# XLReportCom

# User Manual

**Version 1.4**

2009-07

# Contents

# Chapter 1 Introduction

## 1.1 Overview

XLReportCom is a solution that generates reports using Microsoft Excel. Using Microsoft Excel and XLReportCom, you can create all kinds of reports quickly and easily. XLReportCom includes an ActiveX DLL and an executable file that can be used to develop your applications. It will significantly accelerate your application development.

XLReportCom is a template-based solution. To create a report, you need to create a report template file first. The report template file is a Microsoft Excel workbook that defines the layouts and formats of a report. XLReportCom retrieves data from data source and fills data into Excel workbooks.

## 1.2 Features

XLReportCom includes the following features:

- Using Microsoft Excel as your reporting tool

Just use Microsoft Excel as your reporting tool. You design reports like layouts, formats and styles directly using Microsoft Excel. And you will get reports in Microsoft Excel spreadsheet format as a result. Microsoft Excel is powerful, flexible and familiar. You do not need to buy and learn extra reporting tools.

- Making report template directly using Microsoft Excel

The main advantage of using XLReportCom is based on the fact that all formatting is done directly in Microsoft Excel. You can take full advantage of Microsoft Excel including cell formatting, formulas, filtering and sorting, drawing and pictures, charts, multiple sheets, page setup, headers and footers, preview and printing, VBA, macros, and more.

- Accessing to databases using SQL

XLReportCom executes SQL statements to extract data from database. Supports all type SQL: DML, DDL and DCL. Multiple SQL statements can be executed in one report building process. You can perform queries on databases, insert data into databases, and create database objects like tables. The power of SQL can be harnessed for maximum efficiency in reporting.

- Using ADO to access and manipulate data sources

Using ADO, XLReportCom can access and manipulate a wide variety of data sources such as Oracle, DB2, Sybase, Informix, Microsoft SQL Server, Teradata, MySQL, Microsoft Access, dBase.

- Integrating Microsoft Excel into your application

XLReportCom includes an ActiveX DLL for building application. Developers can save time and meet their users needs by integrating the report processing power of XLReportCom into their applications.

- Command line program

XLReportCom includes a command line program ExcelReport.exe. You can use the program to create reports too. It does not require programming. It is enough if you know how to use Microsoft Excel and how to write SQL.

- Various reporting capabilities

XLReportCom provides various reporting capabilities including sorting data, grouping data, subreports, totaling and summarizing data, formatting, charting and pictures. It is easy to create simple reports, and, you can create complex reports.

# Chapter 2 Installation and Startup

## 2.1 Software Requirements

Microsoft Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista or later.
Microsoft Office 97/98, Office 2000, Office XP, Office 2003 or later.

## 2.2 Installing XLReportCom

Run the installation program, and follow the instructions to complete XLReportCom installation.
If you don’t have Microsoft Office installed, please install it first.
If your environment is Windows 95/98 and Office 97, and you don’t have VB6.0 run-time files installed, please install it. For Windows 2000, Windows XP, Windows 2003 and Office 2000 or later, you do not need to install VB6.0 run-time files because they are included in OS and Office. To install VB6.0 run-time files, just run vbrun60sp5.exe, and follow the instructions.
If you don’t have ODBC Driver installed for the database you want to access, please install it.
If your OS is Windows 95/98 and you don’t have Microsoft Data Access Components 2.0 (MDAC_TYP) or later installed, please install it. For Windows 2000, Windows XP and Windows 2003, you do not need to install MDAC_TYP because it is preinstalled in OS. To install MDAC_TYP, just run mdac_typ.exe, and follow the instructions.

## 2.3 Uninstalling XLReportCom

1. Double-click the **Add/Remove Programs** icon in the Windows Control

Panel.

2. Do one of the following:

- For Windows 2000, Windows XP and Windows 2003 Edition:

Click **XLReportCom** in the **Currently installed programs** box, and then click the **Change/Remove** button.

- For Windows 98 and Windows NT 4.0:

Click **XLReportCom** on the **Install/Uninstall** tab, and then click the **Add/Remove** button.

3. Follow the instructions on the screen to complete uninstalling the program.

## 2.4 XLReportCom.dll

XLReportCom.dll is an ActiveX DLL that provides XLReport object. You can write a program to work with the object. Before you can use the XLReport object, you must create a reference to the object. And you should create references to Microsoft Excel Object Library too.

To create a reference to the XLReport object

1. Do one of the following:

- For Visual Basic 6.0

From the **Project** menu, choose **References**.

- For Microsoft Excel Visual Basic For Application

From the **Tools** menu, choose **References**.

2. In the **References** dialog box, select **XLReportCom**.

3. You can use the **Browse** button to search for XLReportCom.dll.

4. In the **References** dialog box, Select Microsoft Excel Object Library to create their references.

5. Declare an object variable of the object's class.

Dim xlrpt As XLReport

6. Assign an object reference to the variable by using the New keyword in a

Set statement.

Set xlrpt = New XLReport

## 2.5 ExcelReport.exe

ExcelReport.exe is an executable program that developed using XLReportCom.dll. It likes XLReportGen command line and can read an XRF file to create an Excel report. The syntax of command is:

*excelreport <xrf file name> [-D] [-U1 user1] [-P1 pwd1] … [-U10 user10] [-P10 pwd10] [pa1 pa2 … pa10]*

| | |
|---|---|
| xrf file name | Specifying an XRF (.xrf) file that tells XLReportCom how to get data from data sources and how to put data into a report. |
| -D | Display the generated report with Microsoft Excel. |
| -U1 user1 … -U10 user10 | Specify the user names. user1 is the user name of the first data source. user2 is the user name of the second data source…… |
| -P1 pwd1 ... -P10 pwd10 | Specify the passwords. pwd1 is the password of the first data source. pwd2 is the password of the second data source…… |
| pa1 … pa10 | The values of the parameters defined in the XRF file. You can use parameters in SQL statements. XLReportCom will replace the names of the parameters in a SQL statement with the actual values before it executes the SQL statement. You can use no more than 10 parameters in one report. |

For example, you have defined two parameters in your XRF file. The first parameter is the sales date, and the second is the category of the product. You can run ExcelReport.exe as follows:

excelreport c:\excelreport\myreport.xrf 1996-05-01 “Dairy Products”

## 2.6 Run-Time Files

You can distribute royalty-free the run-time files of XLReportCom with your applications. The run-time files are files your application must have in order to work correctly after installation. The following are the run-time files you need to distribute:

| File | Description |
|---|---|
| xlreportcom.dll | The XLReportCom ActiveX DLL. It must be registered. |
| xconv.cfg | The file contains the information of the file format. If you are using ExcelReport method to convert files, you should include it and copy it to the same directory as xlreportcom.dll. |
| scrrun.dll | Microsoft script runtime. XLReportCom used some functions in this file. It should be copied to Windows System directory, and must be registered. |

To register a DLL file, use regsvr32.exe. For example,

regsvr32.exe /s “C:\Program Files\LJZsoft\XLReportCom\XLReportCom.dll”

# Chapter 3 Quick Start

## 3.1 Learning how to use XLReportCom

You can teach yourself how to use XLReportCom by choosing from the methods available in this section:

- You can study the samples included with XLReportCom.
- You can use the detailed descriptions and instructions in this document.

## 3.2 Sample Database

XLReportCom comes with Sample.mdb, a sample database you can use when learning the program. Sample.mdb is a Microsoft Access database. Virtually all of the examples in this manual are based on Sample.mdb data.

The sample reports access the sample database through the ODBC data source name “Report Sample”. When you install XLReportCom, you can choose to add the ODBC data source name. And you also can add the ODBC data source name manually.

To create the System DSN “Report Sample”, do as follows:

1. Click the Windows **Start** button, choose **Settings**, and then click **Control Panel**.

2. On computers running Microsoft Windows 2000 or later, double-click **Administrative Tools**, and then double-click **Data Sources (ODBC)**. The **ODBC Data Source Administrator** dialog box appears. On computers running previous versions of Microsoft Windows, double-click **32-bit ODBC** or **ODBC**.

3. Select the **System DSN** tab, and then press **Add** button.

4. Choose **Microsoft Access Driver (*.mdb)**, then press **Finish** button.

5. In the **ODBC Microsoft Access Setup** dialog box, type **Report Sample** in

the **Data Source Name** box.

6. Press the **Select** button, and browse to select **Sample.mdb**.

7. Press **OK** button to close the **ODBC Microsoft Access Setup** dialog box.

8. Press **OK** button to close the **ODBC Data Source Administrator** dialog box.

## 3.3 Samples

After XLReportCom is installed, some samples are installed too. Use these samples to learn XLReportCom.

The samples include a sample database, VB sample programs, VBA sample programs and sample reports. They are located in the Application Data\LJZsoft under All Users or your profile folder. XLReportCom was tested with Microsoft Office 2007. Please download the sample reports for Microsoft Office 2007 from our website.

| Directory | Description |
|---|---|
| {commonappdata}\LJZsoft\Common\SampleDatabase | Contains the sample database “Sample.mdb”. |
| {commonappdata}\LJZsoft\XLReportCom\Samples\ExcelReport | Contains the report template files (.xls) and the XRF files (.xrf). |
| {commonappdata}\LJZsoft\XLReportCom\Samples\VB | Contains the sample programs for VB6.0. |
| {commonappdata}\LJZsoft\XLReportCom\Samples\VBA | Contains the sample programs for Microsoft Excel VBA. |

{commonappdata} is the path to the Application Data folder under All Users. If you install XLReportCom without administrative privileges, {commonappdata} is the path to the Application Data folder under the current user. The Application Data folder is usually at:

Windows 95/98: C:\windows\All Users\Application Data\

Windows NT: C:\WinNT\Profiles\All Users\Application Data\

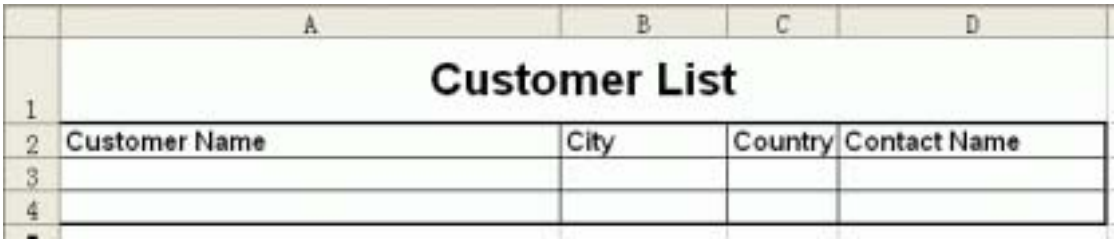
Windows 2000/XP: C:\Documents and Settings\All Users\Application Data\

Windows Vista: C:\ProgramData\

## 3.4 Creating a Report Programmatically

1. Create a template

In Microsoft Excel, create a report template file named “custlist.xls”. Static values and any Excel features included in the template will be included in the generated report. The template file you have created as follows:



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Customer List | | | |
| 2 | Customer Name | City | Country | Contact Name |
| 3 | | | | |
| 4 | | | | |

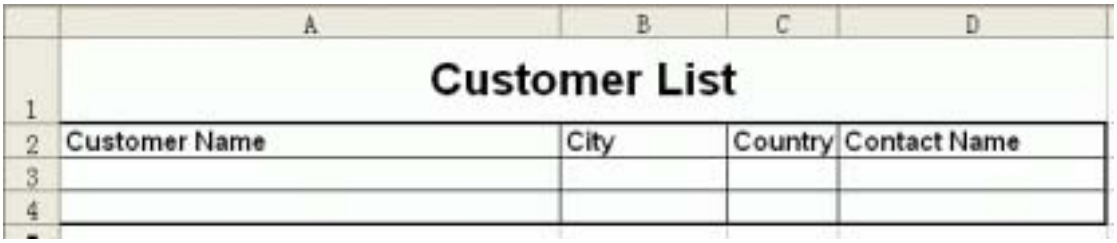
2. Write the code in your application.

```
Set con = New ADODB.Connection
Set rec = New ADODB.Recordset
con.ConnectionString = "Data Source=Report Sample"
con.Open
strSQL = "SELECT CompanyName, CityName, CountryName,
ContactName FROM Customers, Cities, Countries WHERE
Customers.CityCode = Cities.CityCode AND Customers.CountryCode =
Cities.CountryCode AND Customers.CountryCode = Countries.CountryCode
ORDER BY CompanyName, CityName, CountryName"
rec.Open strSQL, con
xlrpt.VarTableReport Recordset:=rec, Worksheet:=xlWorksheet,
CellList:="A3", Reserve:=2
rec.Close
```

## 3.5 Creating a Report with ExcelReport.exe

1. Create a template

In Microsoft Excel, create a report template file named “custlist.xls”. Static values and any Excel features included in the template will be included in the generated report. The template file you have created as follows:



| | A | B | C | D |
|---|---|---|---|---|
| 1 | Customer List | | | |
| 2 | Customer Name | City | Country | Contact Name |
| 3 | | | | |
| 4 | | | | |

2. Create an XRF file

Create an XRF file named “custlist.xrf” using XLReportGen or a text editor. The following is the content of the XRF file.

```
ExcelReport Version 2.0
[Data Source]
Name1=Report Sample

[File]
ReportTemplateFileName=custlist.xls
ReportFileName=Report\custlist.xls
LogFileName=Log\custlist.log

[SQL]
@F1=Report(sheet=1 cell=A3 reserve=2)
SELECT CompanyName
,CityName
,CountryName
```

```
,ContactName
FROM Customers, Cities, Countries
WHERE Customers.CityCode = Cities.CityCode
AND Customers.CountryCode = Cities.CountryCode
AND Customers.CountryCode = Countries.CountryCode
ORDER BY CompanyName,CityName,CountryName
```

3. Run ExcelReport.exe

```
excelreport c:\report\custlist.xrf
```

# Chapter 4 Report Templates

## 4.1 About Reports

The report generated by XLReportCom is a Microsoft Excel workbook that contains one or more worksheets. The layouts, formats and styles of the report are defined by a report template, and the data of the report are got from databases such as Oracle, DB2.

## 4.2 About Report Templates

To make a report using XLReportCom, you should create a report template first. This report template is a Microsoft Excel workbook that defines the layouts, formats and styles of the report. In the Microsoft Excel report template, you can input static contents such as titles, descriptions, comments, a cover, a company logo, format the static content, and define the format of the cells you will fill data.

XLReportCom will generate the report based on the report template file. All static contents and the layouts, formats and styles defined in the report template file will be brought to the final report file.

## 4.3 Excel Basic Concepts

If you have known these concepts of Microsoft Excel, please skip this section. For more detail information about Microsoft Excel, refer to *Microsoft Excel Help*.

### 4.3.1 Workbooks and Worksheets

A Microsoft Excel workbook is a file that contains one or more worksheets, which you can use to organize various kinds of related information. You can enter and edit data on several worksheets simultaneously and perform calculations based on data from more than one worksheet. When you create a chart, you can place the chart on the same worksheet as its related data or on a separate chart sheet.

Worksheet is the primary document that you use in Microsoft Excel to store and work with data. It also called a spreadsheet. A worksheet consists of cells that are organized into columns and rows; a worksheet is always stored in a workbook.

### 4.3.2 Formulas

Formulas are equations that perform calculations on values in your worksheet. A formula starts with an equal sign (=). A formula can contain any or all of the following: functions, references, operators, and constants. You can perform calculations using formulas.

### 4.3.3 Names

A name is a word or string of characters that represents a cell, range of cells, formula, or constant value. Use easy to understand names, such as Products to refer to hard to understand ranges, such as Sales!C20:C30.

### 4.3.4 Headers and Footers

Headers and footers are areas in the top and bottom margins of a worksheet. You can add a header and footer on each worksheet. You can insert a page number, date and time, graphic, file name in a header and footer, and change

the font in header and footer text. You can have only one custom header and one custom footer on each worksheet. If you create a new custom header or footer, it replaces any other custom header or footer on the worksheet.

### 4.3.5 Page Breaks

Microsoft Excel will break pages automatically. You can manually insert horizontal or vertical page breaks.

### 4.3.6 Drawings, Pictures and Diagrams

You can add graphics to your worksheets and charts to make them more visually appealing, to create eye-catching reports, or to add emphasis. For example, you can display a logo on your worksheet, create a flowchart, and use graphics in chart data markers. You can make your worksheet interactive by using graphic objects as hyperlinks or by assigning buttons to macros.

### 4.3.7 Charts

Charts are visually appealing and make it easy for users to see comparisons, patterns, and trends in data. To create a chart, you must first enter the data for the chart on the worksheet. Then select that data and create a chart. A chart is linked to the worksheet data it's created from and is updated automatically when you change the worksheet data.

### 4.3.8 Formatting

You can use these formatting features of Microsoft Excel to effectively display your data.

- Format text and individual characters

To make text stand out, you can format all of the text in a cell or selected characters. You can set font, color, alignment of the text.

- Rotate text and borders

The data in a column is often very narrow while the label for the column is much wider. Instead of creating unnecessarily wide columns or abbreviated labels, you can rotate text and apply borders that are rotated to the same degree as the text.

- Add borders, colors, and patterns

To distinguish between different types of information in a worksheet, you can apply borders to cells, shade cells with a background color, or shade cells with a color pattern.

- Number formats

You can use number formats to change the appearance of numbers, including dates and times, without changing the number behind the appearance. The number format does not affect the actual cell value that Microsoft Excel uses to perform calculations.

- Conditional formatting

The conditional format is a format, such as cell shading or font color, that Excel automatically applies to cells if a specified condition is true.

- Style

The style is a combination of formatting characteristics, such as font, font size, and indentation, that you name and store as a set. When you apply a style, all of the formatting instructions in that style are applied at one time.

## 4.4 Table Reports

### 4.4.1 About Table Reports

A table is made up of rows and columns of cells that you can fill with text and graphics. Tables are often used to make reports, and organize and present

information.

XLReportCom supports two types of table reports: fixed table report, variable table report.

Fixed table report: The number of rows and columns in the table is fixed. When XLReportCom executes a SQL statement, directly puts the result data into cells in the table.

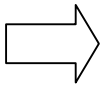
Variable table report: The number of rows or columns in the table is unfixed, and it is variable as the number of result records. When XLReportCom executes a SQL statement, it repeats the table rows or columns for each record or group, and then puts data into cells of the table.

### 4.4.2 Creating a Worksheet for a Fixed Table Report

For a fixed table report, you need to create a worksheet in the report template file according to the report. The format of the worksheet is the same as the format in the report, but the cells that should be filled data into are blank. When XLReportCom executes a SQL statement, the data values from data source will be filled into these cells.

| | A | B |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |

The fixed table defined in the report template file



| | A | B |
|---|---|---|
| 1 | 14 | 3.4 |
| 2 | 20 | 5.2 |
| 3 | 8 | 2.7 |

The fixed table filled data by rows in the report file

### 4.4.3 Creating a Worksheet for a Variable Table Report

For a variable table report, you also need to create a worksheet in the report

template file according to the report. But you just need to reserve some rows/columns in the worksheet for one or two records. XLReportCom will add some rows/columns according to the number of the records returned from data source.

| Date | Item Id | Sales |
|---|---|---|
| | | |
| | | |

| Date | Item Id | Sales |
|---|---|---|
| 1998-01-01 | 3 | 150 |
| 1998-01-02 | 3 | 200 |
| 1998-01-03 | 3 | 250 |
| 1998-01-05 | 3 | 350 |
| 1998-01-10 | 3 | 550 |
| 1998-01-21 | 3 | 150 |
| 1998-01-25 | 3 | 200 |
| 1998-01-31 | 3 | 100 |

The variable-rows table defined in the report template file

The variable-rows table filled data by rows in the report file

One record from data source can be put into two or more rows/columns. To do this, you need to create a repeat range that includes two or more rows/columns.

The format of the last row/column border can be different from the others. For example, the outside borders used double lines, and the inside borders used single lines. To do this, you should reserve the blank rows/columns for 2 records. When XLReportCom inserts some blank rows/columns, the new rows/columns will inherit the format of the first row/column in the reserved rows/columns.

XLReportCom will repeat the range for each record. Ranges can be nested. The inside range is the detail range for detail record, and the external range is the group range for group. XLReportCom will repeat the inside range for each record, and repeat the group range for each group.

### 4.4.4 Formatting a Cell for Pictures

To enhance the visual impact of your report, you can insert pictures into your report. XLReportCom supports many popular graphics file formats: bitmap, JPG, GIF, PNG, TIFF and so on. For the graphics file formats XLReportCom supports, refer to *Microsoft Excel Help*.

You should store the path and name of the graphics files in the database, and identify the image fields in the report function. XLReportCom will read the graphics files, and insert them into the cells in the report file.

To specify the positioning option and size, you should write a formatting expression into the cell in the report template file. XLReportCom will get the text of the cell, and insert a picture into the cell according to the instruction in the format expression. The format expression for pictures as follows:

[placement] [size]

The **placement** specifies the positioning option, and can be one of the following values. The default value is MNS.

| Values | Description |
|---|---|
| MAS | Move and size with cells. |
| MNS | Move but don't size with cells. |
| NMS | Don't move or size with cells. |

The **size** specifies the size of a picture. Possible values are STRETCH, Wnnn or / and Hnnn. "STRETCH" means that the picture is resized to fit within the cell. “W100” means that the width of the picture is set to 100 points. “H50” means that the height of the picture is set to 50 points. The default means the original size. If you just specify the width or height of the picture, not both, XLReportCom will retain the original proportions of the picture when XLReportCom resize it.

**Example**

w84

**Remarks**

On the supposition that the original picture is size 144 x 168 points. XLReportCom will insert a picture, set the positioning option to **Move but don't size with cells**, set the height of the picture to 72 points, and the width to 84 points.

## 4.5 Charts

### 4.5.1 About Charts

Charts are visually appealing and make it easy for users to see comparisons, patterns, and trends in data. You can use Microsoft Excel to add sophisticated, colorful charts in your reports. For example, you can see at a glance whether sales are falling or rising over quarterly periods, or how the actual sales compare to the projected sales. You can create a chart on its own sheet or as an embedded object on a worksheet.

### 4.5.2 Creating a Blank Chart

To create a chart in the report using XLReportCom, you need to add a chart in the report template file. The chart will be brought into the report file with the same chart type, display option, number format, titles, data labels and legends.
To add a chart in the template file:

1. Open the report template file using Microsoft Excel.
2. Enter the sample data for the chart on the worksheet.
3. Select that data and use the Chart Wizard to step through the process of choosing the chart type and the various chart options, or use the **Chart** toolbar to create a basic chart that you can format later.
4. Customize the chart. For example, change the chart type, colors, lines, fills, number formats, titles, data labels and legends in charts.

5. After you have finished the customization, delete data from the chart. You should keep a blank chart in the report template file. You will put data into the worksheet using XLReportCom.

For more detail information, refer to *Microsoft Excel Help*.

# Chapter 5 API Reference

## 5.1 Objects

### 5.1.1 XLReport Object

Represents the XLReportCom. XLReport is the main class for report generation using XLReportCom.

**Using the XLReport Object**

The following example creates an XLReport object in another application and then generates a report using an XRF file.

```
Dim xlrpt As XLReport
Set xlrpt = New XLReport
xlrpt.ExcelReport xlApp, “customer_list.xrf"
```

## 5.2 Methods

### 5.2.1 FixTableReport Method

Generates a fixed table report based on a template. In a fixed table report, the number of rows and columns is fixed. XLReportCom gets data from a recordset object, and directly fills data into the cells of a worksheet.

**Syntax**

*object*.**FixTableReport(*Recordset, Worksheet, CellList, Range, Copy, FillOrder, ImageList, PageBreak*)**

*object* *Required.* The object is the XLReport object.

***Recordset*** Required. An object variable that represents the ADODB.Recordset object to provides data. Before calling this method, please keep the current record position to the first record.

***Worksheet*** Required. An object variable that represents the Excel.Worksheet object to be filled data.

***CellList*** Required. A string that represents the list of cells separated by the "," character. For example, "A2,B2,B3,D2,D3". The cells in the *CellList* should correspond to the data source fields in the recordset. The value of the first field is put into the first cell, and the value of the second field is put into the second cell ……

***Range*** Optional. A string that indicates the range in the worksheet to be used for the records. XLReportCom will skip or repeat the range for each record. You can reference a range of cells like "2:4" or "B2:D5". The default range is the area that includes all cells for the records.

***Copy*** Optional. An integer that indicates whether XLReportCom will copy the range for each record. If the value is zero, XLReportCom will skip the rows/columns of the range for each record. Otherwise it will copy the original range to the range where data will be filled for each record.

***FillOrder*** Optional. An integer that indicates the order in which XLReportCom fills data. If the value is zero, XLReportCom will fill data by rows. 1 means to fill data by columns. Default is 0.

***ImageList*** Optional. A string that indicates which data source fields are the picture files. The *ImageList* is the list of data source fields separated by the "," character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the

worksheet.

***PageBreak*** Optional. A string that indicates the page breaks. The unit of page length is r that means record. For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records. Default is “” that means no page break.

**Example**

This example uses FixTableReport method to make the report “Top 5 Employees for Sales”.

1. Create the template in Microsoft Excel.

| 6 | Rank | Employee Name | Quantity | Amount |
|---|---|---|---|---|
| 7 | 1 | | | |
| 8 | 2 | | | |
| 9 | 3 | | | |
| 10 | 4 | | | |
| 11 | 5 | | | |

2. Write the code in your application.

```
Set con = New ADODB.Connection
Set rec = New ADODB.Recordset
con.ConnectionString = "Data Source=Report Sample"
con.Open
strSQL = "SELECT TOP 5 e.FirstName + ' ' + e.LastName,
SUM(d.Quantity), Sum(d.UnitPrice * d.Quantity * (1-d.Discount)) AS
SalesAmount FROM Orders o, OrderDetails d, Products p, Employees e
WHERE o.OrderID = d.OrderID AND d.ProductID = p.ProductID AND
o.EmployeeID = e.EmployeeID AND YEAR(o.OrderDate) = 1996 AND
MONTH(o.OrderDate) = 04 GROUP BY e.FirstName, e.LastName ORDER BY
3 DESC"
rec.Open strSQL, con
```

```
xlrpt.FixTableReport Recordset:=rec, Worksheet:=xlWorksheet,
CellList:="B7"
rec.Close
```

3. Generate the report.

| 6 | Rank | Employee Name | Quantity | Amount |
|---|---|---|---|---|
| 7 | 1 | Nancy Davolio | 467 | $24,827.45 |
| 8 | 2 | Laura Callahan | 912 | $20,728.13 |
| 9 | 3 | Janet Leverling | 578 | $16,360.12 |
| 10 | 4 | Andrew Fuller | 558 | $13,937.64 |
| 11 | 5 | Margaret Peacock | 481 | $8,298.45 |

## 5.2.2 VarTableReport Method

Generates a variable table report based on a template. In a variable table report, the number of rows or columns in the table is unfixed, and it is variable as the number of the result records. XLReportCom gets data from a recordset object, inserts some blank rows/columns or copy a range for each record, then fills data into the cells of a worksheet.

**Syntax**

*object*.**VarTableReport(*Recordset, Worksheet, CellList, Range, Copy, Reserve, FillOrder, ImageList, PageBreak, NoData*)**

*object* *Required.* The object is the XLReport object.

***Recordset*** Required. An object variable that represents the ADODB.Recordset object to provides data. Before calling this method, please keep the current record position to the first record.

***Worksheet*** Required. An object variable that represents the Excel.Worksheet object to be filled data.

***CellList*** Required. A string that represents the list of cells separated by the "," character. For example, "A2,B2,B3,D2,D3". The cells in the *CellList* should correspond to the data source fields in the recordset. The value of the first field is put into the first cell, and the value of the second field is put into the second cell ……

***Range*** Optional. A string that indicates the range in the worksheet to be used for the records. XLReportCom will repeat the range for each record. You can reference a range of cells like "2:4" or "B2:D5". The default range is the area that includes all cells for the records.

***Copy*** Optional. An integer that indicates whether XLReportCom will copy the range for each record. If the value is zero, XLReportCom will insert the blank rows/columns of the range for each record. Otherwise it will copy the source range and insert the copied range for each record.

***Reserve*** Optional. An integer that indicates the number of records for which you reserved some rows/columns in the report template for the report. Possible values are 1 or 2. One means you reserve some rows/columns for one record, and two means some rows/columns for two records. Default is 1.

***FillOrder*** Optional. An integer that indicates the order in which XLReportCom fills data. If the value is zero, XLReportCom will insert entire rows and fill data by rows. 1 means to insert entire columns and fill data by columns. 2 means to insert range and fill data by rows. 3 means to insert range and fill data by columns. Default is 0.

***ImageList*** Optional. A string that indicates which data source fields are the picture files. The *ImageList* is the list of data source fields separated by the "," character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the

worksheet.

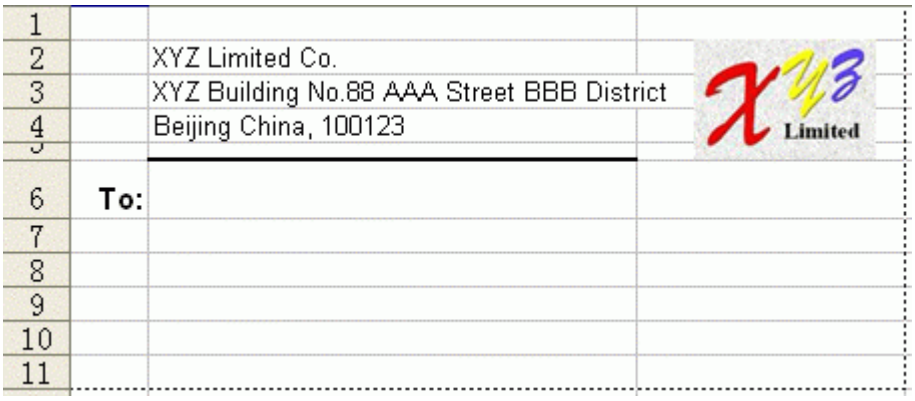
***PageBreak*** Optional. A string that indicates the page breaks. The unit of page length is r that means record. For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records. Default is “” that means no page break.

***NoData*** Optional. An integer that represents an option when no data are returned from data source. If the value is 1, XLReportCom will delete the range when no data are returned. If the value is 2, XLReportCom will delete the sheet when no data are returned. Default is 0. It means to do nothing.

**Example**

This example uses VarTableReport method to make the report “Mail Label”.

1. Create the template in Microsoft Excel.




2. Write the code in your application.

```
Set con = New ADODB.Connection
Set rec = New ADODB.Recordset
con.ConnectionString = "Data Source=Report Sample"
con.Open
strSQL = "SELECT CompanyName, Address, CityName & ', ' &
```

CountryName, PostalCode FROM Customers, Cities, Countries WHERE

Customers.CityCode = Cities.CityCode AND Customers.CountryCode = Cities.CountryCode AND Customers.CountryCode = Countries.CountryCode ORDER BY CompanyName"

rec.Open strSQL, con

xlrpt.VarTableReport Recordset:=rec, Worksheet:=xlWorksheet, CellList:=" B7,B8,B9,B10", Range:="1:11", Copy:=1, PageBreak:="4r"

rec.Close

3. Generate the report.


## 5.2.3 GroupTableReport Method

Generates a variable table report based on a template, and groups data in the report. In a variable table report, the number of rows or columns in the table is unfixed, and it is variable as the number of the result records. XLReportCom gets data from a recordset object, copy the group range for each group, and

copy the detail range for each record.

**Syntax**

*object*.**GroupTableReport(*Recordset, Worksheet, CellList, Range, FillOrder, ImageList, PageBreak, NoData, Group1, GroupRange1, … Group10, GroupRange10*)**

*object* *Required.* The object is the XLReport object.

***Recordset*** Required. An object variable that represents the ADODB.Recordset object to provides data. Before calling this method, please keep the current record position to the first record.

***Worksheet*** Required. An object variable that represents the Excel.Worksheet object to be filled data.

***CellList*** Required. A string that represents the list of cells separated by the "," character. For example, "A2,B2,B3,D2,D3". The cells in the *CellList* should correspond to the data source fields in the recordset. The value of the first field is put into the first cell, and the value of the second field is put into the second cell ……

***Range*** Optional. A string that indicates the range in the worksheet to be used for the details. XLReportCom will repeat the range for each record. You can reference a range of cells like "2:4" or "B2:D5". The default range is the area that includes all cells for the details.

***FillOrder*** Optional. An integer that indicates the order in which XLReportCom fills data. If the value is zero, XLReportCom will insert entire rows and fill data by rows. 1 means to insert entire columns and fill data by columns. 2 means to insert range and fill data by rows. 3 means to insert range and fill data by columns. Default is 0.

***ImageList*** Optional. A string that indicates which data source fields are the

picture files. The *ImageList* is the list of data source fields separated by the “,” character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the worksheet.

***PageBreak*** Optional. A string that indicates the page breaks. The unit of page length is r or g. "r" means record, "g1" means group one, "g2" means group two..... For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records, “1g1” or “1g” means a page break per group one, and “1g1,6r” means a page break per group one or 6 records. Default is “” that means no page break.

***NoData*** Optional. An integer that represents an option when no data are returned from data source. If the value is 1, XLReportCom will delete the range when no data are returned. If the value is 2, XLReportCom will delete the sheet when no data are returned. Default is 0. It means to do nothing.

***Group1…Group10*** Optional. A string that indicates the group that is the list of data source fields separated by the “,” character. You can identify a field using the name of field or the index number of field, but not simultaneously. In one report, there may be up to 10 groups. Notes: the order of groups should be in accordance with the order of ORDER BY clause in the SQL statement.

***GroupRange1…GroupRange10*** Optional. A string that indicates the range of the group in the worksheet. XLReportCom will repeat the range for each group. The range of the group should contain the range of the details and the area that includes all cells for this group. You reference a group range like “2:4” or “B2:D5”. For example, there are two groups, the range of the group one contains all cells for the group one and the range of the group two, and the range of the group two contains all cells for the group two and the range of the

details. The default range is the area that includes all cells for this group and the range or the group range for the lower level group.

**Example**

This example uses GroupTableReport method to make the report “Customer Profile”.

1. Create the template in Microsoft Excel.


2. Write the code in your application.

```
Set con = New ADODB.Connection
Set rec = New ADODB.Recordset
con.ConnectionString = "Data Source=Report Sample"
con.Open
strSQL = "SELECT LEFT(CompanyName,1), CompanyName,
ContactName, 'Phone: ' & Phone, 'Fax: ' & Fax, Address, CityName & ', ' &
CountryName, PostalCode FROM Customers, Cities, Countries WHERE
Customers.CityCode = Cities.CityCode AND Customers.CountryCode =
Cities.CountryCode AND Customers.CountryCode = Countries.CountryCode
ORDER BY CompanyName"
rec.Open strSQL, con
xlrpt.GroupTableReport Recordset:=rec, Worksheet:=xlWorksheet,
CellList:="A6,B7,C7,D7,D8,E7,E8,E9", Range:=” 6:9”, Group1:= “1”,
PageBreak:=”6r”
rec.Close
```

3. Generate the report.

| 5 | Customer Name | Contact Name | Phone/Fax | Address |
|---|---|---|---|---|
| 6 | A | | | |
| 7 | Alfreds Futterkiste | Maria Anders | Phone: 030-0074321 | Obere Str. 57 |
| 8 | | | Fax: 030-0076545 | Berlin, Germany |
| 9 | | | | 12209 |
| 10 | | | | |
| 11 | Ana Trujillo Emparedados y helados | Ana Trujillo | Phone: (5) 555-4729 | Avda. de la Constitución 2222 |
| 12 | | | Fax: (5) 555-3745 | México D.F., Mexico |
| 13 | | | | 5021 |
| 14 | B | | | |
| 15 | Berglunds snabbkop | Christina Berglund | Phone: 0921-12 34 65 | Berguvsvägen 8 |
| 16 | | | Fax: 0921-12 34 67 | Lule, Sweden |
| 17 | | | | S-958 22 |
| 18 | | | | |
| 19 | Blauer See Delikatessen | Hanna Moos | Phone: 0621-08460 | Forsterstr. 57 |
| 20 | | | Fax: 0621-08924 | Mannheim, Germany |
| 21 | | | | 68306 |

## 5.2.4 SetExcelName Method

Gets data from a recordset object, and assigns the values to the names defined in the Excel workbook. XLReportCom will just fetch the first record, no matter how many records are returned from data source.

**Syntax**

*object*.**SetExcelName(*Recordset, Workbook, NameList*)**

*object* *Required.* The object is the XLReport object.

***Recordset*** Required. An object variable that represents the ADODB.Recordset object to provides data.

***Worksheet*** Required. An object variable that represents the Excel.Worksheet object to be filled data.

***NameList*** Required. A string that represents the names you want assign values to. The *NameList* is the list of names separated by the “,” character. For example, “BeginDate, EndDate” means two names: BeginDate and EndDate that should be defined in the report template. The names in the list should correspond to the fields in the SQL statement. The value of the first field is put into the first name, and the value of the second field is put into the second

name …

**Example**

This example uses SetExcelName method to assign the values of fields to names.

1. Define the names BeginDate and EndDate in the report template in Microsoft Excel.
2. Write the code in your application.

```
Set con = New ADODB.Connection
Set rec = New ADODB.Recordset
con.ConnectionString = "Data Source=Report Sample"
con.Open
strSQL = " SELECT min_date, max_date FROM tmp0"
rec.Open strSQL, con
xlrpt.SetExcelName Recordset:=rec, Worksheet:=xlWorksheet, NameList:="BeginDate,EndDate"
rec.Close
```

## 5.2.5 ExcelReport Method

Generates the reports based on the templates and a XRF file. The XRF file tells XLReportCom how to get data from data sources and how to put data into the reports.

**Syntax**

*object*.**ExcelReport(*Application, XrfFile, Param1 … Param10*)**

*object* *Required.* The object is the XLReport object.

***Application*** Required. An object variable that represents the

Excel.Application object.

***XrfFile*** Required. A string that represents the XRF file. You can include a full path.

***Param1 … Param10*** Optional. A string that represents the paramters. These parameters have been defined in the XRF file.

**Example**

This example uses ExcelReport method to make the report “Customer List”.

1. Create the template customer_list.xls using Microsoft Excel.
2. Create the XRF file customer_list.xrf using a text editor.
3. Write the code in your application.

```
Set xlApp = New Excel.Application
Set xlrpt = New XLReport
Call xlrpt.ExcelReport(xlApp, “customer_list.xrf”)
```

## 5.3 Events

### 5.3.1 BeforeConnect Event

Occurs before a connection starts.

**Syntax**

**Private Sub** *object*_**BeforeConnect(***UserID* **As String,** *Password* **As String,** *DataSource* **As String,** *Connection* **As ADODB.Connection)**

*object* The object is the XLReport object.

*UserID* A string that represents a user name for the connection.

*Password* A string that represents a password for the connection.

*DataSource* A string that represents a data source name for the

connection.

*Connection* The ADODB.Connection object.

**Example**

```
Private Sub mxlrpt_BeforeConnect(UserID As String, Password As String,
DataSource As String, Connection As ADODB.Connection)
    Connection.ConnectionTimeout = 15
    Connection.CursorLocation = adUseClient
End Sub
```

## 5.3.2 TemplateOpen Event

Occurs when a template workbook is opened.

**Syntax**

**Private Sub** *object*_**TemplateOpen(ByVal** *Workbook* **As Excel.Workbook)**

*object* The object is the XLReport object.

*Workbook* An object variable that represents the Excel.Workbook object to be opened.

**Example**

```
Private Sub mxlrpt_TemplateOpen(ByVal Workbook As Excel.Workbook)
    Dim i As Integer

    With Workbook
        If .Application.Visible Then
            For i = .Worksheets.Count To 1 Step -1
                .Worksheets(i).DisplayPageBreaks = False
```

```
            Next i
        End If
    End With
End Sub
```

### 5.3.3 ReportComplete Event

Occurs when all report generating process is completed.

**Syntax**

**Private Sub** *object*_**ReportComplete(ByVal** *Workbook* **As Excel.Workbook)**

*object* The object is the XLReport object.

*Workbook* An object variable that represents the Excel.Workbook object.

**Example**

```
Private Sub mxlrpt_ReportComplete(ByVal Workbook As Excel.Workbook)
    ' Close the workbook and do not display the report when get errors
    If mintErrCount > 0 Then
        Workbook.Close
    End If
End Sub
```

### 5.3.4 FunctionBeforeExectue Event

Occurs before a function is executed.

**Syntax**

**Private Sub** *object*_**FunctionBeforeExectue(ByVal** *FunctionNo* **As String, ByVal** *FunctionType* **As Integer, ByVal** *SQLNo* **As Long, ByVal** *SQLText* **As**

**String)**

*object* The object is the XLReport object.

*FunctionNo* A string that represents the label of the function.

*FunctionTyp* An integer that represents the type of the function. 0 means ExecSQL function. 1 means Name function. 2 means Report function.

*SQLNo* A long that represents the number of SQL statements.

*SQLText* A string that contains the SQL statement.

**Example**

```
Private Sub mxlrpt_FunctionBeforeExectue(ByVal FunctionNo As String,
ByVal FunctionType As Integer, ByVal SQLNo As Long, ByVal SQLText As
String)
    frmWait.lblFunctionNo = FunctionNo
    frmWait.lblSQLCount = SQLNo
End Sub
```

## 5.3.5 FunctionAfterExectue Event

Occurs after a function is executed.

**Syntax**

**Private Sub** *object*_**FunctionAfterExectue(ByVal** *FunctionNo* **As String, ByVal** *FunctionType* **As Integer, ByVal** *SQLNo* **As Long, ByVal** *ErrObj* **As ErrObject)**

*object* The object is the XLReport object.

*FunctionNo* A string that represents the label of the function.

*FunctionTyp* An integer that represents the type of the function. 0 means

ExecSQL function. 1 means Name function. 2 means Report function.

*SQLNo* A long that represents the number of SQL statements.

*ErrObj* The Err object.

**Example**

```
Private Sub mxlrpt_FunctionAfterExectue(ByVal FunctionNo As String, ByVal
FunctionType As Integer, ByVal SQLNo As Long, ByVal ErrObj As ErrObject)
    If ErrObj.Number <> 0 Then
        If FunctionType <> 0 Then                'Ignore errors of EXECSQL
            MsgBox ErrObj.Description, vbExclamation, App.ProductName
            mintErrCount = mintErrCount + 1
        End If
    End If
End Sub
```

## 5.3.6 FunctionProgress Event

Occurs periodically during a function processing.

**Syntax**

**Private Sub** *object*_**FunctionProgress(ByVal** *Progress* **As Long, ByVal** *RecordCount* **As Long)**

*object* The object is the XLReport object.

*Progress* A long that indicates the number of records that have currently been processed.

*RecordCount* A long that indicates the total number of records.

**Example**

```
Private Sub mxlrpt_FunctionProgress(ByVal Progress As Long, ByVal
RecordCount As Long)
    frmWait.lblRecordCnt.Caption = Format(Progress, "#,##0") & " / " &
Format(RecordCount, "#,##0")
End Sub
```

## 5.4 Error Messages

The following table lists the trappable errors for the XLReport Object.

| Value | Description |
|---|---|
| -2147221493 | The file *XrfFileName* does not exist. |
| -2147221492 | The file *XrfFileName* is not an ExcelReport file. |
| -2147221491 | Error in reading the file *XrfFileName*. |
| -2147221490 | Report template file *TemplateFileName* does not exist. |
| -2147221489 | The report file is not named correctly. |
| -2147221488 | Failed to create the report file *ReportFileName*. |
| -2147221487 | Failed to open the template file *TemplateFileName*. |
| -2147221486 | Failed to save the report file. |
| -2147221485 | Failed to save the report file. Not support the file format: *FileFormat*. |
| -2147221473 | The ADODB.Recordset object is closed. |
| -2147221453 | Syntax error. The sheet *SheetName* does not exist. |
| -2147221451 | The Excel.Workbook object is not set. |
| -2147221450 | The Excel.Worksheet object is not set. |
| -2147221403 | Syntax error. There is a lack of the parameter "NAME". |
| -2147221393 | Syntax error. There is a lack of the parameter "CELL". |
| -2147221392 | Syntax error. It is not a valid cell "" for the parameter "CELL". |
| -2147221391 | Syntax error. Failed to parse cell list. |
| -2147221383 | The range or copyrange should be *Range*. |

| | |
|---|---|
| -2147221382 | Syntax error. Failed to parse range *Range*. |
| -2147221373 | Syntax error. Failed to parse image. Can not find field *ImageField* in the image list. |
| -2147221372 | Syntax error. Failed to parse image. |
| -2147221363 | Syntax error. Failed to parse group *Group*. Can not find field *GroupField*. |
| -2147221362 | Syntax error. Failed to parse group. |
| -2147221361 | The grouprange of group *N* should be *Range*. |
| -2147221360 | Syntax error. Failed to parse grouprange. |

# Chapter 6 XRF Files

## 6.1 Using XRF files

### 6.1.1 About XRF files

Like XLReportGen, XLReportCom also can read an XRF file to generate a report. The XRF file is a text file with an .xrf extension. It contains information such as the name of the report template file, the name of the report file, the name of the log file, data sources, parameters and functions. The XRF file tells XLReportCom how to get data from data sources and how to put data into a report. Using the XRF file, it will simplify your development.

### 6.1.2 Using an XRF file with XLReport Object

XLReport object provides the ExcelReport method to generate a report based on an XRF file. For example, you have created the XRF file “myreport.xrf” and the template file. In the XRF file, there are two parameters. The first parameter is the sales date “$SalesDate”, and the second is the category of the products “$Category”. You can call ExcelReport method to generate the report.

```
Set xlApp = New Excel.Application
Set xlrpt = New XLReport
Call xlrpt.ExcelReport(xlApp, “c:\excelreport\myreport.xrf”, “1996-05-01”,
“Dairy Products”)
```

XLReportCom will replace “$SalesDate” in SQL statements with “1996-05-01”, replace “$Category” with “Dairy Products”, and then submit SQL statements to data sources.

### 6.1.3 Using an XRF file in command line

In the XLReportCom, there is an executable file ExcelReport.exe that can read an XRF file to generate a report. It is the same as XLReportGen command line. For example, you have created the XRF file “myreport.xrf” and the template file. In the XRF file, there are two parameters. The first parameter is the sales date “$SalesDate”, and the second is the category of the products “$Category”. You can run ExcelReport.exe in command line mode as follows:

excelreport c:\excelreport\myreport.xrf 1996-05-01 “Dairy Products”

XLReportCom will replace “$SalesDate” in SQL statements with “1996-05-01”, replace “$Category” with “Dairy Products”, and then submit SQL statements to data sources.

### 6.1.4 Creating an XRF file

The XRF file is a text file. You can create and modify an XRF file in XLReportGen or a text editor.

Sometimes you want to make an XRF file programmatically. You can write a program to create an XRF file using C, perl or DOS shell, and then run XLReportCom to generate report. The two steps can be written into a batch file.

1. Write a program to make the XRF file as you need.
2. Write a batch file to call the program and ExcelReport.exe.

For example, you write a batch file runrpt.bat as follows. changexrf is an executable file that reads template.txt and output template.xrf. First runrpt.bat call changexrf to make the XRF file, and then call ExcelReport.exe to generate the report.

```
@echo off
if "%1"=="" goto usage
```

```
goto process
:usage
echo Usage: runrpt ReportDate
echo ReportDate    Date format 'YYYY-MM-DD'
goto :EOF
:process
changexrf %1 <"template.txt" >"template.xrf"
ExcelReport "template.xrf" %1
```

### 6.1.5 Using parameters

You can use parameters in the XRF file. You can pass values to XLReportCom when it processes an XRF file. XLReportCom will replace the parameter names with the actual values. You can use the parameters in the SQL statements and the paths and names of the files.

To use a parameter, you must define it first. If you have defined a parameter name, you can use it in SQL statements. In fact, XLReportCom will replace all strings that are the same as the names of the parameters. You should be careful to define a unique name for each parameter. It is a good choice a name begins with the “$” character.

**Example**

Input an order id to get the order information. The field OrderID is numeric type.

1. Defining a parameter

Define a parameter as follows:

Name: $OrderID

Title: Order ID (>=10248)

Default: 10360

2. Using a parameter

You can use the parameter “$OrderID” in SQL statements. For example:

```
SELECT o.OrderID
,o.OrderDate
,SUM(d.UnitPrice * d.Quantity * (1-d.Discount)) AS Amount
FROM Orders o, OrderDetails d
WHERE o.OrderID = d.OrderID
AND o.OrderID = $OrderID
GROUP BY o.OrderID, o.OrderDate
;
```

**Example**

Define two parameters. The first parameter is the sales date, and the second is the category of the products. The field OrderDate is date type, and CategoryName is char type.

1. Defining parameters

Define parameters as follows:

Name1: $SalesDate

Title1: Sales Date

Default1: 1996-05-01

Name2: $Category

Title2: Category of Products

Default2:

2. Using parameters

You can use the parameters “$SalesDate”, “$Category” in SQL statements. For example:

```
SELECT ……
FROM Orders, OrderDetails, Products, Categories
WHERE ……
AND OrderDate = ‘$SalesDate’
```

AND CategoryName LIKE '$Category%'

;

/* For Microsoft Jet SQL, LIKE '$Category*' */

**Example**

Get the information from the database, table and column that you identify when the report is generated.

1. Defining parameters

Define parameters as follows:

Name1: $Database

Title1: Database Name

Default1:

Name2: $Table

Title2: Table Name

Default2:

Name3: $Column

Title3: Column Name

Default3:

2. Using parameters

You can use the parameters “$Database”, “$Table” and “$Column” in SQL statements. For example:

USE $Database;

or

DATABASE $Database;

SELECT $Column

FROM $Table

;

**Example**

Use parameters in the path and name of the report file and the log file.

1. Defining a parameter

Define a parameter as follows:

Name: $CustomerID

Title: Customer ID

Default: C000001

2. Using a parameter

ReportFileName=report\report_$CustomerID.xls

LogFileName=log\report_$CustomerID.log

or

ReportFileName=report\$CustomerID\report.xls

LogFileName=log\$CustomerID\report.log

### 6.1.6 Converting files

You can convert a file from Microsoft Excel to another file format or from another file format to Microsoft Excel. For example, the template file is a Lotus 1-2-3 file with a .wk3 extension, and the report file is a HTML file with a .htm extension. For most file formats, Microsoft Excel converts only the active sheet. To convert the other sheets, open the template file, switch to the sheet you want to save, and save it.

The file formats XLReportCom supports can be one of these. What file format XLReportCom supports is dependent on your Microsoft Excel. For example, Microsoft Excel 2003 supports XML, but Microsoft Excel 97/2000 does not support it. For more information about converting files, please refer to *Microsoft Excel Help*. The file “xconv.cfg” is located in the XLReportCom directory contains the information of the file format. You can expand it if your Microsoft Excel supports more file formats.

| File Format Name | Value | Description | Extension | Converter |
|---|---|---|---|---|
| xlWorkbookNormal | -4143 | Microsoft Excel Workbook | Xls | Office97 |
| xlTemplate | 17 | Template | Xlt | Office97 |

| | | | | |
|---|---|---|---|---|
| XlAddIn | 18 | Microsoft Excel Add-In | xla | Office97 |
| XlHtml | 44 | Web Page | htm html | Office2000 |
| xlWebArchive | 45 | Single File Web Page | mht mhtml | Office2003 |
| xlXMLSpreadsheet | 46 | XML Spreadsheet | xml | Office2003 |
| XlCSV | 6 | CSV (comma delimited) | csv | Office97 |
| xlCSVMac | 22 | CSV (comma delimited) (Macintosh) | csv | Office97 |
| xlCSVMSDOS | 24 | CSV (comma delimited) (MS-DOS) | csv | Office97 |
| xlCSVWindows | 23 | CSV (comma delimited) (Windows) | csv | Office97 |
| xlCurrentPlatformText | -4158 | Text (Tab-delimited) | txt | Office97 |
| xlTextMac | 19 | Text (Tab-delimited) (Macintosh) | txt | Office97 |
| xlTextMSDOS | 21 | Text (Tab-delimited) (MS-DOS) | txt | Office97 |
| xlTextWindows | 20 | Text (Tab-delimited) (Windows) | txt | Office97 |
| xlTextPrinter | 36 | Formatted Text (Space-delimited) | prn | Office97 |
| xlUnicodeText | 42 | Unicode Text | txt | Office2000 |
| xlExcel2 | 16 | Microsoft Excel 2.0 Worksheet | xls | Office97 |
| xlExcel2FarEast | 27 | Microsoft Excel 2.0 Worksheet Far East | xls | Office97 |
| xlExcel3 | 29 | Microsoft Excel 3.0 Worksheet | xls | Office97 |
| xlExcel4 | 33 | Microsoft Excel 4.0 Worksheet | xls | Office97 |
| xlExcel4Workbook | 35 | Microsoft Excel 4.0 Workbook | xlw | Office97 |
| xlExcel5 | 39 | Microsoft Excel 5.0/95 Workbook | xlw | Office97 |
| xlExcel9795 | 43 | Microsoft Excel 97-2003 & 5.0/95 Workbook | xls | Office2000 |
| XlDBF2 | 7 | DBF 2 (dBASE II) | dbf | Office97 |
| XlDBF3 | 8 | DBF 3 (dBASE III) | dbf | Office97 |
| XlDBF4 | 11 | DBF 4 (dBASE IV) | dbf | Office97 |
| XlDIF | 9 | DIF (data interchange format) | dif | Office97 |
| XlSYLK | 2 | SYLK (symbolic link format) | slk | Office97 |

| | | | | |
|---|---|---|---|---|
| XlWJ2WD1 | 14 | WD1 (1-2-3) | wd1 | Office97 |
| XlWK1 | 5 | WK1 (1-2-3) | wk1 | Office97 |
| XlWK1ALL | 31 | WK1, ALL (1-2-3) | wk1 | Office97 |
| XlWK1FMT | 30 | WK1, FMT (1-2-3) | wk1 | Office97 |
| XlWK3 | 15 | WK3 (1-2-3) | wk3 | Office97 |
| XlWK3FM3 | 32 | WK3, FM3 (1-2-3) | wk3 | Office97 |
| XlWK4 | 38 | WK4 (1-2-3) | wk4 | Office97 |
| XlWKS | 4 | WKS (Works) | wks | Office97 |
| XlWorks2FarEast | 28 | Works Far East | wks | Office97 |
| XlWQ1 | 34 | WQ1 (Quattro Pro/DOS) | wq1 | Office97 |

For Microsoft Excel 2007, please copy “xconv2007.cfg” to “xconv.cfg”. This file contains the information of file formats for Microsoft Excel 2007.

| File Format Name | Value | Description | Extension |
|---|---|---|---|
| xlOpenXMLWorkbook | 51 | Excel Workbook | xlsx |
| xlOpenXMLWorkbookMacroEnabled | 52 | Excel Macro-enabled Workbook | xlsm |
| xlExcel12 | 50 | Excel Binary Workbook | xlsb |
| xlExcel8 | 56 | Excel 97-2003 Workbook | xls |
| xlWorkbookNormal | -4143 | Excel 97-2003 WorkbookNormal | xls |
| xlOpenXMLTemplateMacroEnabled | 53 | Excel Macro-enabled Workbook Template | xltm |
| xlOpenXMLTemplate | 54 | Excel Template | xltx |
| xlTemplate | 17 | Excel 97-2003 Template | xlt |
| xlOpenXMLAddIn | 55 | Excel Add-in | xlam |
| XlAddIn | 18 | Excel 97-2003 Add-In | xla |
| XlHtml | 44 | Web Page | htm html |
| xlWebArchive | 45 | Single File Web Page | mht mhtml |
| xlXMLSpreadsheet | 46 | XML Spreadsheet | xml |
| xlCSV | 6 | CSV (comma delimited) | csv |
| xlCSVMac | 22 | CSV (comma delimited) (Macintosh) | csv |
| xlCSVMSDOS | 24 | CSV (comma delimited) (MS-DOS) | csv |
| xlCSVWindows | 23 | CSV (comma delimited) (Windows) | csv |
| xlCurrentPlatformText | -4158 | Text (Tab-delimited) | txt |
| xlTextMac | 19 | Text (Tab-delimited) (Macintosh) | txt |
| xlTextMSDOS | 21 | Text (Tab-delimited) | txt |

| | | | |
|---|---|---|---|
| | | (MS-DOS) | |
| xlTextWindows | 20 | Text (Tab-delimited) (Windows) | txt |
| xlTextPrinter | 36 | Formatted Text (Space-delimited) | prn |
| xlUnicodeText | 42 | Unicode Text | txt |
| xlExcel5 | 39 | Microsoft Excel 5.0/95 Workbook | xlw |
| XlDIF | 9 | DIF (data interchange format) | dif |
| xlSYLK | 2 | SYLK (symbolic link format) | slk |

Note: Some of these file formats may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

## 6.2 XRF File Reference

### 6.2.1 XRF File Format

The layout of an XRF file is as the following:

ExcelReport Version 2.0

[Data Source]

……

[File]

……

[Parameter]

……

[SQL]

……

"ExcelReport" is the flag of the XRF file. "Version 2.0" is the version of the XRF file.

An XRF file contains several sections. The sections of [Data Source], [File], and [Parameter] consist of a group of related settings. The sections and

settings are listed in the XRF file in the following format:

[section name]

keyname=value

In this example, [section name] is the name of a section. The enclosing brackets ([]) are required, and the left bracket must be in the leftmost column on the screen.

The keyname=value statement defines the value of each setting. A keyname is the name of a setting. It can consist of any combination of letters and digits, and must be followed immediately by an equal sign (=). The value can be an integer, a string, or a quoted string, depending on the setting.

You can include comments in these sections. You must begin each line of a comment with a semicolon (;).

The [SQL] section consists of functions. Each function is begin with the “@” character. Syntax:

*@functionno=functionname(arguments)*

*sqlstatement*

The *functionno* is the label of the function.

The *functionname* represents a function.

The *arguments* define various properties for the function. An argument takes the form *Name="Value"*. The argument value can be delimited by single or double quotes.

The *sqlstatement* is a SQL statement.

You can use comments in [SQL] section. A comment is the “/*” characters, followed by any sequence of characters (including new lines), followed by the “*/” characters. You cannot nest comments.

### 6.2.2 [Data Source] Section

The [Data Source] section contains information how to connect to data

sources.

*Name1=<name1>*

*Name2=<name2>*

*……*

*Name10=<name10>*

These settings specify the names of data sources you want to connect to. Name1 specifies the name of the first data source. Name2 specifies the name of the second data source…… You can define up to 10 data sources in one XRF file. You can make a connection to a data source using an ODBC data source name or a connection string. Even if you use a connection string to make a connection, you should define a name that you can reference in functions.

*User1=<username1>*

*User2=<username2>*

*……*

*User10=<username10>*

These settings specify the user names. If you use an ODBC data source name to make a connection, you should define user name and password. If you use a connection string to make a connection, XLReportCom will ignore the setting. User1 specifies the user name of the first data source. User2 specifies the user name of the second data source…… They are optional settings. If defined default user and password in ODBC data source, you may not define them.

*Password1=<password1>*

*Password2=<password2>*

*……*

*Password10=<password10>*

These settings specify the user passwords. If you use an ODBC data source name to make a connection, you should define user name and password. If you use a connection string to make a connection, XLReportCom will ignore the setting. Password1 specifies the password of the first data source. Password2 specifies the password of the second data source…… They are optional settings. If defined default user and password in ODBC data source, you may not define them.

*ConnectionString1=<connectionstring1>*

*ConnectionString2=<connectionstring2>*

*……*

*ConnectionString10=<connectionstring10>*

These settings specify the connection strings. If you defined a connection string, XLReportCom will make a connection to the data source using the connection string, and ignore the settings of the name, user and password. But you must define a name that you can reference in functions. ConnectionString1 specifies the connection string of the first data source. ConnectionString2 specifies the connection string of the second data source…… They are optional settings. If no connection string, XLReportCom will make a connection to data source using the ODBC data source name.

*EncryptPassword =Y/N*

This setting specifies how to save the passwords of the data sources. If the value is Y, passwords will be saved in an encrypted format. If the value is N, the passwords will be saved in plain text.

### 6.2.3 [FILE] Section

[FILE] section contains information about files.

*ReportTemplateFileName=<templatefilename>*

This setting specifies the name of the report template file. <templatefilename> value is the name and path of the report template file. The file path can be a relative path or an absolute path. If it is a relative path, the base path is the path of the XRF file.

*ReportFileName=<reportfilename>*

This setting specifies the name of the report file. <reportfilename> value is the name and path of the report file. The file path can be a relative path or an absolute path. If it is a relative path, the base path is the path of the XRF file. In <reportfilename>, you can use parameters.

*ReportFileType=<reportfiletype>*

This setting specifies the type of the report file. <reportfiletype> value is the name or value of the file format. For example, xlCSV or 6. What file format XLReportCom supports is dependent on your Microsoft Excel.

*ProtectReport=Y/N*

This setting specifies whether the report generated is protected. If the value is Y, the report is protected, and can not be modified. If the value is N, the report is not protected. Default is N.

*ProtectionPassword=<protectionpassword>*

This setting specifies the password that is used to protect the report.

<protectionpassword> value is the password. This setting is valid when ProtectReport is Y. If there is not this setting and ProtectReport is Y, a random password will be created.

*LogFileName=<logfilename>*

This setting specifies the name of the log file. <logfilename> value is the name and path of the log file. The file path can be a relative path or an absolute path. If it is a relative path, the base path is the path of the XRF file. In <logfilename>, you can use parameters.

### 6.2.4 [PARAMETER] Section

[PARAMETER] section contains information about parameters.

*Name1=<name1>*

*Name2=<name2>*

*……*

*Name10=<name10>*

These settings specify the names of the parameters. Name1 specify the name of the first parameter, Name2 specifies the name of the second parameter…... You can define up to 10 parameters in one XRF file.

*Title1=<title1>*

*Title2=<title2>*

*……*

*Title10=<title10>*

These settings specify the titles of the parameters. Title1 specifies the title of the first parameter. Title2 specifies the title of the second parameter……

*Default1=<default1>*

*Default2=<default2>*

*……*

*Default10=<default10>*

These settings specify the default values of the parameters. Default1 specifies the default value of the first parameter. Default2 specifies the default value of the second parameter…...

## 6.3 Function Reference

### 6.3.1 Fixed Table Report

Uses FixTableReport method to generate a fixed table report. In a fixed table report, the number of rows and columns is fixed. XLReportCom executes a SQL statement to get data from data source, and directly fills data into the cells of a worksheet.

**Syntax**

*Report(…)*

*sqlstatement*

**Arguments**

TYPE = “fix”

SHEET = *sheet*

FILLORDER = *fillorder*

CELL= *celllist*

RANGE = *range*

IMAGE = *fieldlist*

PAGEBREAK = *pagelength*

CONNECT = *datasource*

The **TYPE** argument specifies the report type. "fix" means a fixed table report.
The **SHEET** argument identifies a worksheet in the report template. The *sheet* is the name or index number of the worksheet. The index number starts at 1.
The **FILLORDER** argument specifies the order in which XLReportCom fills data. Possible values are row or col. "row" means to fill data by rows, and "col" means to fill data by columns. Default is row.
The **CELL** argument specifies the positions where data values will be inserted. The *celllist* is the list of cells separated by the “,” character. For example, “A2,B2,B3,D2,D3”. The cells in the *celllist* should correspond to the data source fields in the SQL statement. The value of the first field is put into the first cell, and the value of the second field is put into the second cell …… XLReportCom will use the next cell if you omit a cell except the first cell. If FILLORDER=“row”, the next cell is the right cell. If FILLORDER=“col”, the next cell is the below cell.
The **RANGE** or **COPYRANGE** argument specifies the range in the worksheet to be used for the records. XLReportCom will skip or repeat the range for each record. You can reference a range of cells like “2:4” or “B2:D5”. The default range is the area that includes all cells for the records. For RANGE argument, XLReportCom will skip the rows/columns of the range for each record. For COPYRANGE argument, it will copy the original range to the range where data will be filled for each record.
The **IMAGE** argument specifies the data source fields are picture files. The *fieldlist* is the list of data source fields separated by the “,” character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the report template file.
The **PAGEBREAK** argument specifies the page breaks. The unit of page

length is r that means record. For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records. Default is no page break.

The **CONNECT** argument specifies the connection to a data source. The CONNECT can takes a string that expresses a data source name or a number that expresses a data source index. The index number of data source is the sequential number defined in the XRF file, and starts at 1. The default implies the first data source.

The ***sqlstatement*** is a SQL statement such as a SELECT statement.

**Example**

This example uses Fixed Table Report function to make the report “Top 5 Employees for Sales”.

```
@F1=REPORT(sheet="Report6" type=fix cell=B7)
SELECT TOP 5 e.FirstName + ' ' + e.LastName
     , SUM(d.Quantity)
     , Sum(d.UnitPrice * d.Quantity * (1-d.Discount)) AS SalesAmount
FROM Orders o
     ,OrderDetails d
     ,Products p
     ,Employees e
WHERE o.OrderID = d.OrderID
AND d.ProductID = p.ProductID
AND o.EmployeeID = e.EmployeeID
AND YEAR(o.OrderDate) = 1996
AND MONTH(o.OrderDate) = 04
GROUP BY e.FirstName, e.LastName
ORDER BY 3 DESC
```

### 6.3.2 Variable Table Report

Uses VarTableReport method to generate a variable table report. In a variable table report, the number of rows or columns in the table is unfixed, and it is variable as the number of the result records. XLReportCom executes a SQL statement to get data from data source, inserts some blank rows/columns or copy a range for each record, then fills data into the cells of a worksheet.

**Syntax**

*Report(…)*

*sqlstatement*

**Arguments**

TYPE = “var”

SHEET = *sheet*

FILLORDER = *fillorder*

CELL= *celllist*

RANGE = *range*

IMAGE = *fieldlist*

RESERVE = *reserverecords*

PAGEBREAK = *pagelength*

NODATA = *nodataoption*

CONNECT = *datasource*

The **TYPE** argument specifies the report type. "var" means a variable table report. Default is var.

The **SHEET** argument identifies a worksheet in the report template. The *sheet* is the name or index number of the worksheet. The index number starts at 1.

The **FILLORDER** argument specifies the order in which XLReportGen fills data.

Possible values are row, col, rowrange or colrange. "row" means to insert entire rows and fill data by rows. "col" means to insert entire columns and fill data by columns. "rowrange" means to insert range and fill data by rows. "colrange" means to insert range and fill data by columns. Default is row.

The **CELL** argument specifies the positions where data values will be inserted. The *celllist* is the list of cells separated by the “,” character. For example, “A2,B2,B3,D2,D3”. The cells in the *celllist* should correspond to the data source fields in the SQL statement. The value of the first field is put into the first cell, and the value of the second field is put into the second cell …… XLReportCom will use the next cell if you omit a cell except the first cell. If FILLORDER=“row”, the next cell is the right cell. If FILLORDER=“col”, the next cell is the below cell.

The **RANGE** or **COPYRANGE** argument specifies the range in the worksheet to be used for the records. XLReportCom will skip or repeat the range for each record. You can reference a range of cells like “2:4” or “B2:D5”. The default range is the area that includes all cells for the records. For RANGE argument, XLReportCom will insert the blank rows/columns of the range for each record. For COPYRANGE argument, it will copy the original range and insert the copied range for each record.

The **IMAGE** argument specifies the fields are picture files. The *fieldlist* is the list of data source fields separated by the “,” character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the report template file.

The **RESERVE** argument specifies the number of the records for which you reserved some rows/columns in the report template for the report. The *reserverecords* represents the number of the records you reserved in the

report template. Possible values are 1 or 2. One means you reserved some rows/columns for one record, and two means some rows/columns for two records. Default is 1.

The **PAGEBREAK** argument specifies the page breaks. The unit of page length is r that means record. For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records. Default is no page break.

The **NODATA** argument specifies an option when no data are returned from data source. If the value is "delrange", XLReportGen will delete the range when no data are returned. If the value is "delsheet", XLReportGen will delete the sheet when no data are returned. Default is to do nothing.

The **CONNECT** argument specifies the connection to a data source. The CONNECT can takes a string that expresses a data source name or a number that expresses a data source index. The index number of data source is the sequential number defined in the XRF file, and starts at 1. The default implies the first data source.

The ***sqlstatement*** is a SQL statement such as a SELECT statement.

**Example**

This example uses Variable Table Report function to make the report “Mail Label”.

```
@F1=Report(sheet="Mail Label" type=var cell=B7,B8,B9,B10 copyrange=1:11
pagebreak = 4r)
SELECT CompanyName
,Address
,CityName & ', ' & CountryName
,PostalCode
FROM Customers, Cities, Countries
WHERE Customers.CityCode = Cities.CityCode
```

AND Customers.CountryCode = Cities.CountryCode

AND Customers.CountryCode = Countries.CountryCode

ORDER BY CompanyName

### 6.3.3 Group Table Report

Uses GroupTableReport method to generate a variable table report and group data. In a variable table report, the number of rows or columns in the table is unfixed, and it is variable as the number of the result records. XLReportCom executes a SQL statement to get data from data source, copy the group range for each group, copy the detail range for each record, then fills data into the worksheet.

**Syntax**

*Report(…)*

*sqlstatement*

**Arguments**

TYPE = “var”

SHEET = *sheet*

FILLORDER = *fillorder*

CELL= *celllist*

RANGE = *range*

GROUP= *grouplist*

GROUPRANGE = *grouprange*

IMAGE = *fieldlist*

PAGEBREAK = *pagelength*

NODATA = *nodataoption*

CONNECT = *datasource*

The **TYPE** argument specifies the report type. "var" means a variable table report. Default is var.

The **SHEET** argument identifies a worksheet in the report template. The *sheet* is the name or index number of the worksheet. The index number starts at 1.
The **FILLORDER** argument specifies the order in which XLReportGen fills data. Possible values are row, col, rowrange or colrange. "row" means to insert entire rows and fill data by rows. "col" means to insert entire columns and fill data by columns. "rowrange" means to insert range and fill data by rows. "colrange" means to insert range and fill data by columns. Default is row.
The **CELL** argument specifies the positions where data values will be inserted. The *celllist* is the list of cells separated by the “,” character. For example, “A2,B2,B3,D2,D3”. The cells in the *celllist* should correspond to the data source fields in the SQL statement. The value of the first field is put into the first cell, and the value of the second field is put into the second cell …… XLReportCom will use the next cell if you omit a cell except the first cell. If FILLORDER=“row”, the next cell is the right cell. If FILLORDER=“col”, the next cell is the below cell.
The **RANGE** or **COPYRANGE** argument specifies the range in the worksheet to be used for the details. XLReportCom will skip or repeat the range for each record. You can reference a range of cells like “2:4” or “B2:D5”. The default range is the area that includes all cells for the details. For RANGE argument, XLReportCom will insert the blank rows/columns of the range for each record. For COPYRANGE argument, it will copy the original range and insert the copied range for each record. But if the range of any group is not same as the range of the details, RANGE is same as COPYRANGE.
The **GROUP** argument specifies the group of the report. The *grouplist* is the list of data source fields separated by the “,” character. You can identify a field using the name or index number of the field, but not simultaneously. In one report, there may be up to 10 groups. Notes: the order of the groups should be in accordance with the order of the ORDER BY clause in the SQL statement.

The **GROUPRANGE** argument follows the GROUP argument, and specifies the range of the group in the worksheet. For example, the grouprange of level 1 must follow the group of level 1, and the grouprange of level 2 must follow the group of level 2. XLReportCom will repeat the group range for each group. The range of the group should contain the range of the details and the area that includes all cells for this group. You reference a group range like “2:4” or “B2:D5”. For example, there are two groups, the range of the group one contains all cells for the group one and the range of the group two, and the range of the group two contains all cells for the group two and the range of the details. The default range is the area that includes all cells for this group and the range or group range for the lower level group.

The **IMAGE** argument specifies the fields are picture files. The *fieldlist* is the list of data source fields separated by the “,” character. You can identify a field using the name of field or the index number of field, but not simultaneously. In data source, you stored the path and file name of the picture, not the picture. The file path can be a relative path, an absolute path or a URL. If it is a relative path, the base path is the path of the report template file.

The **PAGEBREAK** argument specifies the page breaks. The unit of page length is r or g. "r" means record, "g1" means group one, "g2" means group two..... For example, “6r” or “6” means that XLReportCom will insert a page break per 6 records, “1g1” or “1g” means a page break per group one, and “1g1,6r” means a page break per group one or 6 records. Default is “” that means no page break.

The **NODATA** argument specifies an option when no data are returned from data source. If the value is "delrange", XLReportGen will delete the range when no data are returned. If the value is "delsheet", XLReportGen will delete the sheet when no data are returned. Default is to do nothing.

The **CONNECT** argument specifies the connection to a data source. The

CONNECT can takes a string that expresses a data source name or a number that expresses a data source index. The index number of data source is the sequential number defined in the XRF file, and starts at 1. The default implies the first data source.

The ***sqlstatement*** is a SQL statement such as a SELECT statement.

**Example**

This example uses Group Table Report function to make the report “Customer Profile”.

```
@F1=Report(sheet="Customer Profile" cell=A6,B7,C7,D7,D8,E7,E8,E9
range=6:9 group=1 pagebreak = 6r)
SELECT LEFT(CompanyName,1)
,CompanyName
,ContactName
,'Phone: ' & Phone
,'Fax: ' & Fax
,Address
,CityName & ', ' & CountryName
,PostalCode
FROM Customers, Cities, Countries
WHERE Customers.CityCode = Cities.CityCode
AND Customers.CountryCode = Cities.CountryCode
AND Customers.CountryCode = Countries.CountryCode
ORDER BY CompanyName
```

## 6.3.4 Name

Executes a SQL statement, and assigns the values to the names defined in the Excel workbook. XLReportCom will just fetch the first record, no matter how

many records are returned from data source.

**Syntax**

Name(…)

*sqlstatement*

**Arguments**

NAME= *namelist*

CONNECT= *datasource*

The **NAME** argument specifies the names you want assign values to. The *namelist* is the list of names separated by the “,” character. For example, “BeginDate, EndDate” means two names: BeginDate and EndDate that should be defined in the report template. The names in the list should correspond to the fields in the SQL statement. The value of the first field is put into the first name, and the value of the second field is put into the second name …

The **CONNECT** argument specifies the connection to a data source. The CONNECT can takes a string that expresses a data source name or a number that expresses a data source index. The index number of data source is the sequential number defined in the XRF file, and starts at 1. The default implies the first data source.

The ***sqlstatement*** is a SQL statement such as a SELECT statement.

**Example**

This example uses Name function to assign the values of min_date and max_date to the names: BeginDate and EndDate.

```
@F1=NAME(NAME=BeginDate,EndDate)
SELECT min_date, max_date
FROM tmp0
```

### 6.3.5 ExecSQL

Executes a SQL statement, but no data is returned to the report.

**Syntax**

ExecSQL*(…)*

*sqlstatement*

**Arguments**

CONNECT= *datasource*

The **CONNECT** argument specifies the connection to a data source. The CONNECT can takes a string that expresses a data source name or a number that expresses a data source index. The index number of data source is the sequential number defined in the XRF file, and starts at 1. The default implies the first data source.

The ***sqlstatement*** is a SQL statement that can be DDL (Data Definition Language), DML (Data Manipulation Language) and even DCL (Data Control Language).

Using EXECSQL function, you can open a database, create a temporary table, insert data into a temporary table, update data, execute a stored procedure, and drop a table. It is very useful to create a temporary table, and prepare data for REPORT function.

**Example**

This example uses ExecSQL functions to create a table tmp0, and add some records into the table. No result is returned to the report.

```
@F1=EXECSQL()
CREATE TABLE tmp0 (
min_date DATE,
max_date DATE)
;
```

```
@F2=EXECSQL()
INSERT INTO tmp0
SELECT …
```

# Chapter 7 Advanced Reports

## 7.1 Executing multiple SQL statements

In one report building process, XLReportCom can execute multiple SQL statements. This enables you to

1. Create a report like building block. You may divide one report into several parts, and respectively use the different SQL statements to make each part of the report. You can use the different queries to get the data located in the different tables or databases.

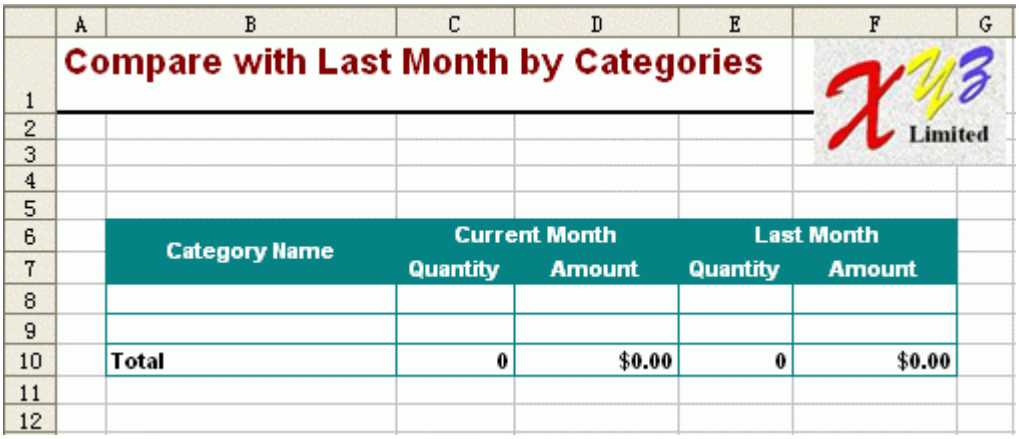
2. Create a complex report using the temporary table. First, you create a temporary table. Second, use several SQL statements to prepare data in the temporary table. You can execute INSERT, UPDATE, DELETE, INSERT SELECT statements. And then put the prepared data from the temporary table into your report.

3. Create one report file with several reports. For example, you may create one workbook with several worksheets.

**Example**

This example executes multiple SQL statements to create a report.

1. Create the template in Microsoft Excel.



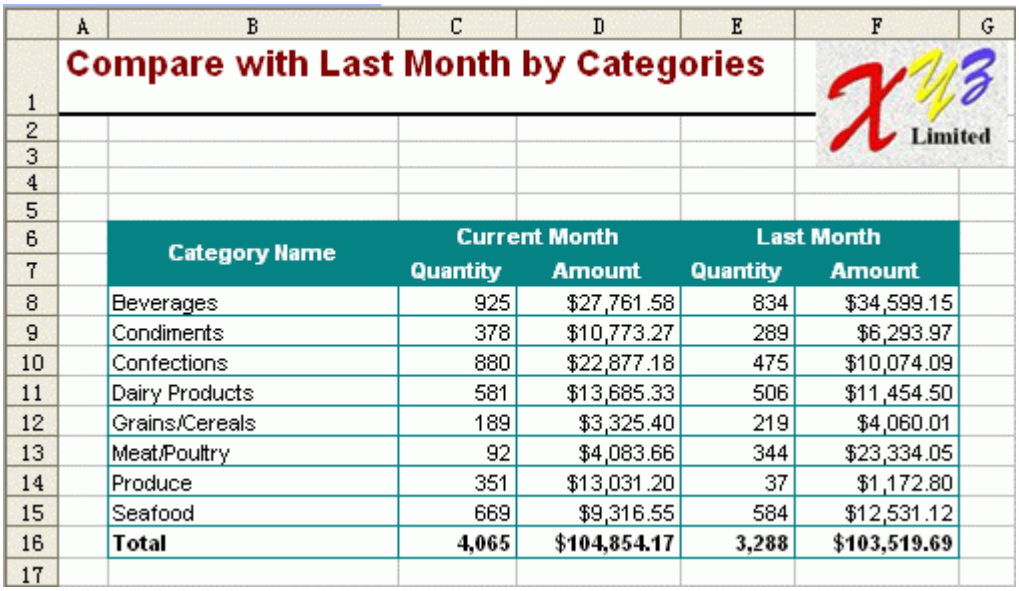
2. Write SQL statements in an XRF file.

```
/*******************************************************
Compare with Last Month by Categories
*******************************************************/
/* Drop table tmp_category_sales */
@F9_1=EXECSQL()
DROP TABLE tmp_category_sales

/* Create table tmp_category_sales */
@F9_2=EXECSQL()
CREATE TABLE tmp_category_sales (
CategoryID INTEGER,
Quantity INTEGER,
Amount MONEY
)

/* Get the sales amount by categories in the current month */
@F9_3=EXECSQL()
INSERT INTO tmp_category_sales (CategoryID, Quantity, Amount)
SELECT p.CategoryID, SUM(d.Quantity), Sum(d.UnitPrice * d.Quantity *
(1-d.Discount))
FROM Orders o
     ,OrderDetails d
     ,Products p
WHERE o.OrderID = d.OrderID
AND d.ProductID = p.ProductID
AND YEAR(o.OrderDate) = YEAR('1996-04-01')
```

```
AND MONTH(o.OrderDate) = MONTH('1996-04-01')
GROUP BY p.CategoryID

/* Show the sales amount by categories in the current month */
@F9_4=REPORT(sheet="Report9" type=var cell=B8 reserve=2)
SELECT c.CategoryName, IIF(IsNull(t.Quantity),0,t.Quantity),
IIF(IsNull(t.Amount),0,t.Amount)
FROM Categories c LEFT JOIN tmp_category_sales t
ON c.CategoryID = t.CategoryID
ORDER BY c.CategoryName

/* Delete from table tmp_category_sales */
@F9_5=EXECSQL()
DELETE FROM tmp_category_sales

/* Get the sales amount by categories in the last month */
@F9_6=EXECSQL()
INSERT INTO tmp_category_sales (CategoryID, Quantity, Amount)
SELECT p.CategoryID, SUM(d.Quantity), Sum(d.UnitPrice * d.Quantity *
(1-d.Discount))
FROM Orders o
     ,OrderDetails d
     ,Products p
WHERE o.OrderID = d.OrderID
AND d.ProductID = p.ProductID
AND o.OrderDate >= DateAdd('m',-1,#1996-04-01#)
AND o.OrderDate < #1996-04-01#
GROUP BY p.CategoryID
```

```
/* Show the sales amount by categories in the last month */
@F9_7=REPORT(sheet="Report9" type=fix cell=E8)
SELECT IIF(IsNull(t.Quantity),0,t.Quantity), IIF(IsNull(t.Amount),0,t.Amount)
FROM Categories c LEFT JOIN tmp_category_sales t
ON c.CategoryID = t.CategoryID
ORDER BY c.CategoryName
```

3. Generate the report.



**Compare with Last Month by Categories**

XYZ Limited

| Category Name | Current Month | | Last Month | |
|---|---|---|---|---|
| | Quantity | Amount | Quantity | Amount |
| Beverages | 925 | $27,761.58 | 834 | $34,599.15 |
| Condiments | 378 | $10,773.27 | 289 | $6,293.97 |
| Confections | 880 | $22,877.18 | 475 | $10,074.09 |
| Dairy Products | 581 | $13,685.33 | 506 | $11,454.50 |
| Grains/Cereals | 189 | $3,325.40 | 219 | $4,060.01 |
| Meat/Poultry | 92 | $4,083.66 | 344 | $23,334.05 |
| Produce | 351 | $13,031.20 | 37 | $1,172.80 |
| Seafood | 669 | $9,316.55 | 584 | $12,531.12 |
| **Total** | **4,065** | **$104,854.17** | **3,288** | **$103,519.69** |

## 7.2 Using Excel Formulas

Formulas are equations that perform calculations on values in your worksheet. A formula starts with an equal sign (=). For example, the following formula multiplies 2 by 3 and then adds 5 to the result.

```
=5+2*3
```

A formula can also contain any or all of the following: functions, references, operators, and constants. For more detail information about formulas, functions and references, refer to *Microsoft Excel Help*.

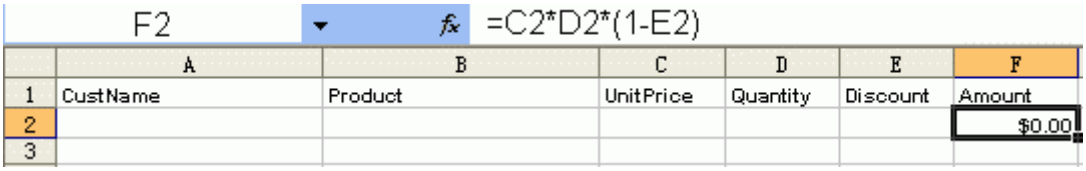
In a report template file, you can use all kind of Microsoft Excel formulas. And then all formulas in the report template file will be brought to the final report file.

**Example**

Show Unit Price, Quantity, Discount and Amount. The Amount will be changed if an end user changes Unit Price, Quantity or Discount.

You can use a formula to show Amount.

1. Create a template file as follows, and define the formula "=C2*D2*(1-E2)" in cell F2. You must use the relative reference.




2. Write the report function as follow, and use COPYRANGE to copy the formula to all following cells for each record. For the first record, XLReportCom will directly put data into row 2. For the other records, it will copy row 2 to the current row, and then put data into the current row. So the formula in cell F2 will copy to cell F3, F4… and Microsoft Excel will automatically change the formula to "=C3*D3*(1-E3)" …

```
@F1=Report(sheet="Sheet1" cell=A2 copyrange=2:2)
SELECT c.CompanyName AS Customer
,p.ProductName
,d.Quantity
,d.UnitPrice
,d.Discount
FROM Orders o
, Customers c
, OrderDetails d
, Products p
WHERE o.CustomerID = c.CustomerID
```

AND o.OrderID = d.OrderID

AND d.ProductID = p.ProductID

AND YEAR(o.OrderDate) = YEAR('1996-04-01')

AND MONTH(o.OrderDate) = MONTH('1996-04-01')

ORDER BY 1, 2

3. Generate the report.

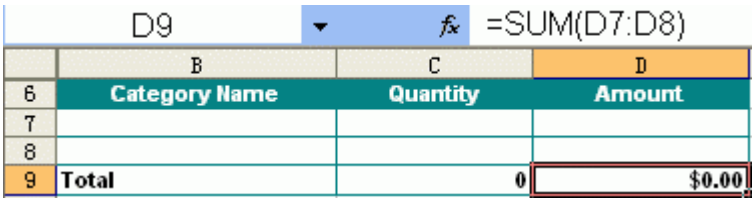
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | CustName | Product | UnitPrice | Quantity | Discount | Amount |
| 2 | Bon app' | Carnarvon Tigers | $30.00 | 50 | 0% | $1,500.00 |
| 3 | Bon app' | Tunnbröd | $15.00 | 7 | 0% | $108.00 |
| 4 | Bon app' | Wimmers gute Semmelknöde | $8.00 | 27 | 0% | $212.80 |
| 5 | B's Beverages | Boston Crab Meat | $10.00 | 15 | 0% | $147.00 |
| 6 | B's Beverages | Gnocchi di nonna Alice | $20.00 | 30 | 0% | $608.00 |

**Example**

Add totals such as Total Quantity, Total Amount.

You can use the math functions of Microsoft Excel, such as SUM.

1. Create a report template file as follows, define the formula of total quantity as "=SUM(C7:C8)" in cell C9, and the formula of total amount as "=SUM(D7:D8)" in cell D9. You must use the relative reference.




2. Write the report function as follow. When XLReportCom insert some rows according to the records, Microsoft Excel will automatically change the formulas.

@F2=REPORT(sheet="Report2" type=var cell=B7 reserve=2)

SELECT c.CategoryName, SUM(d.Quantity), Sum(d.UnitPrice * d.Quantity * (1-d.Discount))

FROM Orders o

,OrderDetails d

,Products p

```
      ,Categories c
WHERE o.OrderID = d.OrderID
AND d.ProductID = p.ProductID
AND p.CategoryID = c.CategoryID
AND YEAR(o.OrderDate) = YEAR('1996-04-01')
AND MONTH(o.OrderDate) = MONTH('1996-04-01')
GROUP BY c.CategoryName
ORDER BY c.CategoryName
;
```

3. The following is the generated report. The formula of total quantity is changed to “=SUM(C7:C14)”, and the formula of total amount is changed to “=SUM(D7:D14)”.

| Category Name | Quantity | Amount |
|---|---|---|
| Beverages | 925 | $27,761.57 |
| Condiments | 378 | $10,773.27 |
| Confections | 880 | $22,877.18 |
| Dairy Products | 581 | $13,685.32 |
| Grains/Cereals | 189 | $3,325.40 |
| Meat/Poultry | 92 | $4,083.66 |
| Produce | 351 | $13,031.20 |
| Seafood | 669 | $9,316.54 |
| **Total** | **4,065** | **$104,854.15** |

## 7.3 Sorting, Grouping and Totaling

### 7.3.1 Sorting data

Sorting means placing data in some kind of order to help you find and evaluate it. For example, you may want to have a customer list sorted alphabetically by name or by country.

To sort your data, you may use SQL. Use the **ORDER BY** clause to have your results displayed in a sorted order.

```
SELECT EmployeeID
,LastName
```

```
,FirstName
,HireDate
FROM Employees
ORDER BY HireDate;   /* ascending sort */
```

In the example above, results will come back in ascending order by hire date. To explicitly specify ascending or descending order, add ASC or DESC, to the end of your ORDER BY clause. The following is an example of a descending order sort.

```
ORDER BY HireDate DESC;     /* descending sort */
```

## 7.3.2 Totaling

You can sum the values, count all the values or only those values that are distinct from one another, and determine the maximum, minimum, average. To add totals, there are two ways.

1. You can add the totals using the math functions of Microsoft Excel, such as SUM. For more detail information, refer to “Using Excel Formulas” in this document.

2. You can use the aggregate functions in SQL statement, such as COUNT, SUM, AVG, MAX, MIN.

(1) In the fixed table report, you can add a total directly using a separate SQL.

(2) In the variable table report, you must add the total first using a Fixed Table report function before you use the Variable Table report function. Because the cell address of the total field will change after you use Variable Table report function.

## 7.3.3 Grouping data and Subreports

Grouped data is data that is sorted and broken up into meaningful groups. In a customer list, for example, a group might consist of all those customers living

in the same Region.

To group data in a report, you should use GROUP TABLE REPORT function. For more detail information, refer to “GroupTableReport Method” and “Group Table Report” in this document.

Using the feature of grouping data, you can make subreports within a report. A subreport would typically be used to perform one-to-many lookups such as Customer / Order / OrderDetails.

To make sub reports within the main report,

1. Write a JOIN SQL statement to access data from two or more tables. For example, you can join Customers, Orders and OrderDetails tables.
2. Use GROUP TABLE REPORT function.

For more detail information, refer to the samples invoice.xrf, product_catalog.xrf and sales_detail.xrf within XLReportCom.

### 7.3.4 Subtotaling

A subtotal is a summary that totals or sums numeric values in a group. You can sum the values in each group, count all the values in each group, and determine the maximum, minimum, average in each group. For example, determine the total sales per sales representative in a sales report.

To add subtotals, you can use the functions of Microsoft Excel or aggregate functions in SQL statement.

1. You can add sub-totals using the math functions of Microsoft Excel, such as SUM.

(1) The range of the SUM function should contain the cells for the details in the report template file.

(2) The range of SUM function must contain at least one row/column that is not included in the range for the details. For example, the row 13 is for the details, you should add blank row 14, and write the function as SUM(H13:H14). If you

do not want to show the blank row in the report, you may hide the row.
(3) You should use the relative references. For example, SUM(H13:H14). Microsoft Excel will change the function automatically when XLReportCom adds some rows in the report.
2. If you want to have a total and sub-totals,
(1) You can add the total using SUMIF function. The range of SUMIF function must contain one row/column that is not included in the range of the group. For example, the range of the group is rows 1:15, you should add blank row 16, and write the function as SUMIF(G:G,"Subtotal:",H1:H16). You may hide the blank row.
(2) You can add the total using the aggregate function in SQL statement. You must add the total first using a Fixed Table report function before you use the Variable Table report function. Because the cell address of the total field will change after you use Variable Table report function.
3. You can add sub-totals using the aggregate function in SQL statement too.
(1) Use aggregate function and GROUP BY clause, get summary data for each group, and insert results into a temporary table.
(2) If you have the different kinds of summaries, repeat the step 1, and insert results into another temporary table.
(3) Use group table report function, and join the detail data and the summary data using JOIN. The summary fields must be included in the group list.
For more detail information, please refer to the samples invoice.xrf and sales_detail.xrf within XLReportCom.

## 7.4 Charting

Charts are visually appealing and make it easy for users to see comparisons, patterns, and trends in data. You can use Microsoft Excel to add sophisticated, colorful charts in your reports. For example, you can see at a glance whether

sales are falling or rising over quarterly periods, or how the actual sales compare to the projected sales.

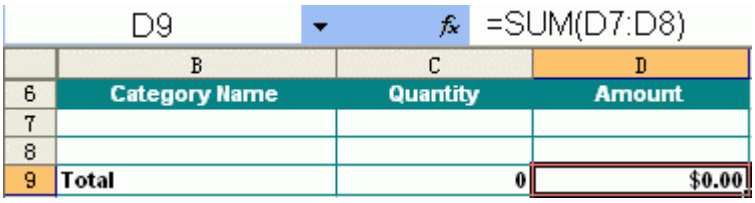
To create a chart in a report, you should create the chart in the template file. You can create a chart on its own sheet or as an embedded object on a worksheet. For more detail information how to create chart, refer to *Microsoft Excel Help*.

To create a chart in the report template file, you may use some sample data. Using sample data, you can set the various chart options. After you have made the report template, delete the sample data. When you generate the report, XLReportCom will put data into the report, and you will get the chart. For more detail information about charting, refer to the sample monthly_sales.xrf within XLReportCom.

**Example**

This example uses REPORT function to create the chart: Sales by Categories.

1. Create a report template with a blank worksheet and a blank chart. The worksheet of the chart defined in the report template as follws:



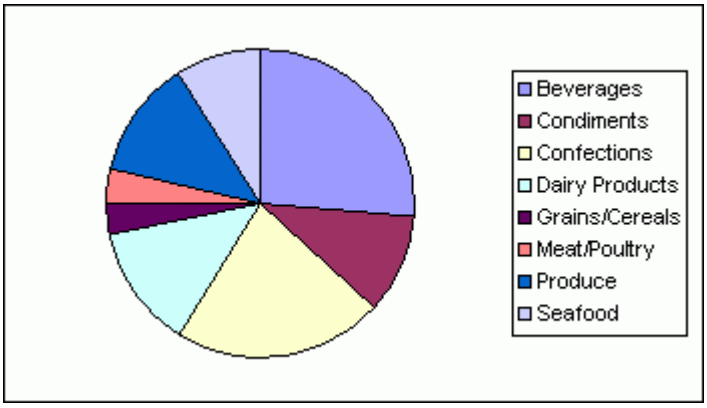

2. Write the report function in an XRF file. The function puts data into the worksheet that provides data for the chart.

```
@F2=REPORT(sheet="Report2" type=var cell=B7 reserve=2)
SELECT c.CategoryName
     , SUM(d.Quantity)
     , Sum(d.UnitPrice * d.Quantity * (1-d.Discount))
FROM Orders o
     ,OrderDetails d
     ,Products p
```

```
      ,Categories c
WHERE o.OrderID = d.OrderID
AND d.ProductID = p.ProductID
AND p.CategoryID = c.CategoryID
AND YEAR(o.OrderDate) = YEAR('1996-04-01')
AND MONTH(o.OrderDate) = MONTH('1996-04-01')
GROUP BY c.CategoryName
ORDER BY c.CategoryName
```

3. Generate the worksheet and the chart.

| Category Name | Quantity | Amount |
|---|---|---|
| Beverages | 925 | $27,761.57 |
| Condiments | 378 | $10,773.27 |
| Confections | 880 | $22,877.18 |
| Dairy Products | 581 | $13,685.32 |
| Grains/Cereals | 189 | $3,325.40 |
| Meat/Poultry | 92 | $4,083.66 |
| Produce | 351 | $13,031.20 |
| Seafood | 669 | $9,316.54 |
| **Total** | **4,065** | **$104,854.15** |




## 7.5 Pictures

### 7.5.1 Inserting pictures into a report template

To make eye-catching reports, you can add pictures to your reports. You can directly insert pictures into the report template in Microsoft Excel. For example, you want to display a logo in your report. You can insert the logo graphics file

into the report template. For more information about adding pictures to worksheets, refer to *Microsoft Excel Help*.

### 7.5.2 Inserting pictures into a report

Except for inserting the static pictures during report design, you may insert pictures during report building process. XLReportCom can put the graphics files into the report, and support all graphics file format that Microsoft Excel support.

To insert pictures into a report using XLReportCom, you should do as follows:

1. Store the paths and names of the graphics files in the database

You store the paths and names of the picture files in database, do not store the pictures. The file path can be a relative path, an absolute path or a URL. For example, you store "images\emp1.jpg" in Photo field.

2. Specify the positioning option and size in the report template

To specify the positioning option and size, you should write a formatting expression into the cell in the report template file. XLReportCom will get the text of the cell, and insert a picture into the cell according to the instruction in the format expression.

3. Write the report function in an XRF file, and identify the image fields using the IMAGE argument. For example,

@F1=Report(sheet="Employee Profile" ... image=photo)

4. Use XLReportCom to generate report with pictures

XLReportCom will submit the SQL statement and get the data from database, read the graphics files according to the paths and names, and insert them into the report. If the path and file name of the picture is “”, XLReportCom will return “”. XLReportCom will return “#Error” if it does not find the file of the picture.

For more detail information about pictures, refer to the samples employee_profile.xrf, product_catalog.xrf within XLReportCom.

# Chapter 8 Hints and Tips

You can run ExcelReport.exe in command line. The format is:

excelreport <xrf file name> [-d] [-u1 user1] [-p1 pwd1] … [pa1 pa2 …]

For example:

excelreport c:\excelreport\monthlysales.xrf 199605

ExcelReport.exe can be scheduled with Windows Scheduled Tasks or other tools. The process of generating reports can be fully automated, periodically or on events.

XLReportCom comes with a sample database, VB sample programs, VBA sample programs and sample reports. You can use them when learning the program. To use the samples, you must add a data source named “Report Sample” to specify the sample database.

To make a report template, you can use some sample data. It is very useful especially for formatting. After you have made the report template, you delete the sample data.

To create a chart in the report template file, you can use some sample data. Using sample data, you can set the various chart options. After you have made the report template, you delete the sample data.

You can use formulas to perform calculations in a report template file.

XLReportCom is a converter too. Besides Microsoft Excel workbook, you can generate a report in other file format such as HTML, XML, Lotus 1-2-3, CSV,

text. You also can convert data from database to other file format.

You can protect the generated report so that it can not be modified.

You can edit an XRF file (.xrf) with a text editor such as Notepad.

In an XRF file, for the report template file, report file and log file, it is possible to give a relative path. If it is a relative path, the base path is the path of the XRF file.

In an XRF file, you can use parameters in the SQL statements. To use parameters, you must define them first.

In an XRF file, you can use parameters in the paths and names of the report file, template file and log file. To use parameters, you must define them first.

You should be careful to define a unique name for each parameter, because XLReportCom will replace all strings that are the same as the names of the parameters. It is a good choice a name begins with the “$” character such as “$ReportDate”.

If you get some errors when you run ExcelReport.exe, you can check the default log file “ExcelReport.log” under the XLReportCom program directory. If you do not define the log file in the XRF file, or can not create the log file defined, you can find log information in the ExcelReport.log.

In the [SQL] section in the XRF file, you can use comments. A comment is the “/*” characters, followed by any sequence of characters (including new lines),

followed by the “*/” characters. You cannot nest comments.

To add totals or subtotals, you can use the functions of Microsoft Excel or aggregate functions in SQL statement.

To group data in a report, you should use GroupTableReport method or Group Table Report function.

In Group Table Report function in the XRF file, the order of groups should be in accordance with the order of ORDER BY clause in the SQL statement.

You can create reports with pictures using XLReportCom. You should store the path and name of the graphics file in the database, identify the image fields in the report function, and specify the positioning option and size in the report template file.

To convert from pixels to points, it is depend on the screen resolution (DPI). If you have a 96 dpi screen (Windows PC), 4 pixels are equal to 3 points.

It is very useful to create a temporary table. You can prepare data using INSERT/UPDATE/DELETE/INSERT SELECT, and then make a report using REPORT function.

You can write a program to make an XRF file using C, perl or DOS shell, and then run ExcelReport.exe to generate report. The two steps can be written into a batch file.

In general, group variable table report is slower than non-group variable table

report. But if the ranges of all groups are same as the range of details, it is faster.

It may take a lot of time to add pagebreaks. If you change the default printer or delete all printers on your computer, it will probably impact the performance.

XLReportCom supports Microsoft Excel 2007. You can use xlsx file as report file and template file. Please copy “xconv2007.cfg” to “xconv.cfg”.

# Chapter 9 License and Support

## 9.1 License

**Your Agreement to This License**

You should carefully read the following terms and conditions before using, installing, copying, or distributing this software. Your use, installation, copying, or distribution of XLReportCom indicates your acceptance of this agreement ("License").

**NO WARRANTY**

XLReportCom IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

**Evaluation License**

XLReportCom is not free software. You may use this software for evaluation purposes without charge for a period of 30 days. If you use this software after the 30 day evaluation period, you must purchase it.

You may copy the evaluation version of this software and documentation as you wish, and give exact copies of the original evaluation version to anyone, and distribute the evaluation version of the software and documentation in its unmodified form via electronic means. You are specifically prohibited from charging, or requesting donations without permission from the author.

**Developer License**

The software is licensed per developer. This means that each developer using the software needs one license. The developer may use the software on one or more computers. You may develop your application that bundles or makes use of the software directly/indirectly. You can not use the software to build competitive products of any kind, like XLReportGen.

You may not resell, rent, lease, sub-license or distribute the software alone. The software must be distributed as a component of an application and bundled with an application or with the application's installation files. You may distribute royalty-free the run-time files of the software with your applications. You need to duly inform your customers that they are not allowed to use the software independently from your application.

## 9.2 Technical Support

If you encounter any problems in usage of XLReportCom, and need the technical support:

- Go to our support web site at: http://www.ljzsoft.com/support.htm
- Send email to support@ljzsoft.com