials\Data\4bw01_SampleCentricData.xml
/C:\Thermo\Custom|Report|Tutorials\Reports\MyDCCReport.xlsm/s/h
```

   For detailed descriptions of the parameters used in this code, see "Testing the Template" on page 25.

3. Click **OK**.

Ensure that the generated report on your screen is similar to the following figure (actual data will vary depending on the batch you used):

Figure 94. DCC Report example



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile — **Default Laboratory**

Data File:	4bw01			**Cal File:**	DCC_S04_4bw01_01.calx
Sple Name:	Blindwert			**Instr.**	Thermo Scientific Instrument
Inj Vol:	1.00	**Dil Factor:**	1.00	**Vial:**	2
Inst Meth:	20x	**Proc Mtd.:**	DCC_S04_4bw01_01_DCC_S04.	**Acqu. Date:**	7/28/2009 4:10:13 PM

Name	Symbol	RT	Cal. Amt	Resp	Type
11b-Hydroxyandrosterone		11.2	0 ng/ml	9776313	Area
11b-Hydroxyetiocholanolone		11.48	0 ng/ml	2053655	Area
16-Androstenol		3.74	0 ng/ml	19374398	Area
Allotetrahydrocortisol		14.51	0 ng/ml	70152984	Area
Androstendiole1		0	0 ng/ml	0	Area
Androstendiole2		8.85	0 ng/ml	7640536	Area
Androstendione		0	0 ng/ml	0	Area
Androsterone	Andro	8.59	0 ng/ml	116751300	Area
Cortisol		15.03	0	1914083	Area
Cortisone		14.77	0 ng/µl	1298874	Area
Dehydroepiandrosterone	DHEA	9.58	0 ng/ml	1504483	Area
Dihydrotestosterone		10.24	0 ng/ml	254852	Area
Epitestosterone	E	10.07	0 ng/ml	3013612	Area
Etiocholanolone	Eto	8.61	0 ng/ml	58022249	Area
Etiocholanolone_5_Artefact	5_Eto	13.43	0 ng	197613	Area
Methyltestosterone	MT	12.26	0 ng	12914184	Area
Norandrosterone		0	0 ng/ml	0	Area
Pregnandiole		12.58	0 ng/ml	31890354	Area
Testosterone	T	10.54	0 ng/ml	11490026	Area
Tetrahydrocortisol		14.44	0 ng/ml	88502080	Area
Androsterone_mono	Andro mon	7.13	0 ng/ml	811198	Area
d3_Stanozolol_M1		0	0 ng/ml	0	Area
d5-Androsterone_bis	D5-Andro	8.53	0 ng	771910	Area
d5-Norandrosterone		8.94	0 ng/ml	153352	Area
Testosterone d3		10.51	0 ng	33293300	Area
Cyclo_DHEA	c-DHEA	5.79	0 ng/ml	1079378	Area

Diagnostic Behaviors
Estimated:

T/E (resp)	< 4	3.81		
T/E (conc)	< 4	0.00	□	
Andro/Eto		0.00		NoCD
Andro/T		0.0		

Expected and Reference limits (ng/ml)

Andro	<10,000	0	E	< 200	0.0	□
Eto	<10,000	0	T	< 200	0.0	
DHEA	<200	0.0	DHEA/c-DHEA		1.4 area ratio	

Alter
Eto/(Eto + 5_Eto) 100 %

Hydrolyse Korr:
D5-Andro/MT 41.84 250 % □

Derivatization
Andro/(Andro+Andro mono) 99% □

IS-Areas
Methyltestosterone 12,914,184 □

Verified: Bacteria: □ Positive: □ Substances:

You are now ready to integrate the template into the TraceFinder application.

❖ To use the template in the TraceFinder application

1. Copy your new template into the dedicated folder of the application:

   C:\Thermo\Shared\Templates\Reports

2. Use this template the same as the delivered reports in the TraceFinder application.

   For detailed information about how to use custom reports, refer to the *TraceFinder User Guide* or Chapter 3, "Getting Started."

Frequently Asked Questions

This appendix answers frequently asked questions about TraceFinder custom reports. Additional instructions in response to these questions are given in the tutorial chapters.

Contents

General Questions

This section covers general questions about TraceFinder custom reports.

- How Do I Find the Excel Version Number?
- How Do I Enable the Developer Tab?
- What Is a Named Range in Excel?
- How Do I Use the Command Line to Generate the Excel Report?
- Why Does My Excel Report Show a Security Warning When I Open It?
- Why Are Page Numbers Not Sequential Across All Pages?
- How Do I Change the Default Logo Image for All Reports?
- Why Do I Need to Use the Add-in Copy/Paste Function Instead of the Excel Copy Function?
- How Do I Add a Watermark to a Report?

How Do I Find the Excel Version Number?

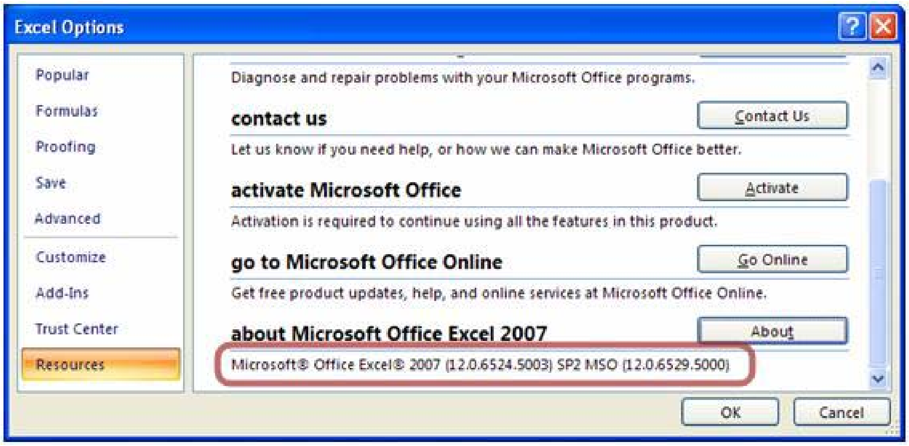
Follow this procedure to find the version number of the Excel application.

❖ To find the Excel version number

1. Click the **Excel application** icon, .
2. At the bottom of the menu, click **Excel Options**.
3. In the left pane of the Excel Options dialog box, click **Resources**.

The Resources page of the Excel Options dialog box shows the version number.

Figure 95. Excel Options Resources dialog box



Diagnose and repair problems with your Microsoft Office programs.

contact us

Let us know if you need help, or how we can make Microsoft Office better.

activate Microsoft Office

Activation is required to continue using all the features in this product.

go to Microsoft Office Online

Get free product updates, help, and online services at Microsoft Office Online.

about Microsoft Office Excel 2007

Microsoft® Office Excel® 2007 (12.0.6524.5003) SP2 MSO (12.0.6529.5000)

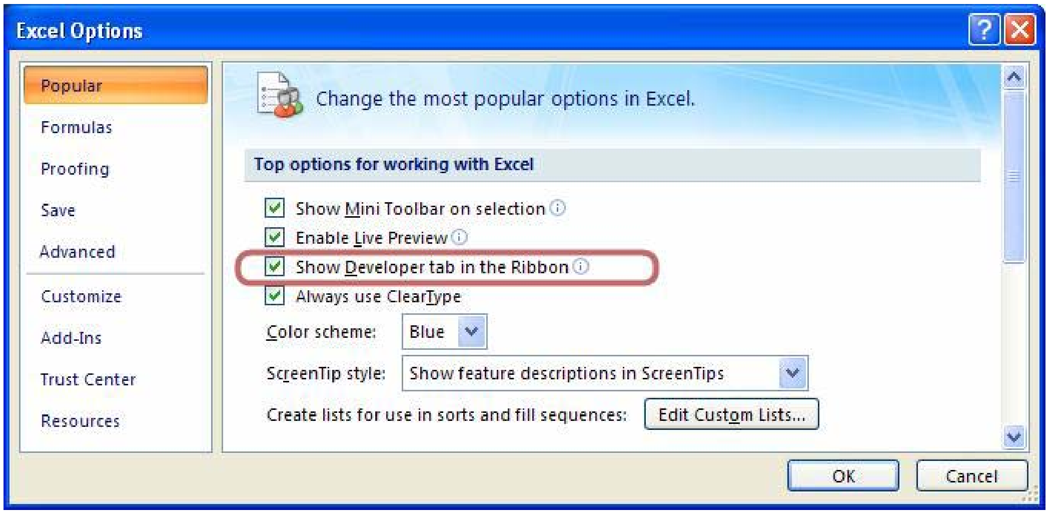
How Do I Enable the Developer Tab?

To start Visual Basic development in the Excel application, you must enable the Developer tab, which is hidden by default.

❖ To enable the Developer tab

1. Click the **Excel application** icon, .
2. Click the **Excel Options** button, Excel Options .
3. Click **Popular**.
4. Select the **Show Developer Tab in the Ribbon** check box.

Figure 96. Excel Options Popular dialog box



Excel Options

Change the most popular options in Excel.

Top options for working with Excel

☑ Show Mini Toolbar on selection
☑ Enable Live Preview
☑ Show Developer tab in the Ribbon
☑ Always use ClearType

Color scheme: Blue

ScreenTip style: Show feature descriptions in ScreenTips

Create lists for use in sorts and fill sequences: Edit Custom Lists...

What Is a Named Range in Excel?

You can define a name for a range of cells, and you can use this name in formulas or the VBA code to refer to that cell range.

❖ To specify a name for a range of cells

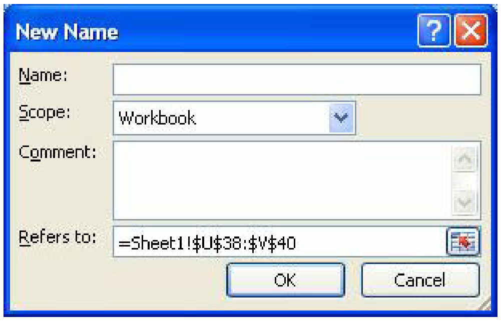
1. Select the cells.

2. Click the **Formulas** tab.

3. Click **Define Name**.

   The selected cell range automatically appears in the **Refers To** box of the New Name dialog box.

   Figure 97. New Name dialog box



New Name

Name:

Scope: Workbook

Comment:

Refers to: =Sheet1!$U$38:$V$40

The Excel application automatically assigns a default name to some objects, for example, data tables and shapes.

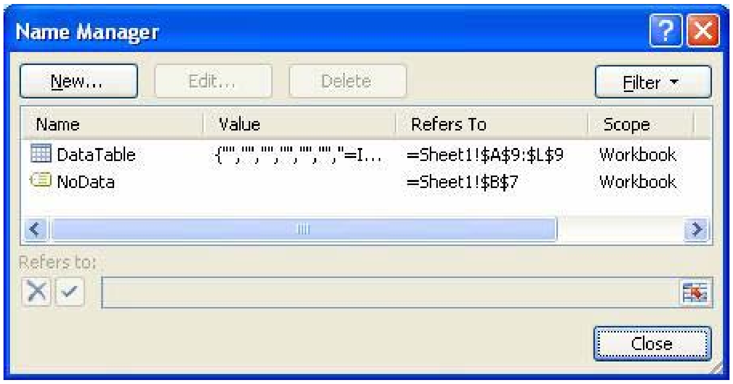
❖ To add, remove, or modify a name

1. Click the **Formulas** tab.

2. Click **Name Manager**.

   The Name Manager dialog box opens.

   Figure 98. Name Manager dialog box



Name Manager

Name	Value	Refers To	Scope
DataTable	{"","","","","","","=I...	=Sheet1!$A$9:$L$9	Workbook
NoData		=Sheet1!$B$7	Workbook

Refers to:

3. In the Name Manager dialog box, do one of the following:

- Click **New** and create a new name.

  –or–

- Select the name to edit and click **Edit** or **Delete**.

How Do I Use the Command Line to Generate the Excel Report?

When you develop a new report template or modify an existing one, you might want to test it before using it in the application. You can use the Windows Run box in the Start menu for this purpose. Use this command:

```
Excel.exe /n [/embedded] TEMPLATEFILE
/x/SAMPLECENTRICDATAFILE/OUTPUTFILE[/s][/h][/p PRINTERNAME]
[/v][/c][/h][/f][/l][/b][/g[/t][/d]]
```

Following is an example:

```
Excel.exe /n “C:\Thermo\Custom Report Tutorials\MyBatchReport.xltm”
/x/C:\Thermo\Custom|Report|Tutorials\Data\1ngrep5_SampleCentricData.x
ml/C:\Thermo\Custom|Report|Tutorials\Reports\BatchReport.xlsm/s/h
```

where:

- The /n switch tells Windows that you are using this template to create an Excel spreadsheet, not to edit the template itself. Without this switch, the Excel application opens the specified template for editing.
- The /embedded switch tells Windows to run Excel embedded—that is, in silent mode. In this case, you do not see the Excel spreadsheet on the screen, but it is still visible in the Task Manager. If this switch is on, make sure you also include the /c custom switch to automatically close the Excel application after it generates the report. Otherwise, the Excel application runs until you shut down Windows or you end it in the Task Manager.
- The template file path must be enclosed in quotation marks if it contains any spaces. Otherwise, the Excel application considers each item separated by a space as a separate Excel workbook and tries to load it. As a result, you receive an error message.

> **IMPORTANT** There must be a space before /x to tell Windows to pass everything after this space to the Excel template. Do not add a quotation mark after this space. If you do, Windows considers it an instruction to open a second spreadsheet, and you receive an error message.

- The /x switch is an internal switch special to custom reports. The custom report add-in processes all parameters after /x.

> **IMPORTANT** Parameters after /x cannot contain spaces. This requirement is imposed by the Excel application and the operating system. If you need a space, use a vertical bar (|) instead. The add-in replaces all bars with spaces. The example given earlier in this section uses vertical bars in the name of the Custom Report Tutorials folder to illustrate.

- The custom parameters after /x use a slash, /, as a separator. The first two parameters after /x are required. All others are optional.

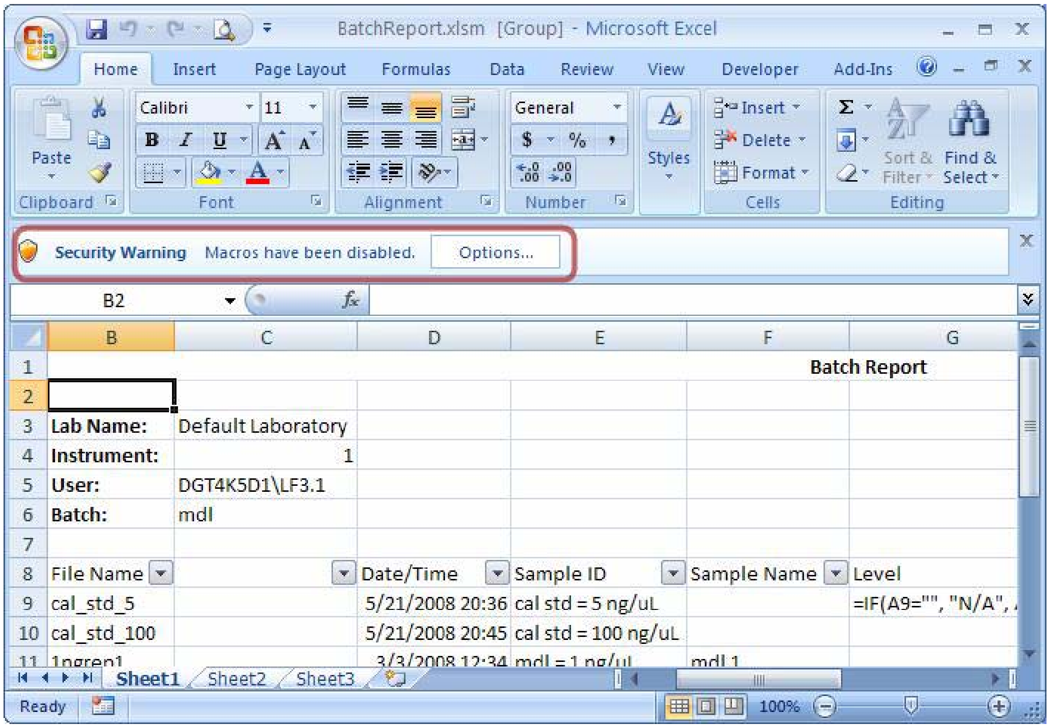
- The first parameter after /x tells the add-in where the XML Source data is located. The template location does not matter, as long as you correctly specify the path name. However, for the TraceFinder application to use the template, the template must be located in a dedicated folder.
- The second parameter after /x specifies the name of the path where the add-in saves the generated Excel spreadsheet.
- The custom switches are the following:
  - /s tells the add-in to save the file in the location specified by the second custom parameter before closing the workbook. Without this switch, the spreadsheet is generated, but not saved.
  - /h tells the Excel application to hide the screen update. This switch makes the report generation faster, especially when the report is large. It also reduces screen flashes.
  - /p prints the report. You can optionally specify the printer after this switch. If you do not specify a printer, the switch uses the default printer.
  - /v previews what will be printed. You must manually close the preview window before the printing can continue.
  - /c closes the Excel workbook after it generates the report.
  - /f saves the Excel spreadsheet as a PDF file.
  - /l locks the sheet with an optional password. When protected, a sheet is read-only. To modify it, unlock the sheet by using the Review tab in the Excel window.
  - /b copies data using Excel's native copy function, which uses the Clipboard for transferring data. Thermo Fisher Scientific recommends that you avoid using this function, because generating reports is a long process running in the background, and using the Clipboard could cause some unwanted side effects.
  - /g creates a log file to log all messages generated by the LogMsg function. The log file, named ThermoCustomReport.log, resides in the current user's temporary folder. If this switch is on, the /d and /t switches add a time stamp to the log file name.
  - /d adds a date to the log file name in the format ThermoCustomReport_*yyyyMMdd*.log, where *yyyy* is the year (for example, 2010), *MM* is the month of the year (for example, 12 for December), and *dd* is the two-digit day of the month (for example, 06 for the sixth day).
  - /t adds a time stamp to the log file name in the format ThermoCustomReport_*yyyyMMdd_HHmmss*.log, where *yyyy* is the year, *MM* is the month of the year, *dd* is the day of the month, *HH* is the hour in 24-hour format, *mm* is the minute, and *ss* is the second. An example is ThermoCustomReport_20100902_150208.log.

Tip By default, the TraceFinder application file structure results in a long file path. The Windows Run dialog box limits you to 256 characters. To work around this limitation, either copy files to a folder close to the root directory or put the command line into a batch file.

Why Does My Excel Report Show a Security Warning When I Open It?

The Excel application has a strict security requirement to prevent unintended execution of the macros, and it applies this requirement to the templates and reports. The Excel application displays the following security warning message when you open a spreadsheet with macros or VBA code from an untrustworthy location.

Figure 99. Security warning message



Security Warning Macros have been disabled. Options...

	B	C	D	E	F	G
1					Batch Report	
2						
3	Lab Name:	Default Laboratory				
4	Instrument:	1				
5	User:	DGT4K5D1\LF3.1				
6	Batch:	mdl				
7						
8	File Name		Date/Time	Sample ID	Sample Name	Level
9	cal_std_5		5/21/2008 20:36	cal std = 5 ng/uL		=IF(A9="", "N/A",
10	cal_std_100		5/21/2008 20:45	cal std = 100 ng/uL		
11	1ngrep1		3/3/2008 12:34	mdl = 1 ng/uL	mdl 1	

Without the enabled macros, the add-in component cannot run and generate a report.

In order for the macro-enabled templates to run in the Excel application, you must add the directory of the add-in and the template files to the list of trusted locations.

- If you installed the TraceFinder application, the installer should have added the C:\Thermo\Shared\Templates\Reports default template folder to the Excel application's list of trusted locations. The security setting is adequate, as long as you run your templates from this folder.
- You can add a folder to the list of trusted locations (for example, if you decide to save the template in a different folder) by following the next procedure.

❖ To add a folder to the list of trusted locations

1. Click the **Excel application** icon, .
2. Click **Excel Options**.
3. Click **Trust Center**.

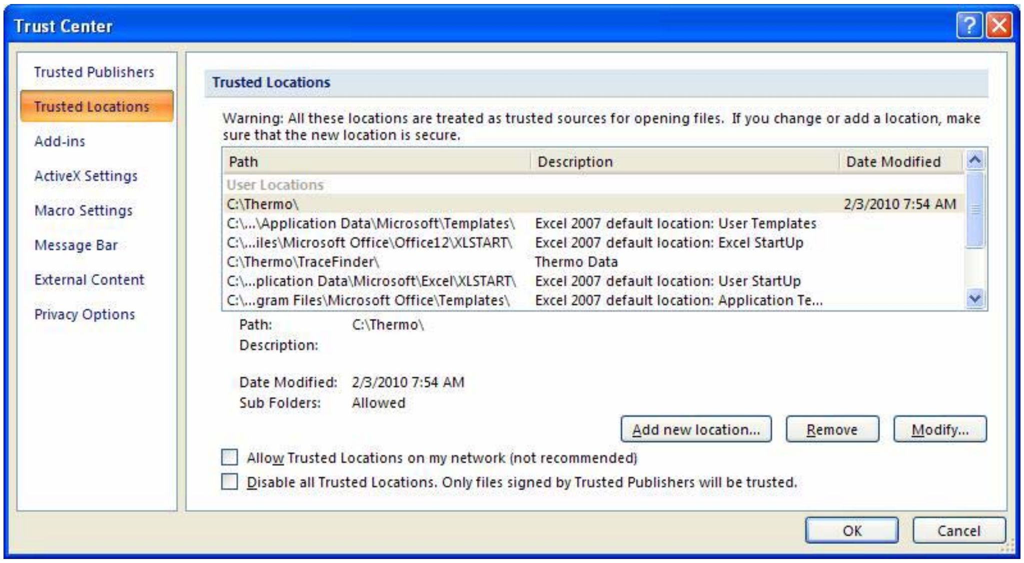
4. Click **Trust Center Settings**.

5. Click **Trusted Locations**.

6. Click **Add New Location**.

7. In the Microsoft Office Trusted Location dialog box, add the new location.

Figure 100. Trust Center dialog box



Trusted Locations

Warning: All these locations are treated as trusted sources for opening files. If you change or add a location, make sure that the new location is secure.

Path	Description	Date Modified
User Locations		
C:\Thermo\		2/3/2010 7:54 AM
C:\...\Application Data\Microsoft\Templates\	Excel 2007 default location: User Templates	
C:\...iles\Microsoft Office\Office12\XLSTART\	Excel 2007 default location: Excel StartUp	
C:\Thermo\TraceFinder\	Thermo Data	
C:\...plication Data\Microsoft\Excel\XLSTART\	Excel 2007 default location: User StartUp	
C:\...gram Files\Microsoft Office\Templates\	Excel 2007 default location: Application Te...	

Path: C:\Thermo\
Description:

Date Modified: 2/3/2010 7:54 AM
Sub Folders: Allowed

☐ Allow Trusted Locations on my network (not recommended)
☐ Disable all Trusted Locations. Only files signed by Trusted Publishers will be trusted.

Why Are Page Numbers Not Sequential Across All Pages?

In Excel, the page numbers normally start from 1 for each sheet. However, you can assign sequential page numbers to multiple sheets by grouping the sheets together.

When you generate a report through a custom report add-in, all sheets in the workbook are automatically grouped to make sure that they have sequential page numbers. Grouping is especially important for some reports, for example, the Confirmation Report, when each page is placed in a separate sheet for better performance. Under certain circumstances, grouped sheets might not be obvious.

You can break this grouping by clicking any of the inactive sheets. You can also break the grouping by right-clicking any tab and selecting **Ungroup Sheets** from the menu.

If pages are not numbered sequentially across all pages, it is probably because you have manually edited the sheets (see "Why is My Data Table Not Editable?" on page 193) and forgotten to group them afterward, or the sheet grouping was unintentionally broken, for example, by clicking inactive sheets.

To group sheets, right-click the tab and select **Select All Sheets** from the menu. You can also group or select sheets individually by using the CTRL and SHIFT keys while clicking the sheet tab.

How Do I Change the Default Logo Image for All Reports?

By default, custom reports use the logo image saved as C:\Thermo\Shared\Templates\Reports\mylogo.jpg. You can replace this file with your own logo image.

❖ To change the logo for a specific report

1. Use the IUtil_SetupPageHeader() function of the MyUtil class.

   This function normally calls the mUtil.SetupPageFooter add-in function to create the default header.

2. Replace the default header with your code.

Why Do I Need to Use the Add-in Copy/Paste Function Instead of the Excel Copy Function?

The Excel copy function is powerful in duplicating formatted cells, but it uses the Clipboard to transfer data. The Clipboard is a system-wide resource shared by all running applications, and report generation is a long process that normally runs in background. Therefore, using the Clipboard to transfer data might cause some unwanted side effects. For example, if you interactively use the Clipboard in the foreground to write e-mail while the reporting process runs in the background, the Excel application might inadvertently generate an incorrect report.

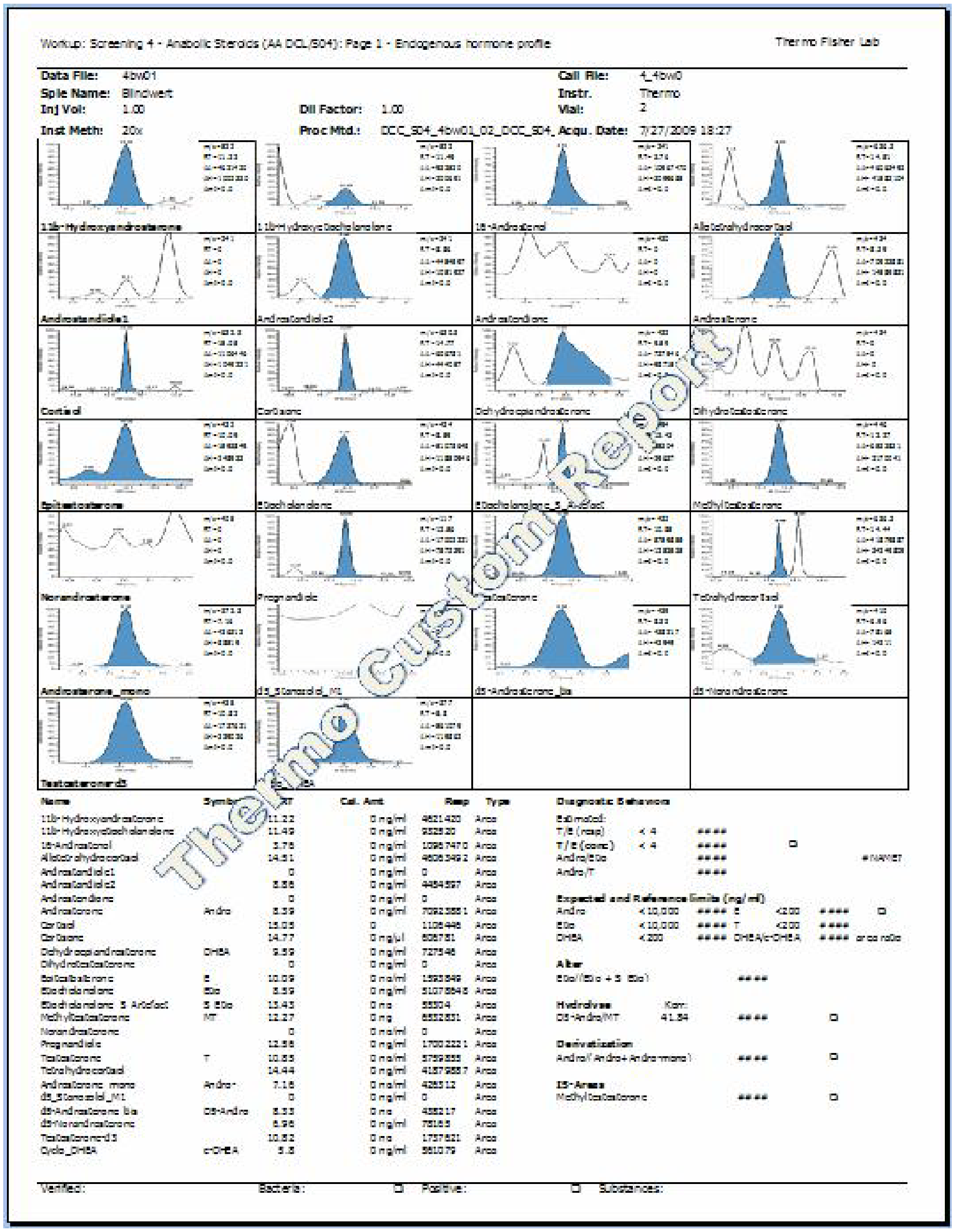
The Excel application lacks a copy function that does not use the Clipboard; however, you can use a special copy/paste function created by the add-in component. It is similar to the Excel native copy function except for its Clipboard usage. When you want to copy and paste cell data in the VBA code, use the add-in's copy/paste function rather than the Excel copy function. For instructions on using this function, see Appendix B, "Custom Report Add-in API."

How Do I Add a Watermark to a Report?

There are two add-in functions for adding watermarks to reports: one adds text as a watermark, and the other adds an image as a watermark. The following function call adds a text watermark to all the pages of the report:

```
mUtil.AddTextAsWatermark "Thermo Custom Report",
Range("Sheet1!A1:T78")
```

Figure 101. Custom report with a watermark



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile

Thermo Fisher Lab

Data File:	4bw01			**Cali File:**	4_4bw0
Sple Name:	Blindwert			**Instr.**	Thermo
Inj Vol:	1.00	**Dil Factor:**	1.00	**Vial:**	2
Inst Meth:	20x	**Proc Mtd.:**	DCC_S04_4bw01_02_DCC_S04_	**Acqu. Date:**	7/27/2009 18:27

Name	Symbol	RT	Cal. Amt	Resp	Type
11b-Hydroxyandrosterone		11.22	0 ng/ml	4621420	Area
11b-Hydroxyetiocholanolone		11.49	0 ng/ml	952520	Area
16-Androstenol		3.78	0 ng/ml	10987470	Area
Allotetrahydrocortisol		14.51	0 ng/ml	46063492	Area
Androstendiol1		0	0 ng/ml	0	Area
Androstendiol2		8.88	0 ng/ml	4484597	Area
Androstendione		0	0 ng/ml	0	Area
Androsterone	Andro	8.39	0 ng/ml	70923881	Area
Cortisol		15.05	0	1108446	Area
Cortisone		14.77	0 ng/ul	806781	Area
Dehydroepiandrosterone	DHEA	9.59	0 ng/ml	727546	Area
Dihydrotestosterone		0	0 ng/ml	0	Area
Epitestosterone	E	10.09	0 ng/ml	1593849	Area
Etiocholanolone	Eto	8.59	0 ng/ml	51078848	Area
Etiocholanolone_5_Artefact	5_Eto	13.43	0 ng	55304	Area
Methyltestosterone	MT	12.27	0 ng	6552831	Area
Norandrosterone		0	0 ng/ml	0	Area
Pregnandiole		12.58	0 ng/ml	17002221	Area
Testosterone	T	10.85	0 ng/ml	5759855	Area
Tetrahydrocortisol		14.44	0 ng/ml	41879887	Area
Androsterone_mono	Andro-	7.18	0 ng/ml	426512	Area
d5_Stanozolol_M1		0	0 ng/ml	0	Area
d5-Androsterone_bis	D5-Andro	8.33	0 ng	438217	Area
d5-Norandrosterone		6.96	0 ng/ml	78165	Area
Testosterone d3		10.82	0 ng	1737621	Area
Cyclo_DHEA	c-DHEA	5.8	0 ng/ml	561079	Area

Diagnostic Behaviors

Estimated:						
T/E (resp)	< 4	####				
T/E (conc)	< 4	####		□		
Andro/Eto		####				#NAME?
Andro/T		####				

Expected and Reference limits (ng/ml)

Andro	< 10,000	####	E	<200	####	□
Eto	< 10,000	####	T	<200	####	
DHEA	< 200	####	DHEA/c-DHEA		####	area ratio

Alter

Eto/(Eto + 5_Eto) ####

Hydrolyse

	Korr.		
D5-Andro/MT	41.84	####	□

Derivatization

Andro/(Andro+Andro-mono) #### □

IS-Areas

Methyltestosterone #### □

Verified: Bacteria: □ Positive: □ Substances:

Data Table Questions

This section answers questions about data tables in TraceFinder custom reports.

- How Do I Create a Data Table in Excel?
- How Do I Add or Insert a Column in a Data Table?
- How Do I Remove a Column from a Data Table?
- How Do I Move a Column in a Data Table?
- How Do I Determine If a Column in a Data Table Is Linked to an XML Source?
- Why Do So Many Data Tables Appear on My Spreadsheet?
- How Do I Remove Data from the Template After Importing It?
- Why Does the Design Interface Layout Change After I Import Data?
- How Do I Use the Same Field More Than Once in a Report?
- How Do I Add an XML Map to the Workbook?
- How Do I Force the Formulas to Reevaluate?
- Why Does My Formula in a Data Table Not Expand?
- Where Can I Find a RawFileName_SampleCentricData.xml File?
- Why is My Data Table Not Editable?

How Do I Create a Data Table in Excel?

A data table (also called a list in earlier Excel versions) is a table of related data. There are several ways to create a data table in the Excel application. For custom reports, you create a data table only by dragging a tree node from the XML source to the Excel spreadsheet. Each node becomes a column in the data table. The data table has two rows: the column header and an empty row beneath it that acts as a data holder. The second row expands when you import data.

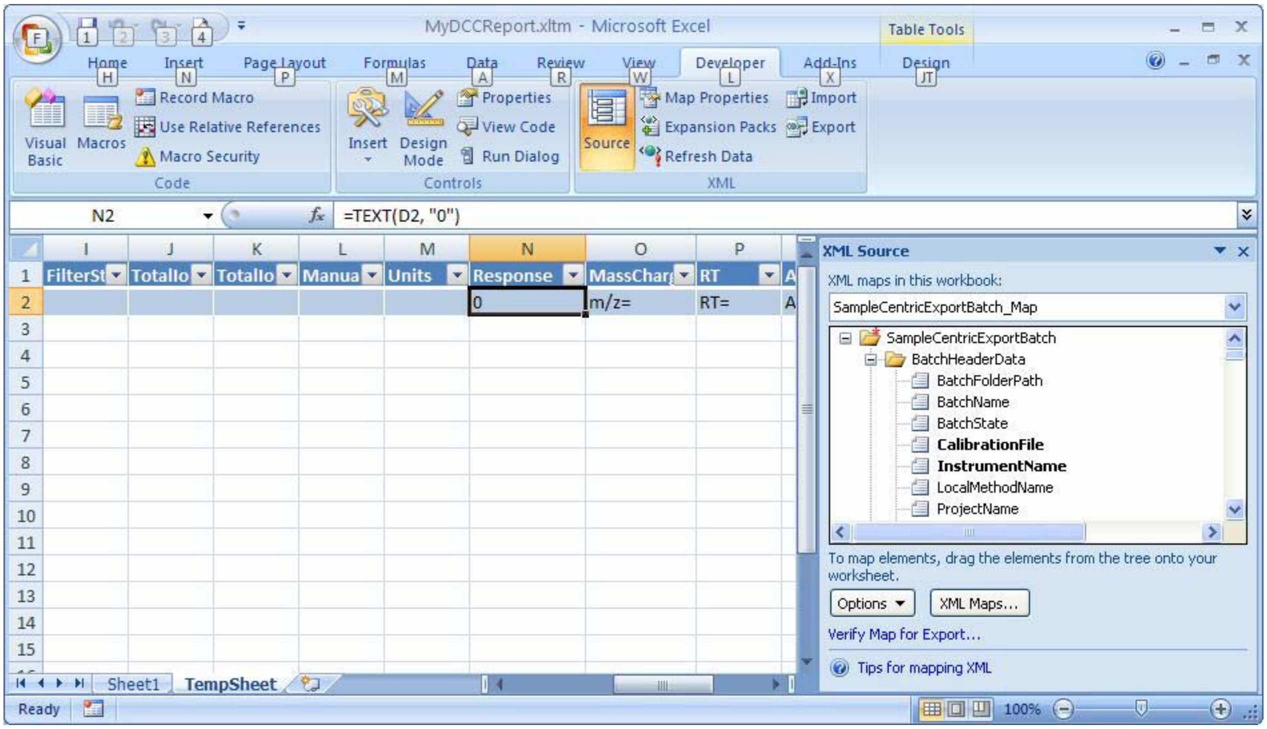
Note The XML source tree contains two kinds of nodes: a single-value node and a collection node. When you place a single-value node in the spreadsheet, it is linked to a single cell, and only the single cell is highlighted. When you place a collection node in the spreadsheet, the Excel application automatically creates a data table with two rows.

A field mapped to a single-value node on the spreadsheet is not visible before data is imported unless the XML source pane is visible. The data table is always visible.

Data tables can have columns that are not linked to the XML source tree. You can use these columns for two purposes in custom reports:

- To reserve space for text if the space in the previous column is insufficient. For example, you can use an unlinked column to accommodate long compound names.
- To create derived columns using Excel formulas for preformatting values based on other columns, enter formulas in the second row. The Excel application automatically expands and evaluates them in all rows after you import data.

Figure 102. Data table before data is imported



N2 =TEXT(D2, "0")

	I	J	K	L	M	N	O	P
1	FilterSt	Totallo	Totallo	Manua	Units	Response	MassChar	RT
2						0	m/z=	RT=

SampleCentricExportBatch_Map

- SampleCentricExportBatch
  - BatchHeaderData
    - BatchFolderPath
    - BatchName
    - BatchState
    - **CalibrationFile**
    - **InstrumentName**
    - LocalMethodName
    - ProjectName

❖ To import data

1. Click the **Developer** tab to display the Developer functions.
2. Click **Import.**

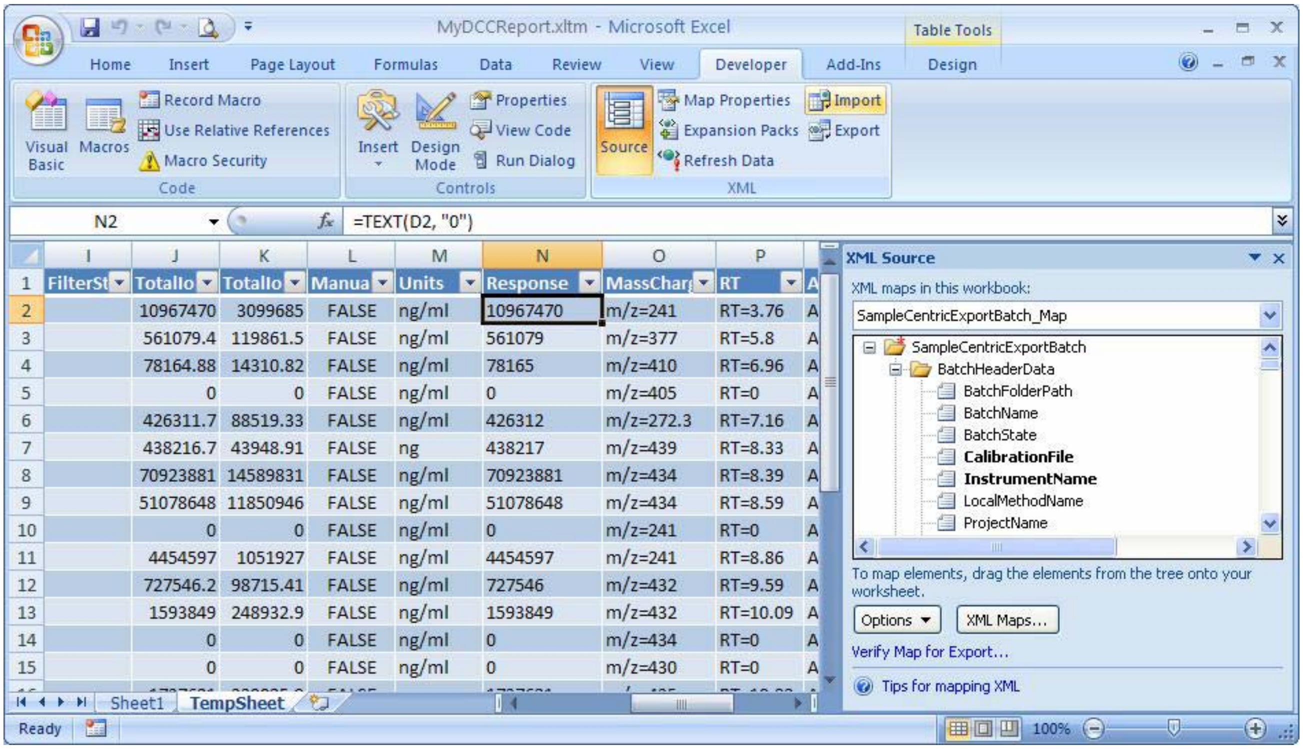
   The Import XML dialog box opens.

   > **IMPORTANT** Because Excel cannot undo the importation of data, always save the template before you import data. To avoid corrupting the template, do not save the template with imported data.

3. Navigate to your .xml file and click **Import**.

   The Excel application populates the spreadsheet with data for you to preview.

Figure 103. Preview of imported data



N2 =TEXT(D2, "0")

	I	J	K	L	M	N	O	P	
1	FilterSt	Totallo	Totallo	Manua	Units	Response	MassChar	RT	A
2		10967470	3099685	FALSE	ng/ml	10967470	m/z=241	RT=3.76	A
3		561079.4	119861.5	FALSE	ng/ml	561079	m/z=377	RT=5.8	A
4		78164.88	14310.82	FALSE	ng/ml	78165	m/z=410	RT=6.96	A
5		0	0	FALSE	ng/ml	0	m/z=405	RT=0	A
6		426311.7	88519.33	FALSE	ng/ml	426312	m/z=272.3	RT=7.16	A
7		438216.7	43948.91	FALSE	ng	438217	m/z=439	RT=8.33	A
8		70923881	14589831	FALSE	ng/ml	70923881	m/z=434	RT=8.39	A
9		51078648	11850946	FALSE	ng/ml	51078648	m/z=434	RT=8.59	A
10		0	0	FALSE	ng/ml	0	m/z=241	RT=0	A
11		4454597	1051927	FALSE	ng/ml	4454597	m/z=241	RT=8.86	A
12		727546.2	98715.41	FALSE	ng/ml	727546	m/z=432	RT=9.59	A
13		1593849	248932.9	FALSE	ng/ml	1593849	m/z=432	RT=10.09	A
14		0	0	FALSE	ng/ml	0	m/z=434	RT=0	A
15		0	0	FALSE	ng/ml	0	m/z=430	RT=0	A

XML Source

XML maps in this workbook:
SampleCentricExportBatch_Map

- SampleCentricExportBatch
  - BatchHeaderData
    - BatchFolderPath
    - BatchName
    - BatchState
    - **CalibrationFile**
    - **InstrumentName**
    - LocalMethodName
    - ProjectName

To map elements, drag the elements from the tree onto your worksheet.

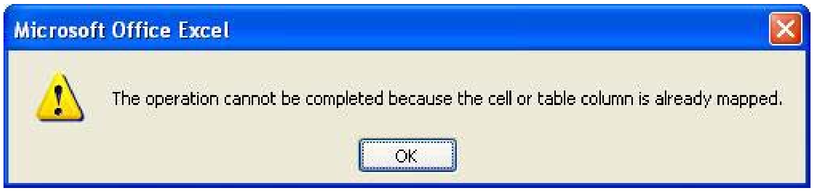
Options ▾ | XML Maps...

Verify Map for Export...

Tips for mapping XML

How Do I Add or Insert a Column in a Data Table?

You can place an XML source node in the spreadsheet only at the beginning or end of an existing Excel data table. You cannot place a source node in the middle of a data table if the column is already mapped to the XML source. If you do, the following error message appears; however, you can place a source node in any unmapped column.



Microsoft Office Excel

The operation cannot be completed because the cell or table column is already mapped.

❖ To place a node in the middle of a data table

1. Do one of the following:

   a. Place the tree node at the beginning or the end of the data table.

   b. Move the column to any place within the data table.

   For more information on moving columns within a data table, see "Creating the Report Headers" on page 94.

   –or–

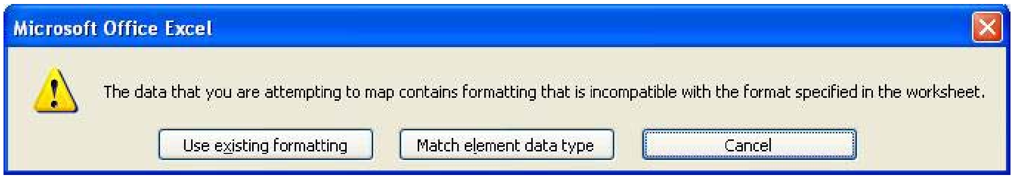
   a. Insert an empty (unmapped) column by right-clicking the column and choosing **Insert > Table Columns to the Left**.

   The new column (usually called Column1) is unmapped.

   b. Drag and drop a tree node to it.

2. If you receive the following error message, click **Match Element Data Type** to dismiss it.

Figure 104. Formatting error message



Microsoft Office Excel

The data that you are attempting to map contains formatting that is incompatible with the format specified in the worksheet.

❖ To verify a mapped node

Do one of the following:

- Select the column.

  If the column is mapped to a tree node, the Excel application selects the mapped node in the XML source tree and displays it in bold font.

  –or–

- Select the mapped node in the XML source tree.

All mapped nodes are displayed in bold font. If the node is mapped to a column, the Excel application automatically selects the column in the data table.

When you insert an empty column and then drag an XML source node to the empty column, the default column header does not change.

For additional instructions, see "Creating the Report Headers" on page 94.

How Do I Remove a Column from a Data Table?

You might need to remove a column in a data table.

- To delete an entire column of the spreadsheet that includes a column in the data table, right-click the column header cell and choose **Delete** from the shortcut menu.

  Note Pressing the Delete key does not delete the column.

- To delete a column in a data table without deleting the columns outside the data table, right-click the column, and choose **Delete > Table Columns**.

- To remove the mapping from the XML source tree, right-click the node shown in bold font and choose **Remove Element**.

  This step removes the link between this node and the data table column, but it does not delete the data table column. The data table column becomes unmapped, but the previous name remains.

For additional instructions, see "Creating the Report Headers" on page 94.

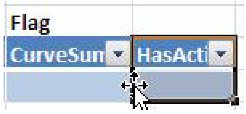
How Do I Move a Column in a Data Table?

Follow this procedure to move a column in a data table.

❖ **To move table columns**

1. Select the column and hold the mouse cursor on the border of the selected cells.
2. When the pointer changes to Resize All, drag the column to its new location.

Figure 105. Moving table columns



For additional instructions, see “Creating the Report Headers” on page 94.

How Do I Determine If a Column in a Data Table Is Linked to an XML Source?

A column in a data table might or might not be mapped to an XML source. To determine if it is mapped, open the XML source pane by choosing **Developer** > **Source** and then click the column to be checked in the spreadsheet. For a mapped column, the Excel application automatically selects the mapped node in the XML source tree. This node automatically appears in the visible area.

All nodes in the XML source window shown in bold font are mapped nodes. The Excel application makes the mapped field or column visible when you click a node appearing in bold font in the XML source tree.

For additional instructions, see “Creating the Report Headers” on page 94.

Why Do So Many Data Tables Appear on My Spreadsheet?

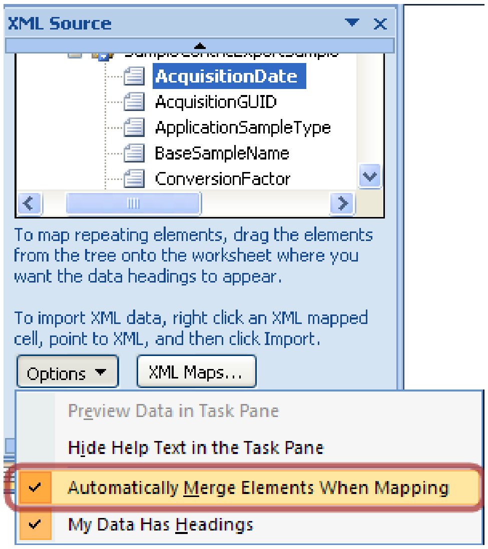
When you drag a node from the XML source tree and place it next to another node, Excel automatically merges them into one data table.

❖ To control the automatic merging of two adjacent nodes

1. Choose **Developer > Source**.

2. Click **Options** and choose **Automatically Merge Elements When Mapping**.

   Figure 106. Automatically merging nodes into one data table



To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped cell, point to XML, and then click Import.

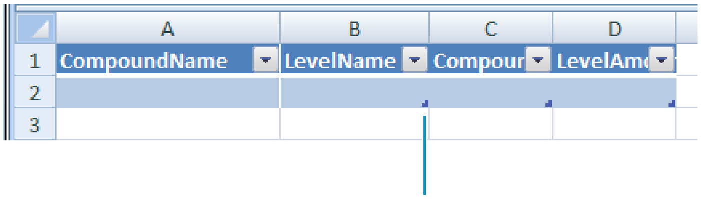
The Automatically Merge Elements When Mapping check box is selected by default. If you do not select this check box, Excel creates a new data table each time that you drop a node from the XML source, so several data tables might appear on your spreadsheet.

Excel also creates a new data table if you place a node in a cell that is not adjacent to an existing data table. The Automatically Merge Elements When Mapping option has no effect on this activity.

If you create data tables separately, you cannot merge them later.

Each data table on the spreadsheet is marked by a small triangle at the bottom right corner, as shown in the following figure. The A and B columns belong to one data table. The C and D columns are separate data tables.

Figure 107. Triangular data table marker

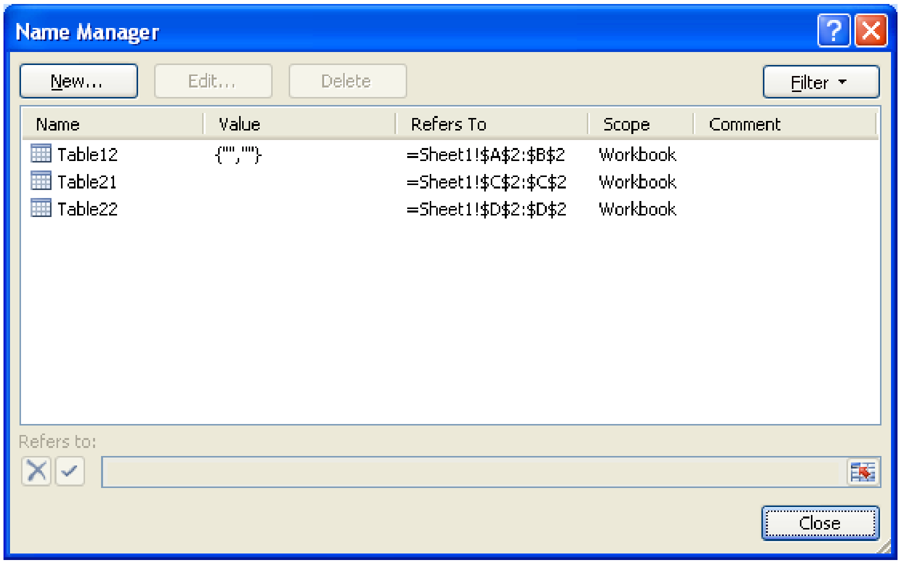


	A	B	C	D
1	CompoundName	LevelName	Compour	LevelAm
2				
3				

Data table marker

Excel automatically assigns a default name like Table1 or Table2 to each data table. If you select Formulas > Name Manager, you can view these names displayed in the Name Manager dialog box, as shown in the following figure:

Figure 108. Data table default names in the Name Manager dialog box



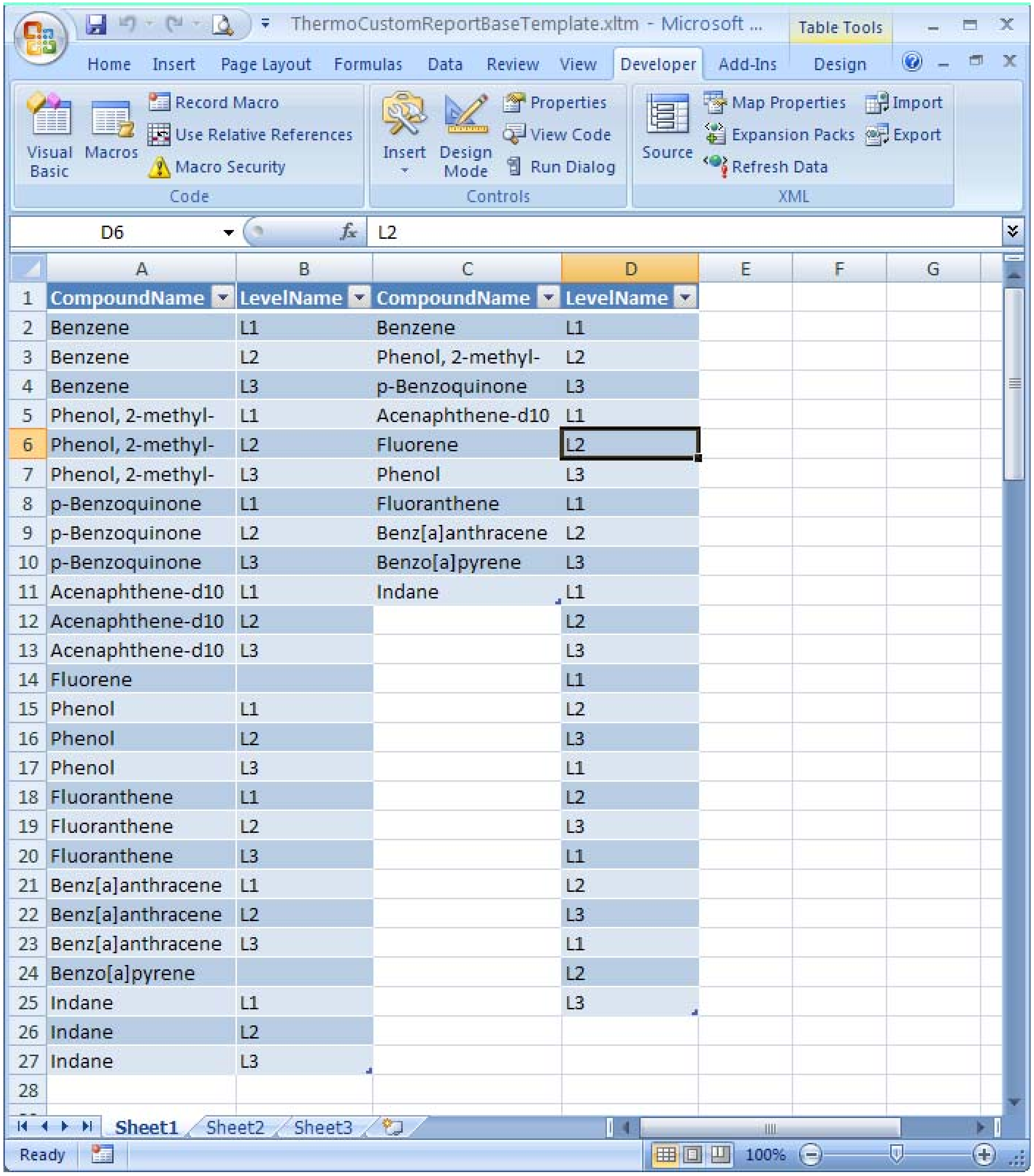
Name	Value	Refers To	Scope	Comment
Table12	{"",""}	=Sheet1!$A$2:$B$2	Workbook	
Table21		=Sheet1!$C$2:$C$2	Workbook	
Table22		=Sheet1!$D$2:$D$2	Workbook	

Thermo Fisher Scientific recommends that you rename the automatically assigned data table names with distinctive names that you can refer to in formulas or in the VBA code.

You might want to leave some space between two columns. For example, you might want to leave enough space to accommodate long compound names. Do not create separate tables to do this, although it might work in some cases. Instead, create an empty column. For instructions, see "Adding Empty Columns" on page 37.

The effect of separate data tables versus one data table might not become obvious until you place nodes from different levels in the XML source tree or import source data into the data tables. In the following figure, the A and B columns belong to one data table. The C and D columns are separate data tables.

Figure 109. One data table (columns A and B) with two separate data tables (columns C and D)



	A	B	C	D	E	F	G
1	CompoundName	LevelName	CompoundName	LevelName			
2	Benzene	L1	Benzene	L1			
3	Benzene	L2	Phenol, 2-methyl-	L2			
4	Benzene	L3	p-Benzoquinone	L3			
5	Phenol, 2-methyl-	L1	Acenaphthene-d10	L1			
6	Phenol, 2-methyl-	L2	Fluorene	L2			
7	Phenol, 2-methyl-	L3	Phenol	L3			
8	p-Benzoquinone	L1	Fluoranthene	L1			
9	p-Benzoquinone	L2	Benz[a]anthracene	L2			
10	p-Benzoquinone	L3	Benzo[a]pyrene	L3			
11	Acenaphthene-d10	L1	Indane	L1			
12	Acenaphthene-d10	L2		L2			
13	Acenaphthene-d10	L3		L3			
14	Fluorene			L1			
15	Phenol	L1		L2			
16	Phenol	L2		L3			
17	Phenol	L3		L1			
18	Fluoranthene	L1		L2			
19	Fluoranthene	L2		L3			
20	Fluoranthene	L3		L1			
21	Benz[a]anthracene	L1		L2			
22	Benz[a]anthracene	L2		L3			
23	Benz[a]anthracene	L3		L1			
24	Benzo[a]pyrene			L2			
25	Indane	L1		L3			
26	Indane	L2					
27	Indane	L3					
28							

"LevelName" is a sub-collection of a compound. In columns B and D, one compound has three calibration levels, except for fluorene and benzo[a]pyrene, which are internal standard compounds and are not associated with any calibration level.

The third and fourth columns are directly mapped to the XML source. However, in the first data table, each "CompoundName" is repeated for every "LevelName," and each "LevelName" is repeated for every "CompoundName."

A target compound—for example, benzene—can have multiple calibration levels, but internal standard compounds—for example, fluorene—do not have any. When you map the “LevelName” in a separate data table like column D, internal standard compounds like fluorene do not have any entries. However, if you map “CompoundName” and “LevelName” together in one data table like columns A and B, fluorene must be listed, although its corresponding “LevelName” is empty.

Recognizing this difference between one data table and separate data tables is very important when you create a new report because it helps you decide when to use a separate data table and when to avoid using one. You can find an example showing how to use these two table types together in the delivered High-Density Sample Report Long 2 template.

How Do I Remove Data from the Template After Importing It?

Importing data is one of the actions in Excel that you cannot undo. Thermo Fisher Scientific recommends that you save the template before importing data. Do not save the template after importing data.

Note Leaving imported data in the template might confuse other template developers, but it does not cause any harm to generated reports because the data is overwritten at run time.

If you saved the template with the imported data, you can remove the data from the template by following these procedures:

- For single-value cells, select the cell and press the **Delete** key to remove the text. The mapping of the cell is not removed.

- For data tables, select the entire table except the header row, right-click and select **Delete > Table Rows**. You must exclude the header rows from selection to avoid deleting the entire data table. Pressing the Delete key only removes the text from the template; it does not reduce the data table to two rows.

Why Does the Design Interface Layout Change After I Import Data?

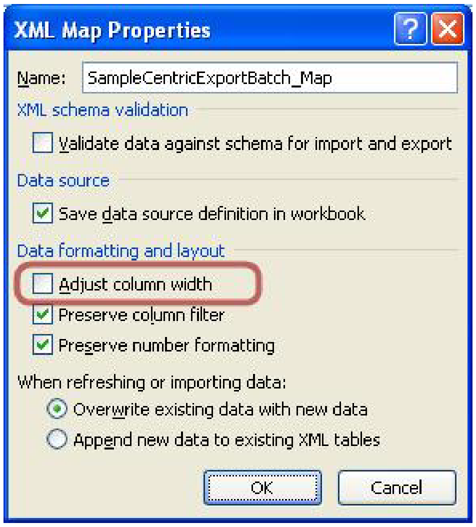
If you find that the design interface layout changes after you import data, the likely cause is the Adjust Column Width property defaulting to True.

❖ To turn off the Adjust Column Width property

1. Click the **Developer** tab in the Excel spreadsheet.
2. Select the related map.
3. Click **Map Properties**.

   The XML Map Properties dialog box opens.

   Figure 110. XML Map Properties dialog box



XML Map Properties

Name: SampleCentricExportBatch_Map

XML schema validation

□ Validate data against schema for import and export

Data source

☑ Save data source definition in workbook

Data formatting and layout

□ Adjust column width

☑ Preserve column filter

☑ Preserve number formatting

When refreshing or importing data:

● Overwrite existing data with new data

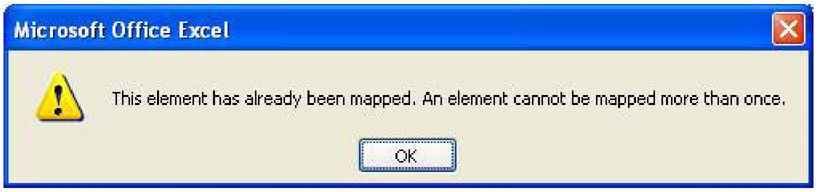
○ Append new data to existing XML tables

4. Clear the **Adjust Column Width** check box.

   When you select this check box, the cell expands to fit the data when you import data, and the interface layout changes as a result. This check box is selected by default. However, Thermo Fisher Scientific recommends that you clear this check box for all XML maps to preserve layout design.

How Do I Use the Same Field More Than Once in a Report?

You can reference a tree node in the XML source only once. If you try to drag an already mapped tree node to the spreadsheet, the following error message appears.

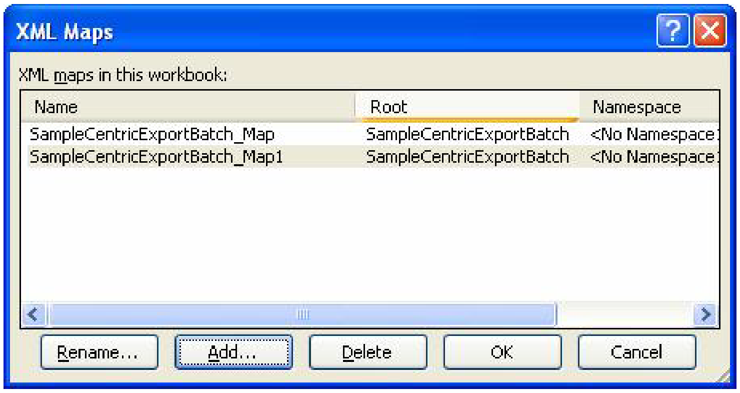


Microsoft Office Excel

This element has already been mapped. An element cannot be mapped more than once.

For custom reports, always use a separate XML map when you need a field more than once. You can add XML maps to the same file as many times as necessary.

Figure 111. Multiple mappings added to the same file



XML Maps

XML maps in this workbook:

Name	Root	Namespace
SampleCentricExportBatch_Map	SampleCentricExportBatch	<No Namespace>
SampleCentricExportBatch_Map1	SampleCentricExportBatch	<No Namespace>

Because you have only a single .xml file to import, the add-in component populates all maps when you import the .xml file. You must keep the default names SampleCentricExportBatch_Map, SampleCentricExportBatch_Map1, and so on.

How Do I Add an XML Map to the Workbook?

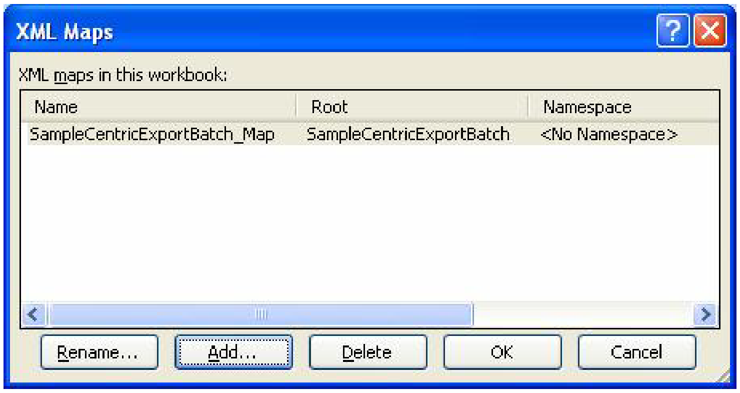
You can add an XML map to the workbook.

❖ To add an XML map to the workbook

1. In the Excel spreadsheet, click the **Developer** tab to display the Developer features.
2. Click the **Source** button, Source.
3. Click the **XML Maps** button.

   The XML Maps dialog box opens. It lists all the existing maps.

Figure 112. XML Maps dialog box



Name	Root	Namespace
SampleCentricExportBatch_Map	SampleCentricExportBatch	<No Namespace>

You can use the XML Maps dialog box to add a new map, rename a map, or delete a map.

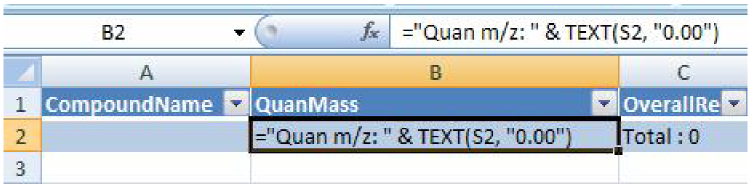
How Do I Force the Formulas to Reevaluate?

Sometimes you might want the Excel application to reevaluate formulas on the sheet, for example, before copying. You can do this by calling the Application.Calculate function in the VBA code.

Why Does My Formula in a Data Table Not Expand?

When you add a new column to a data table, the cell format defaults to text. In text format, the formula that you entered into the cell appears as text and does not expand when you import data into the data table. An equals sign (=) in a cell distinguishes text format in the cell from evaluated results, as shown in the B2 cell in the following figure.

Figure 113. Cell in text format marked by an equals sign



B2 ="Quan m/z: " & TEXT(S2, "0.00")

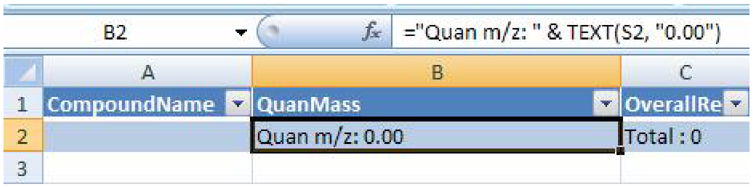
	A	B	C
1	CompoundName	QuanMass	OverallRe
2		="Quan m/z: " & TEXT(S2, "0.00")	Total : 0
3			

❖ To expand a formula in a data table

1. Manually change the cell format to another format, such as the general format shown in the following figure.

2. To update the cell format throughout the data table, click each cell and press ENTER.

Figure 114. Cell in general format marked by the absence of an equals sign



B2 ="Quan m/z: " & TEXT(S2, "0.00")

	A	B	C
1	CompoundName	QuanMass	OverallRe
2		Quan m/z: 0.00	Total : 0
3			

Where Can I Find a *RawFileName*_SampleCentricData.xml File?

The TraceFinder application generates the *RawFileName*_SampleCentricData.xml file when it checks any custom report as part of submitting the batch or sample for reporting. This file contains all sample-specific data and is the only file used by custom reports. The file is normally located in the Data folder of the batch:

```
C:\Thermo\TraceFinder\1.1\Projects\PROJECTNAME\SUBPROJECTNAME\
BATCHNAME\Data
```

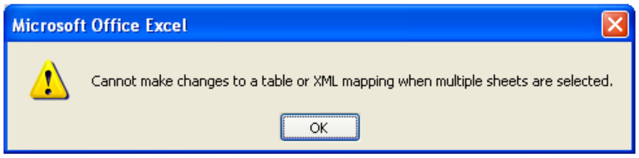
Example file location:

```
C:\Thermo\TraceFinder\1.1\Projects\ProjectA\SubprojectA\
ML_BATCH_01\Data
```

If you use the Run box on the Start menu to test your template, you can place this file anywhere on your local drive as long as you pass the correct path to the template in the Run box.

Why is My Data Table Not Editable?

If you try to edit data tables in the generated report, you might receive the following error message from Excel.

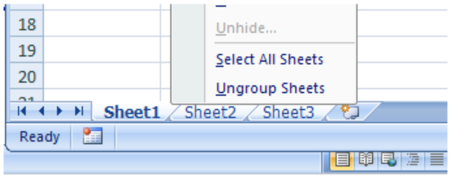


Cannot make changes to a table or XML mapping when multiple sheets are selected.

This error message results from grouping the sheets. Grouped sheets are selected sheets in the Excel spreadsheet. If a generated report contains multiple sheets (tabs), the add-in component automatically groups them all together to make sure that the pages in all sheets are sequentially numbered. Without this grouping, pages in each sheet start from 1, which is normally undesirable.

You can break this grouping by clicking any of the inactive sheets. You can also break the grouping by right-clicking any tab and selecting **Ungroup Sheets** from the menu. To group them again, right click the tab and select **Select All Sheets** from the menu.

Figure 115. Ungrouping multiple sheets



If you break the grouping manually, be sure to restore it after editing and before printing to ensure that the page numbering is correct.

Image Handling Questions

The questions in this section pertain to the handling of images in TraceFinder custom reports.

- How Do I Add an Image to a Report?
- How Do I Remove an Image from a Report?

How Do I Add an Image to a Report?

The TraceFinder application exports images as byte arrays embedded in the *RawFileName*_SampleCentricData.xml file. However, the Excel spreadsheet cannot directly convert byte arrays into an image. Byte arrays for an image are normally too large to be placed directly into a cell.

As a workaround to this limitation, when you drag and drop an image node onto the spreadsheet, the add-in component captures this event and instructs you to create a named range that acts as a placeholder for the image that you drag, as shown in Figure 116.

The image is loaded to this placeholder at run time through a special add-in function called CreateImage in the VBA code. For information on this function, see "Public Function CreateImage" on page 204.

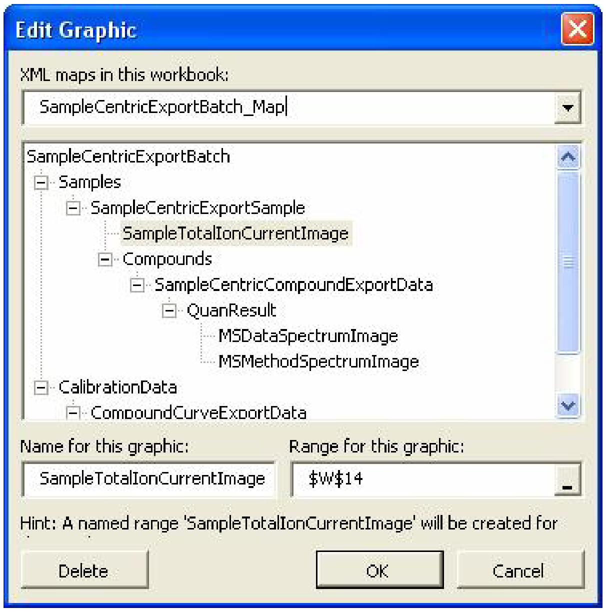
Note All image nodes in the exported XML file have the "Image" suffix appended to their names, for example, SampleTotalIonCurrentImage for the TIC image of the sample. This naming convention enables the add-in component to capture drag-and-drop events as images so that the application can handle them correctly.

❖ To add an image to a report

1. Drag an image node from the XML source tree to your spreadsheet.

   The Edit Graphic dialog box opens.

Figure 116. Edit Graphic dialog box



XML maps in this workbook: SampleCentricExportBatch_Map

Name for this graphic: SampleTotalIonCurrentImage

Range for this graphic: $W$14

Hint: A named range 'SampleTotalIonCurrentImage' will be created for

The tree shown in this dialog box only displays the image nodes with the Image suffix appended to their names. The currently selected tree node is automatically highlighted.

2. Specify the name and cell range for the graphic in the Edit Graphic dialog box.

Tip You can enter the range directly in the Range for This Graphic box in the Edit Graphic dialog box, or select the cells from the list on the right.

Tip You do not have to create named range placeholders for images, because the VBA code can reference the cells directly. However, creating named ranges for images makes later maintenance easier.

For the syntax of the CreateImage function, see "Public Function CreateImage" on page 204.

How Do I Remove an Image from a Report?

Because images are loaded through the VBA code, you must remove the corresponding VBA code that the template is based on if you want to remove an image. When removing an image, you might also remove the cells reserved for the image on the spreadsheet by deleting or hiding them. You can find examples of this usage in the delivered Quantitation report, where the visibility of the TIC image is controlled by the value of the SampleCentricExportBatch/MethodHeaderData/ShowChromatogramOnQuanReport node.

Debugging Questions

This section provides information on debugging VBA code.

- How Do I Debug the VBA Code That the Template Is Based On?
- How Do I Debug the Add-in Code?

How Do I Debug the VBA Code That the Template Is Based On?

You can use several methods to debug the VBA code that the template is based on, but you must first stop running the template. The template runs by default but stops if any variable is undefined, assuming that the code module includes the Option Explicit statement at the beginning.

❖ To debug the VBA code

1. Estimate where the problem might be in the code.
2. Change any variable in that place to an undefined variable.

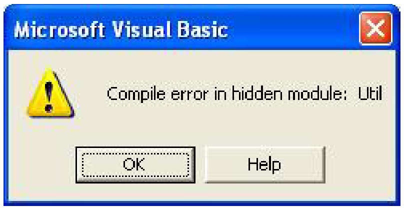
   You will receive a compiler error when you try to compile the code, but you can ignore this error for debugging purposes.
3. Run the template.

   The compiler stops at the beginning of the function where the undefined variable is located.
4. Change the variable back, set a break point in the code, and press the F5 key to run the program.

   The program stops at the first break point that you set in this function. Once it stops, you can step through the code as usual. You can also switch to the spreadsheet any time to determine how the code has worked so far.

How Do I Debug the Add-in Code?

The add-in code is a regular VBA project. You can use the same procedure described in "How Do I Debug the VBA Code That the Template Is Based On?" on page 197 to debug the add-in code. The only difference is that the add-in project is protected by a password. Any error in a password-protected module generates a generic Microsoft Visual Basic error message.



Compile error in hidden module: Util

To debug the error, unlock the add-in project with a valid password. After the project is unlocked, you can debug it as you would any other VBA code.

Custom Report Add-in API

The custom report add-in application programming interface (API), an Excel add-in component, provides a framework for customizing reports by using the Excel template for applications.

The add-in file is called Thermo Report Add-In.xla, which is usually installed in the C:\Program Files\Microsoft Office\Office12\Library folder. This add-in component is normally associated with a template, and it is loaded only when you open or use the associated template.

The add-in component performs two main tasks:

- Automates the custom report generation.
- Provides some helper functions (APIs) for template developers to use.

Chapter 2, "Overview," discusses the automatic generation of custom reports. This appendix describes all public APIs that you can use in the template VBA code. You can generate most custom reports without changing the add-in code.

Contents

Contents, Cont'd

Contents, Cont'd

All API functions documented in this appendix reside in the Util class. A public instance (mUtil) is created in Module1 of the add-in so that all templates referring to the add-in component can use it directly, for example, mUtil.ShiftRange(…).

Note If you have access to the add-in source code, you might see more APIs than those described in this appendix. Some of them were created for an earlier version of the Excel application and are no longer needed but are retained for backward compatibility.

Tip The add-in VBA project (ThermoReportAddIn) is password-protected. However, if you must access the add-in source code, consult your Thermo Fisher Scientific representative for the password.

Public Sub SetupPageHeader

```
Public Sub SetupPageHeader(Optional ByVal withPrintTimestamp As Boolean =
True)
```

Purpose

This function provides the default page header commonly used in reports.

It is normally called when the IUtil interface is implemented. If you do not need the default header, remove this call from IUtil_SetupPageHeader() in the MyUtil class.

Parameters

Table 2 describes the parameters that you must provide values for in the Public Sub SetupPageHeader syntax.

Table 2. Public Sub SetupPageHeader parameters

Parameter	Description
withPrintTimestamp	Specifies whether the page header includes a time stamp as the second line in the middle when you print reports. • True: Includes the time stamp in the header. • False: Does not include the time stamp in the header. The default is True. This parameter is optional.

Public Sub SetupPageFooter

```
Public Sub SetupPageFooter(text As String, Optional bottomMarginInInch As
Double = 0.75, Optional picFileName As String)
```

Purpose

This function provides the default page footer commonly used in reports.

This function is normally called when the IUtil interface is implemented. If you do not need the default footer, remove this call from IUtil_SetupPageFooter() in the MyUtil class.

Parameters

Table 3 describes the parameters that you must provide values for in the Public Sub SetupPageFooter syntax.

Table 3. Public Sub SetupPageFooter parameters

Parameter	Description
text	Specifies the footer text.
bottomMarginInInch	Specifies the page margin at the bottom of the page, in inches. The default is 0.75. This parameter is optional.
picFileName	Specifies the image file (full path) to be shown. This parameter has no default. This parameter is optional. Only one image can appear in the footer. The text must contain “&G” to indicate in the Excel application where to show the image. Without “&G” in the text, no image is shown. If you do not provide an image file, the square.jpg image appears in the C:\Thermo\Shared\Templates\Reports report folder. It is used to indicate manual integration in the legend.

Public Function CreateImage

```
Public Function CreateImage(ByVal target, ByVal imagePath As String,
Optional ByVal cropTop As Long = 1, Optional ByVal cropLeft As Long = 1,
Optional ByVal cropRight As Long = 1, Optional ByVal cropBottom As Long =
1) As String
```

Purpose

This function creates an image based on the byte array at *imagePath* from the XML file and specifies its target location.

Parameters

Table 4 describes the parameters that you must provide values for in the Public Function CreateImage syntax.

Table 4. Public Function CreateImage parameters (Sheet 1 of 2)

Parameter	Description
target	Specifies a range of cells either by a named range (for example, QuanChart) or by a cell range (for example, Sheet1!A1:E5) to indicate where to set the image.
imagePath	Specifies a full XPath to the image node in the XML source tree. Here is an example: `SampleCentricExportBatch/Samples/ SampleCentricExportSample/Compounds/ SampleCentricCompoundExportData [CompoundKey='" & compId & "']/ QuanResult/QuanPeakResults/ CompoundQuanPeakResultExportData/ PeakImage.`
cropTop	Specifies how much to crop at the top of the image. The default is 1. This parameter is optional.
cropLeft	Specifies how much to crop to the left of the image. The default is 1. This parameter is optional.

Table 4. Public Function CreateImage parameters (Sheet 2 of 2)

Parameter	Description
cropRight	Specifies how much to crop to the right of the image. The default is 1. This parameter is optional.
cropBottom	Specifies how much to crop at the bottom of the image. The default is 1. This parameter is optional.

Example

Here is an example taken from a DCC report:

```
mUtil.CreateImage mUtil.ShiftRange("QuanChart", cc, rr, cc, rr),
"SampleCentricExportBatch/Samples/SampleCentricExportSample/Compounds
/SampleCentricCompoundExportData[CompoundKey='" & compId & "']
/QuanResult/QuanPeakResults/CompoundQuanPeakResultExportData/
PeakImage", 20, 5, 12, 2
```

In this example:

- The *target* parameter is a shifted named range.
- The *imagePath* is an XPath with an expression that specifies the compound identification.
- All margins surrounding the image are cropped: 20 on the top, 5 on the left, 12 on the right, and 2 on the bottom.

Public Sub CopyPaste

```
Public Sub CopyPaste(ByVal source, ByVal target, Optional ByVal
inclRowHeight As Boolean = True, Optional ByVal valueOnly = False,
Optional ByVal copyFormula = True)
```

Purpose

This function replaces Excel's Selection.Copy function, primarily to avoid using the Clipboard. The reporting process normally runs in the background and might take some time. When you use the Clipboard to move data on the spreadsheet, you might inadvertently cause an incorrect report to be generated.

This function is a little slower than Excel's Selection.Copy function, especially when you copy a large amount of data. If you prefer to use the Clipboard and you are certain that no one is working on the computer while the application is generating reports in the background, you can switch to Excel's Selection.Copy function by setting the UseClipboard global parameter to True in the Util class. You can also switch by using the /b command-line option.

Parameters

Table 5 describes the parameters that you must provide values for in the Public Sub CopyPaste syntax.

Table 5. Public Sub CopyPaste parameters (Sheet 1 of 2)

Parameter	Description
source	Specifies a range of cells by a named range (for example, DataTable) or by a cell range (for example, TempSheet!A1:E5) to indicate the location of the source to copy.
target	Specifies a range of cells to indicate the location to copy to. You can specify this range by a named range (for example, ChartLegend) or by a cell range (for example, TempSheet!A1:E5). The target cell range does not have to match the source cell range. If it is smaller than the source cell range, it expands to all the source cells copied. It could be just the first cell of the paste target.
inclRowHeight	Specifies whether the row height should be copied to the target. • True: Copies the row height to the target. • False: Does not copy the row height to the target. The default is True. This parameter is optional.

Table 5. Public Sub CopyPaste parameters (Sheet 2 of 2)

Parameter	Description
valueOnly	Determines whether the formatting and formulas in the cells are copied along with the values. • True: Copies only the values of the cells. • False: Copies the formatting and formulas in addition to the values of the cells. The default is False. This parameter is optional.
copyFormula	• Specifies whether formulas are copied. • True: Copies formulas. • False: Does not copy formulas. The default is True. This parameter is optional. It takes effect only if you set the *valueOnly* parameter to False.

Example

The following example copies the M2 through M29 cells on the tempSheet sheet to Sheet1, starting from the H51 cell. It does not include row height adjustment and copies only text values (no formatting or formulas).

```
mUtil.CopyPaste "tempSheet!M2:M29", "Sheet1!H51", False, True
```

Public Function IsXPath

```
Public Function IsXPath(ByVal rng As Range) As Boolean
```

Purpose

This function determines whether the specific cell or cell range contains a valid XPath. The only reason to include this function is to prevent the Excel application from throwing an exception when the value is accessed if it is not a valid XPath.

Parameters

Table 6 describes the parameters that you must provide values for in the Public Function IsXPath syntax.

Table 6. Public Function IsXPath parameters

Parameter	Description
rng	Specifies a range of cells by a named range (for example, DataTable), by a cell range (for example, TempSheet!A1:E5), or by a cell. • True: The cell or cell range contains a valid XPath. • False: The cell or cell range does not contain a valid XPath.

Public Function HiddenRowCount

```
Public Function HiddenRowCount(ByVal Range1) As Long
```

Purpose

This function returns the count of hidden rows in the specified cell range. It is primarily used to calculate copy locations. For example, it is used in quantitation reports.

Parameters

Table 7 describes the parameters that you must provide values for in the Public Function HiddenRowCount syntax.

Table 7. Public Function HiddenRowCount parameters

Parameter	Description
`Range1`	Specifies a range of cells by a named range (for example, DataTable), by a cell range (for example, TempSheet!A1:E5), or by a cell.

Public Function VisibleRowCount

```
Public Function VisibleRowCount(ByVal Range1) As Long
```

Purpose

This function returns the count of visible rows in the specified cell range. It is primarily used to calculate copy locations. For example, it is used in quantitation reports.

Parameters

Table 8 describes the parameters that you must provide values for in the Public Function VisibleRowCount syntax.

Table 8. Public Function VisibleRowCount parameters

Parameter	Description
Range1	Specifies a range of cells by a named range (for example, DataTable), by a cell range (for example, TempSheet!A1:E5), or by a cell.

Public Function Contains

```
Public Function Contains(ByVal Range1, ByVal Range2) As Boolean
```

Purpose

This function determines whether a range contains another range. This function is primarily used in the Validate() function to check the named ranges.

Parameters

Table 9 describes the parameters that you must provide values for in the Public Function Contains syntax.

Table 9. Public Function Contains parameters

Parameter	Description
Range1	Specifies a range of cells by a named range (for example, DataTable), by a cell range (for example, TempSheet!A1:E5), or by a cell. • True: The range contains another range. • False: The range does not contain another range.
Range2	Specifies a range of cells either by a named range (for example, DataTable), by a cell range (for example,TempSheet!A1:E5), or by a cell. • True: The range contains another range. • False: The range does not contain another range.

Public Sub ShowErrMsg

```
Public Sub ShowErrMsg(ByVal err_msg As String, ByVal warn_msg As String,
Optional ByVal showWarning As Boolean = True)
```

Purpose

This function displays a dialog box with a specified message. It is primarily used in the Validate() function to check the named ranges.

Parameters

Table 10 describes the parameters that you must provide values for in the Public Sub ShowErrMsg syntax.

Table 10. Public Sub ShowErrMsg parameters

Parameter	Description
err_msg	Specifies the error message text to display.
warn_msg	Specifies the warning message text to display.
showWarning	Specifies whether the warning message is shown if *err_msg* is not defined. If *err_msg* is defined, the message box always appears. • True: Displays the warning message if *err_msg* is not defined. • False: Does not display the warning message if *err_msg* is not defined. The default is True. This parameter is optional.

Public Sub CheckRequiredName

```
Public Sub CheckRequiredName(err_msg As String, ByVal reqName As String,
Optional ByVal msg As String)
```

Purpose

This function determines whether the specified required name (*reqName*) is defined as a named range. If it is not, the function appends the message specified by *msg* to the error message (*err_msg*) to act as the error message. If *reqName* is defined, this function does not alter the original error message. This function is primarily used in the Validate() function to check the named ranges.

Parameters

Table 11 describes the parameter that you must provide values for in the Public Sub CheckRequiredName syntax.

Table 11. Public Sub CheckRequired Name parameters

Parameter	Description
err_msg	Specifies the error message text to display.
reqName	Specifies the name of the required named range.
msg	Specifies the message text to use in a default message if *reqName* is not defined as a named range. This parameter is optional.

Public Sub CheckRequiredNameInsideOf

```
Public Sub CheckRequiredNameInsideOf(err_msg As String, ByVal reqName As
String, ByVal containerName As String, Optional ByVal msg As String)
```

Purpose

This function checks whether the required name specified by *reqName* is defined as a named range and is within the named range specified by *containerName*. If it is not, the function appends the message specified by *msg* to the error message specified by *err_msg* to act as the error message. If the required name is defined, this function does not alter the original error message. This function is primarily used in the Validate() function to check the named ranges.

Parameters

Table 12 describes the parameters that you must provide values for in the Public Sub CheckRequiredNameInsideOf syntax.

Table 12. Public Sub CheckRequiredNameInsideOf parameters

Parameter	Description
err_msg	Specifies the error message text to display.
reqName	Specifies the name of the required named range.
containerName	Specifies the named range of the container cells.
msg	Specifies the message text to use if the required name specified by *reqName* is not defined as a named range or is defined outside the range specified by *containerName*. If *msg* is missing and *reqName* is undefined, the function generates a default message. This parameter is optional.

Public Sub CheckRequiredNameOutsideOf

```
Public Sub CheckRequiredNameOutsideOf(err_msg As String, ByVal reqName As
String, ByVal containerName As String, Optional ByVal msg As String)
```

Purpose

This function determines whether the required name specified by *reqName* is defined as a named range and is outside the named range specified by *containerName*. If it is not, the function appends the message specified by *msg* to the error message specified by *err_msg* to act as the error message. If the required name is defined, this function does not alter the original error message. This function is primarily used in the Validate() function to check the named ranges.

Parameters

Table 13 describes the parameters that you must provide values for in the Public Sub CheckRequiredNameOutsideOf syntax.

Table 13. Public Sub CheckRequiredNameOutsideOf parameters

Parameters	Description
err_msg	Specifies the error message text to display.
reqName	Specifies the name of the required named range.
containerName	Specifies the named range of the container cells.
msg	Specifies the message text to use if the required name specified by *reqName* is not defined as a named range or is defined but is outside the range specified by *containerName*. If *msg* is missing and *reqName* is undefined, the function generates a default message. This parameter is optional.

Public Sub CheckOptionalName

```
Public Sub CheckOptionalName(info_msg As String, ByVal optName As String,
Optional ByVal msg As String)
```

Purpose

This function determines whether the specified optional name specified by *optName* is defined as a named range. If it is not, it appends a message specified by *msg* to the informational message specified by *info_msg* to act as the informational message. If *optName* is defined, this function does not alter the original informational message. This function is primarily used in the Validate() function to check the named ranges.

Parameters

Table 14 describes the parameters that you must provide values for in the Public Sub CheckOptionalName syntax.

Table 14. Public Sub CheckOptionalName parameters

Parameter	Description
info_msg	Specifies the informational message text to display.
optName	Specifies the name of the optional named range.
msg	Specifies the message text to use if the optional name (*optName*) is not defined as a named range. If *msg* is missing and *optName* is undefined, the function generates a default message. This parameter is optional.

Public Sub AddTextAsWatermark

```
Public Sub AddTextAsWatermark(text As String, rng As Range, Optional size
As Double = 40#)
```

Purpose

This function adds a textual watermark to each printed page of the report.

Parameters

Table 15 describes the parameters that you must provide values for in the Public Sub AddTextAsWatermark syntax.

Table 15. Public Sub AddTextAsWatermark parameters

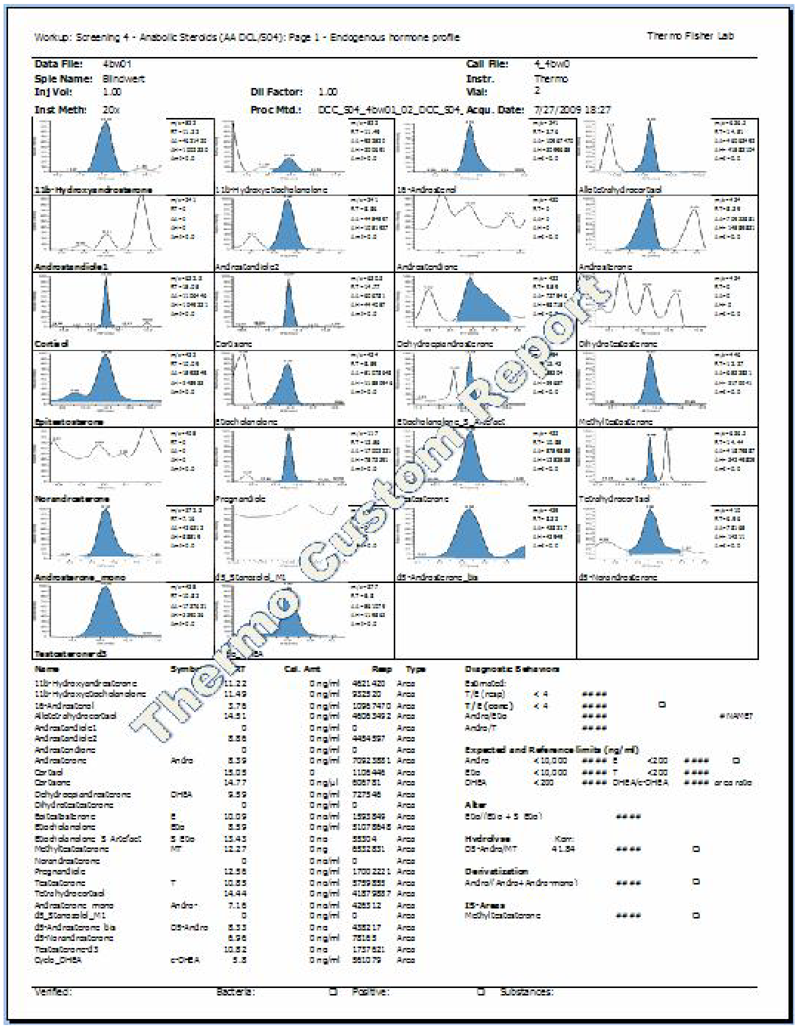
Parameter	Description
text	Specifies the text to display as the watermark.
rng	Specifies the range of cells used to display the watermark.
size	Specifies the font size of the watermark text. The default is 40. This parameter is optional.

Example

Adding the following line to the MyDCCReport template at the end of the ProcessSheet1Data() function generates the watermark shown in Figure 117.

```
mUtil.AddTextAsWatermark "Thermo Custom Report", Range("Sheet1!A1:T78")
```

Figure 117. Watermark on printed page



Workup: Screening 4 - Anabolic Steroids (AA DCL/S04): Page 1 - Endogenous hormone profile

Thermo Fisher Lab

Data File: 4bw01
Sple Name: Blindwert
Inj Vol: 1.00
Inst Meth: 20x
Dil Factor: 1.00
Proc Mtd.: DCC_S04_4bw01_02_DCC_S04_
Cali File: 4_4bw0
Instr. Thermo
Vial: 2
Acqu. Date: 7/27/2009 18:27

Name	Symbol	RT	Cal. Amt	Resp	Type
11b-Hydroxyandrosterone		11.22	0 ng/ml	4621420	Area
11b-Hydroxyetiocholanolone		11.49	0 ng/ml	952520	Area
16-Androstenol		3.78	0 ng/ml	10987470	Area
Allotetrahydrocortisol		14.51	0 ng/ml	4808349.2	Area
Androstendiol1		0	0 ng/ml	0	Area
Androstendiol2		8.88	0 ng/ml	4484597	Area
Androstendione		0	0 ng/ml	0	Area
Androsterone	Andro	8.39	0 ng/ml	70923881	Area
Cortisol		15.05	0	1108446	Area
Cortisone		14.77	0 ng/µl	808781	Area
Dehydroepiandrosterone	DHEA	9.59	0 ng/ml	727548	Area
Dihydrotestosterone		0	0 ng/ml	0	Area
Epitestosterone	E	10.09	0 ng/ml	1593849	Area
Etiocholanolone	Eto	8.59	0 ng/ml	51078848	Area
Etiocholanolone 5 Artefact	5 Eto	13.43	0 ng	55304	Area
Methyltestosterone	MT	12.27	0 ng	8532831	Area
Norandrosterone		0	0 ng/ml	0	Area
Pregnandiol		12.58	0 ng/ml	17002221	Area
Testosterone	T	10.85	0 ng/ml	5759855	Area
Tetrahydrocortisol		14.44	0 ng/ml	41879887	Area
Androsterone mono	Andro-	7.18	0 ng/ml	428512	Area
d3_Stanozolol_M1		0	0 ng/ml	0	Area
d5-Androsterone bis	D5-Andro	8.33	0 ng	438217	Area
d3-Norandrosterone		8.96	0 ng/ml	78185	Area
Testosterone d3		10.82	0 ng	1737821	Area
Cyclo_DHEA	c-DHEA	5.8	0 ng/ml	581079	Area

Diagnostic Behaviors

Estimated:

T/E (resp)	< 4	####				
T/E (conc)	< 4	####			□	
Andro/Eto		####				#NAME?
Andro/T		####				

Expected and Reference limits (ng/ml)

Andro	< 10,000	####	E	<200	####	□
Eto	< 10,000	####	T	<200	####	
DHEA	< 200	####	DHEA/c-DHEA		####	area ratio

Alter

Eto/(Eto + 5 Eto) ####

Hydrolyse	Korr.		
D5-Andro/MT	41.84	####	□

Derivatization

Andro/(Andro+Andro-mono) #### □

IS-Areas

Methyltestosterone #### □

Verified: Bacteria: □ Positive: □ Substances:

Public Sub AddNamedShapeAsWatermark

```
Public Sub AddNamedShapeAsWatermark(name As String, rng As Range)
```

Purpose

This function adds the named shape (normally an image) to use as a watermark to each printed page of the report.

Parameters

Table 16 describes the parameters that you must provide values for in the Public Sub AddNamedShapeAsWatermark syntax.

Table 16. Public Sub AddNamedShapeAsWatermark parameters

Parameter	Description
name	Specifies the name of the shape (normally an image) to use as a watermark. The Excel application automatically assigns names to all shapes.
rng	Specifies the range of cells used to display the watermark.

Public Function GetXmlElement

```
Public Function GetXmlElement(XPath As String) As String
```

Purpose

This function returns the element content for the specified XPath of the current XML source file.

Parameters

Table 17 describes the parameters that you must provide values for in the Public Function GetXmlElement syntax.

Table 17. Public Function GetXmlElement parameters

Parameter	Description
XPath	Specifies the XPath to a node in an XML document. XPath might contain expressions.

Public Function GetXmlNodesText

```
Public Function GetXmlNodesText(XPath As String) As String()
```

Purpose

This function returns a string array containing the nodes for the specified XPath of the current XML source file.

Parameters

Table 18 describes the parameters that you must provide values for in the Public Function GetXmlNodesText syntax.

Table 18. Public Function GetXmlNodesText parameters

Parameter	Description
XPath	Specifies the XPath to a node in an XML document. XPath might contain expressions.

Public Function NameExists

```
Public Function NameExists(name As String) As Boolean
```

Purpose

This function determines whether the specified named range is defined.

Parameters

Table 19 describes the parameters that you must provide values for in the Public Function NameExists syntax.

Table 19. Public Function NameExists parameters

Parameter	Description
name	Specifies the name of the named range to check. • True: The specified named range exists. • False: The specified named range does not exist.

Public Function GetNamedRange

```
Public Function GetNamedRange(ByVal name As String, c1 As String, r1 As
Long, Optional c2 As String, Optional r2 As Long) As Boolean
```

Purpose

This function determines whether the specified named range is found.

Parameters

Table 20 describes the parameters that you must provide values for in the Public Function GetNamedRange syntax.

Table 20. Public Function GetNamedRange parameters

Parameter	Description
name	Specifies the name of the named range to retrieve.
c1	Returns the start column letter if the named range is found. • True: The specified named range exists. The remaining *c1*, *r1*, *c2*, *r2* options return the cell range. • False: The specified named range does not exist.
r1	Returns the start row number if the named range is found. • True: The specified named range exists. The remaining *c1*, *r1*, *c2*, *r2* options return the cell range. • False: The specified named range does not exist.
c2	Returns the end column letter if the named range is found. • True: The specified named range exists. The remaining *c1*, *r1*, *c2*, *r2* options return the cell range. • False: The specified named range does not exist. This parameter is optional.
r2	Returns the end row number if the named range is found. • True: The specified named range exists. The remaining *c1*, *r1*, *c2*, *r2* options return the cell range. • False: The specified named range does not exist. This parameter is optional.

Public Sub HideColumns

```
Public Sub HideColumns(ByVal target, Optional ByVal hide As Boolean =
True)
```

Purpose

This function makes the target range invisible or visible.

Parameters

Table 21 describes the parameters that you must provide values for in the Public Sub HideColumns syntax.

Table 21. Public Sub HideColumns parameters

Parameter	Description
target	Specifies the target range to hide. The target range could be a named range (for example, QuanChart), a cell range (for example, A1:E5), a row range (for example, 10:30), or a column range (for example, A:C).
hide	Determines whether the target range is invisible or visible. • True: Makes the target range invisible. • False: Makes the target range visible. The default is True. This parameter is optional.

Public Sub SetPageBreakAt

```
Public Sub SetPageBreakAt(ByVal target)
```

Purpose

This function sets a page break at the specified cell location or at the beginning of a cell range.

Parameters

Table 22 describes the parameters that you must provide values for in the Public Sub SetPageBreakAt syntax.

Table 22. Public Sub SetPageBreakAt parameters

Parameter	Description
target	Specifies the target range for setting the page break.

Public Sub SetFrame

```
Public Sub SetFrame(ByVal target, Optional ByVal vLines As Boolean =
False, Optional ByVal hLines As Boolean = False)
```

Purpose

This function sets a border frame around the specified cell range. You can use the optional parameters to create a frame of horizontal lines only or vertical lines only.

Table 23 describes the parameters that you must provide values for in the Public Sub SetFrame syntax.

Table 23. Public Sub SetFrame parameters

Parameter	Description
target	Specifies the target range of the border frame.
vLines	Specifies whether vertical lines appear in the border frame. • True: Vertical lines appear. • False: Vertical lines do not appear. The default is False. This parameter is optional.
hLines	Specifies whether horizontal lines appear in the border frame. • True: Horizontal lines appear. • False: Horizontal lines do not appear. The default is False. This parameter is optional.

Public Function GetRowIndex

```
Public Function GetRowIndex(str As String, col As String, startRow As
Long) As Long
```

Purpose

This function returns the row index of the specified string at a specific column or –1 if does not find the specified string. It stops searching if it encounters an empty cell.

Parameters

Table 24 describes the parameters that you must provide values for in the Public Function GetRowIndex syntax.

Table 24. Public Function GetRowIndex parameters

Parameter	Description
str	Specifies the string to search for.
col	Specifies the column letter of the specified string.
startRow	Specifies the row where the search starts.

Public Function GetColCount

```
Public Function GetColCount(row As Long, startCol As String, Optional
toCol As Long = 0) As Long
```

Purpose

This function returns the count of the non-empty cells in a specified row between two columns.

Parameters

Table 25 describes the parameters that you must provide values for in the Public Function GetColCount syntax.

Table 25. Public Function GetColCount parameters

Parameter	Description
row	Specifies the row number.
startCol	Specifies the letter of the first column to count.
toCol	Specifies the letter of the last column to count. If it is 0, the function counts all columns. The default is 0. This parameter is optional.

Public Function GetRowCount

```
Public Function GetRowCount(col, Optional startRow As Long = 1, Optional
toRow As Long = 0) As Long
```

Purpose

This function returns the count of the non-empty cells in a specified column between designated rows.

Parameters

Table 26 describes the parameters that you must provide values for in the Public Function GetRowCount syntax.

Table 26. Public Function GetRowCount parameters

Parameter	Description
col	Specifies the column number or letter.
startRow	Specifies the number of the first row to count. The default is 1. This parameter is optional.
toRow	Specifies the number of the last row to count. If it is 0, this function counts all rows. The default is 0. This parameter is optional.

Public Function ColumnNumber

```
Public Function ColumnNumber(ByVal ColumnLetter As String) As Long
```

Purpose

This function converts the specified column letter to a number, for example, 1 for column A or 2 for column B.

Parameters

Table 27 describes the parameters that you must provide values for in the Public Function ColumnNumber syntax.

Table 27. Public Function ColumnNumber parameters

Parameter	Description
ColumnLetter	Specifies the column letter.

Public Function DeleteName

```
Public Function DeleteName(name As String) As Boolean
```

Purpose

This function determines whether to delete or retain the named range.

Parameters

Table 28 describes the parameters that you must provide values for in the Public Function DeleteName syntax.

Table 28. Public Function DeleteName parameters

Parameter	Description
name	Specifies the name of the named range to be deleted or retained. • True: Deletes the named range. • False: Retains the named range.

Public Function GetUniqueList

```
Public Function GetUniqueList(ByVal target, Optional ByVal col As Long =
1, Optional startIndex As Long = 1, Optional ByVal inclHiddenRows As
Boolean = False) As String()
```

Purpose

This function returns an array of unique strings containing the text for the specified column from the *startIndex*. It can optionally include hidden columns in the returned array.

Parameters

Table 29 describes the parameters that you must provide values for in the Public Function GetUniqueList syntax.

Table 29. Public Function GetUniqueList parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
col	Specifies the column to retrieve the list from. The default is 1. This parameter is optional.
startIndex	Specifies the start index of the rows. The default is 1. This parameter is optional.
inclHiddenRows	Specifies whether the returned list should include hidden rows. • True: Includes hidden rows. • False: Does not include hidden rows. The default is False. This parameter is optional.

Public Function GetListColumn

```
Public Function GetListColumn(ByVal tableName As String, ByVal columnName
As String) As String()
```

Purpose

This function returns an array of strings containing the text for the specified column.

Parameters

Table 30 describes the parameters that you must provide values for in the Public Function GetListColumn syntax.

Table 30. Public Function GetListColumn parameters

Parameter	Description
tableName	Specifies the name of the data table.
columnName	Specifies the name of the column, which is the column header text.

Public Function Value

```
Public Function Value(ByVal target, Optional ByVal rowOffset As Long = 0,
Optional ByVal colOffset As Long = 0)
```

Purpose

This function returns the value of the target cell with optional offsets.

Parameters

Table 31 describes the parameters that you must provide values for in the Public Function Value syntax.

Table 31. Public Function Value parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
rowOffset	Specifies the offset of the row. The default is 0. This parameter is optional.
colOffset	Specifies the offset of the column. The default is 0. This parameter is optional.

Public Function GetValue

```
Public Function GetValue(ByVal target, ByVal defaultValue)
```

Purpose

This function returns the value of the target cell or the specified default value if the target is not found.

Parameters

Table 32 describes the parameters that you must provide values for in the Public Function GetValue syntax.

Table 32. Public Function GetValue parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
defaultValue	Specifies the default value if the target cannot be found. The default is nothing. This parameter is optional.

Public Function HideListColumn

```
Public Function HideListColumn(listName As String, colName As String)
```

Purpose

This function hides the specified column in the specified data table.

Parameters

Table 33 describes the parameters that you must provide values for in the Public Function HideListColumn syntax.

Table 33. Public Function HideListColumn parameters

Parameter	Description
listName	Specifies the name of the list or data table.
colName	Specifies the name of the column, which is the column header text.

Public Sub SortList

```
Public Sub SortList(listName As String, sortColNames As String, Optional
sheet As Worksheet)
```

Purpose

This function sorts the list or data table by the specified column.

Parameters

Table 34 describes the parameters that you must provide values for in the Public Sub SortList syntax.

Table 34. Public Sub SortList parameters

Parameter	Description
listName	Specifies the name of the list or data table.
sortColName	Specifies the name of the column, which is the column header text.
sheet	Specifies the sheet (also called the tab). If this parameter is missing, the function uses the active sheet by default. This parameter is optional.

Public Sub EnumerateColumn

```
Public Sub EnumerateColumn(ByVal target, ByVal colIndex As Long)
```

Purpose

This function enumerates the specified column to create a sequential number (1, 2, 3, and so forth).

Parameters

Table 35 describes the parameters that you must provide values for in the Public Sub EnumerateColumn syntax.

Table 35. Public Sub EnumerateColumn parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
colIndex	Specifies the index of the column.

Public Function CreatePivotTable

```
Public Function CreatePivotTable(ByVal dataTableName As String, ByVal
target, ByVal rowFields As String, ByVal columnFields As String, ByVal
valueFields As String, Optional ByVal pivotTableName = "PivotTable1") As
Range
```

Purpose

The Excel application is a powerful tool for creating pivot tables. However, creating these tables is normally a manual process that involves several steps, which many Excel users might not understand well. The CreatePivotTable function tries to simplify the process by hiding many internal details and exposing only a few relevant parameters to the template developer. The goal is to create a pivot table that is ready to be used (copied) in custom reports; for example, custom reports never use background colors and borders. You can find an example of this function in Chapter 6, “Creating a Calibration Report.”

Parameters

Table 36 describes the parameters that you must provide values for in the Public Function CreatePivotTable syntax.

Table 36. Public Function CreatePivotTable parameters

Parameter	Description
dataTableName	Specifies the name of the data table that the pivot table is based on.
target	Specifies a range of cells by a named range, a cell range, or a cell.
rowFields	Specifies the column names of the original data table to translate into pivot table rows. You can concatenate multiple columns by using semicolons.
columnFields	Specifies the column names of the original data table to translate into pivot table columns. You can concatenate multiple columns by using semicolons.
valueFields	Specifies the column names of the original data table to translate into pivot table values. You can concatenate multiple column names by using semicolons.
pivotTableName	Specifies the name of the resulting pivot table. The Excel application normally assigns a default name to a pivot table automatically. If you need to programmatically refer to the pivot table, you must give it a name so you can refer to it. The default name is PivotTable1. This parameter is optional.

Public Sub SetManuallyIntegrated

```
Public Sub SetManuallyIntegrated(ByVal dataTableName As String, ByVal
refColName As String, ByVal valueColName As String, Optional ByVal
negateValue As Boolean = False, Optional ByVal extraCols As Long = 0)
```

Purpose

In Thermo Fisher Scientific reports, the manually integrated values are normally displayed with a rectangular box around them. The Public Sub SetManuallyIntegrated function is specifically created for two primary purposes:

- You can use it to format the data table for final display. In this case, *refColName* is normally a hidden column containing the manual integration flag. *ValueColName* is the column enclosed in a box. If more than one column is to be enclosed in a box, you can use the *extraCol* parameter. For example, a long compound name might use three columns, although its value is defined in one (the first) column. In this case, *extraCols* must be 2.

- You can use it to reformat the data for further processing. For example, in creating the calibration report, you can use this function to format the data table before it is translated into a pivot table. This step is necessary because the Excel pivot table can only work with numbers.

Parameters

Table 37 describes the parameters that you must provide values for in the Public Sub SetManuallyIntegrated syntax.

Table 37. Public Sub SetManuallyIntegrated parameters (Sheet 1 of 2)

Parameter	Description
dataTableName	Specifies the name of the data table.
refColName	Specifies the name of the column, which is the column header text, that acts as a reference value.
valueColName	Specifies the name of the column containing the value to be manipulated.

Table 37. Public Sub SetManuallyIntegrated parameters (Sheet 2 of 2)

Parameter	Description
negateValue	Specifies whether the value should be negated. • True: Negates the value. • False: Does not negate the value. The default is False. This parameter is optional.
extraCols	Specifies the number of extra columns to be included in the manual integration frame. The default is 0. This parameter is optional.

Public Sub ReformatManuallyIntegrated

```
Public Sub ReformatManuallyIntegrated(ByVal target As Range)
```

Purpose

This function is a special-purpose function that adds a box around the cell in a specified column whose value is negative. It also turns a negative value to positive. This function is used in the calibration report for post-processing the manual integration flags after the pivot table is created.

Parameters

Table 38 describes the parameters that you must provide values for in the Public Sub ReformatManuallyIntegrated syntax.

Table 38. Public Sub ReformatManuallyIntegrated parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.

Public Sub SetNoData

```
Public Sub SetNoData(ByVal target, Optional ByVal text As String = "No
data for this report")
```

Purpose

This function sets specified text in the target cells. You can use it to create more consistent reports.

Parameters

Table 39 describes the parameters that you must provide values for in the Public Sub SetNoData syntax.

Table 39. Public Sub SetNoData parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
text	Specifies the text to display. This parameter defaults to “No data for this report.” This parameter is optional.

Public Sub HideRepeatArea

```
Public Sub HideRepeatArea(Optional name As String = "RepeatArea",
Optional hideHelperColumns As Boolean = True)
```

Purpose

This function hides the rows in the named range of the repeat area (RepeatArea). It is normally called after the sheet is processed. Some templates have helper columns (for example, data tables) next to the repeat area that must be hidden as well. The optional *hideHelperColumns* parameter hides the helper columns in one function call.

Parameters

Table 40 describes the parameters that you must provide values for in the Public Sub HideRepeatArea syntax.

Table 40. Public Sub HideRepeatArea parameters

Parameter	Description
name	Specifies the name of the repeat area. The named range of the repeat area is normally defined as RepeatArea. The default is RepeatArea. This parameter is optional.
hideHelperColumns	Specifies whether to hide the helper columns (normally to the right of the repeat area). • True: Hides the helper columns. • False: Does not hide the helper columns. The default is True. This parameter is optional.

Public Sub FormatValue

```
Public Sub FormatValue(ByVal target, ByVal limitValue, ByVal biggerFormat
As String, ByVal smallerFormat As String, Optional ByVal zeroFormat =
Nothing)
```

Purpose

This function formats the cell value on the basis of the limit value and given formats.

Parameters

Table 41 describes the parameters that you must provide values for in the Public Sub FormatValue syntax.

Table 41. Public Sub FormatValue parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
limitValue	Specifies the limit value.
biggerFormat	Specifies the format to use for the value bigger than *limitValue*.
smallerFormat	Specifies the format to use for the value smaller than *limitValue*.
zeroFormat	Specifies the format to use for zero-value cells. This parameter defaults to nothing. If it is set to nothing, zero-value cells are turned into empty cells. This parameter is optional.

Public Sub ReplaceValue

```
Public Sub ReplaceValue(ByVal target, ByVal fromValue, ByVal toValue)
```

Purpose

This function replaces one cell value with another.

Parameters

Table 42 describes the parameters that you must provide values for in the Public Sub ReplaceValue syntax.

Table 42. Public Sub ReplaceValue parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
fromValue	Specifies the value to replace.
toValue	Specifies the value that replaces *fromValue*.

Public Function ShiftRange

```
Public Function ShiftRange(ByVal target, ByVal left As Long, ByVal top As
Long, Optional ByVal right As Long = 0, Optional ByVal bottom As Long = 0)
As Range
```

Purpose

This function shifts the target range by the specified margins. If the right and bottom parameters are non-zero, it also shrinks or expands the original range. This function is often used in a loop to calculate the range for images.

Parameters

Table 43 describes the parameters that you must provide values for in the Public Function ShiftRange syntax.

Table 43. Public Function ShiftRange parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
left	Specifies the shift value to apply to the left of the range.
top	Specifies the shift value to apply to the top of the range.
right	Specifies the shift value to apply to the right of the range. The default is 0. If it is 0, no shift is applied. This parameter is optional.
bottom	Specifies the shift value to apply to the bottom of the range. The default is 0. If it is 0, no shift is applied. This parameter is optional.

Public Function GetText

```
Public Function GetText(ByVal target, Optional ByVal rowDelim As String =
vbLf, Optional ByVal colDelim As String = vbTab) As String
```

Purpose

This function returns a string containing the text of the specified target range with the given delimiters.

Parameters

Table 44 describes the parameters that you must provide values for in the Public Function GetText syntax.

Table 44. Public Function GetText parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
rowDelim	Specifies the delimiter to use to concatenate the rows. The default is vbLf. This parameter is optional.
colDelim	Specifies the delimiter to be used to concatenate the columns. The default is vbTab. This parameter is optional.

Public Sub MergeRowCells

```
Public Sub MergeRowCells(ByVal target, Optional ByVal rowAutoFit As
Boolean = True)
```

Purpose

This function merges cells in rows in the specified target range. It is normally used to accommodate long values that expand to multiple cells. Text type values can run into the next cell as long as the next cell is empty. However, other type values (for example, dates or numbers) cannot run into the next cell. Instead, "#" appears in the cell if the values cannot fit. In this case, you must merge the cells to get the values to display correctly in final reports.

Parameters

Table 45 describes the parameters that you must provide values for in the Public Sub MergeRowCells syntax.

Table 45. Public Sub MergeRowCells parameters

Parameter	Description
target	Specifies a range of cells by a named range, a cell range, or a cell.
rowAutoFit	Specifies whether the row height should automatically fit the text after the merge. • True: The row height automatically expands to fit the text. • False: The row height does not automatically expand to fit the text. The default is True. This parameter is optional.

Public Function ArraySize

```
Public Function ArraySize(arr() As String) As Long
```

Purpose

This function returns the size of an array. It returns 0 if the array is not initialized.

Parameters

Table 46 describes the parameters that you must provide values for in the Public Function ArraySize syntax.

Table 46. Public Function ArraySize parameters

Parameter	Description
arr	Specifies the array to check.

Public Sub LogMsg

```
Public Sub LogMsg(ByVal msg As String, Optional ByVal fileName As String =
"")
```

Purpose

This function adds a message to the log file. It automatically prepends each message in the log file with a time stamp. If *fileName* is “”, the log message is not logged unless the command line contains the /g option. The command line in the application does not include this option, so no log file is generated with the report. This function is primarily designed for developing templates.

Parameters

Table 47 describes the parameters that you must provide values for in the Public Sub LogMsg syntax.

Table 47. Public Sub LogMsg parameters

Parameter	Description
msg	Specifies the message text to add to the log file.
fileName	Specifies the name of the log file. If you do not specify *fileName*, the function uses the default log file name specified by the /g, /d, or /t command-line option. The default is “”. This parameter is optional.

Public Function GetValidSheetName

```
Public Function GetValidSheetName(name As String) As String
```

Purpose

This function returns a string as a valid sheet name that is based on the specified name. The sheet name in the Excel application is like a file name in Windows, which does not allow certain specific characters. This function replaces those special characters with a space. In addition, the sheet name cannot exceed 30 characters, so this function trims long names up to 25 characters and appends ellipses (…). Furthermore, sheet names in an Excel workbook cannot have duplicates. If the same name sheet already exists, this function adds a number to the name. This function is primarily used to make a compound name a qualified sheet name, which is used in many reports.

Parameters

Table 48 describes the parameters that you must provide values for in the Public Function GetValidSheetName syntax.

Table 48. Public Function GetValidSheetName parameters

Parameter	Description
name	Specifies the name to be used as the basis of the sheet name.

Public Function FileExists

```
Public Function FileExists(ByVal fileName As String) As Boolean
```

Purpose

This function determines whether the specified file already exists. This function is a Windows API wrapper.

Parameters

Table 49 describes the parameters that you must provide values for in the Public Function FileExists syntax.

Table 49. Public Function FileExists parameters

Parameter	Description
fileName	Specifies the name of the file to check for. • True: The specified file already exists. • False: The specified file does not already exist.