

SAMPLE CHAPTER

SHAREPOINT 2007  
DEVELOPER'S GUIDE TO

# Business Data Catalog



Brett Lonsdale  
Nick Swan



***SharePoint 2007***  
***Developer's Guide to Business Data Catalog***  
by Brett Lonsdale  
and Nick Swan

**Chapter 10**

Copyright 2010 Manning Publications

## *brief contents*

---

- 1 ■ Introducing the Business Data Catalog 1
- 2 ■ Understanding the application definition file 17
- 3 ■ Security 41
- 4 ■ Out-of-the-box BDC Web Parts 59
- 5 ■ Using the Business Data field type in lists and libraries 83
- 6 ■ Configuring BDC search 109
- 7 ■ MOSS user profiles 129
- 8 ■ The ApplicationRegistry namespace 143
- 9 ■ Creating a custom BDC Web Part 171
- 10 ■ Integrating the Business Data Catalog with Microsoft Office 193
- 11 ■ Writing back to the line-of-business system 215

# 10

## *Integrating the Business Data Catalog with Microsoft Office*

---

### ***This chapter covers***

- Introducing Office Business Applications (OBA)
- Understanding where the Business Data Catalog fits in
- Introducing Visual Studio Tools for Office (VSTO)

Office Business Applications (OBA) is a huge topic area that deserves a book in its own right. Steve Fox has done a good job of this already with his book, *Programming Microsoft Office Business Applications*. The website [OBACentral.com](http://OBACentral.com) is also well worth visiting, and includes videos of BDC solutions that you can watch online. But following an introduction to OBA in this chapter, we want to concentrate on where the Business Data Catalog can provide data from your LOB systems and make that data available in your Microsoft Office client applications. We'll look at the development tools available throughout the Microsoft Office

suite, and explore in detail the different project templates available to us with VSTO (Visual Studio Tools for Office), including the ability to create custom task panes, ribbons, and documents. We'll then utilize the web service that we created in chapter 8, which will enable us to consume our business data in Office clients for each user whose device is remote from the SharePoint Web Front-End Server.

## 10.1 *An introduction to Office Business Applications (OBA)*

---

OBA is difficult to describe, as it's a solution that uses parts of the Microsoft Office suite of services, which includes server-side services such as Business Intelligence and workflow, through to client applications such as Microsoft Word 2007. The application can be a collection of services and devices brought together to create an Office-based solution. It's fair to say that most information workers spend a lot of time in Microsoft Outlook, Excel, Word, PowerPoint, and even SharePoint. They also have a collection of devices such as desktop PCs, laptops, and PDAs or smart phones. Using OBA, we can utilize these devices and software that our information workers are already familiar with to help build a solution that saves time and money for your organization.

### **The history of Office**

Microsoft Office 3.0 was the first version of Microsoft Office for Windows. Back then, there was little or no interoperability between the applications. As Microsoft Office has evolved, we've seen the products work very closely together. SharePoint has now become the hub for all of the Office applications.

I remember teaching my first SharePoint 2003 Power End User class and being blown away by how easy it was to search my portal from within the Research task pane. You could configure the services to allow a SharePoint Portal Server Search, as well as search for translations or thesaurus. Being able to perform a search on portal content from within the Office client was especially useful when you were working on a team-related document, and wanted to quickly search your portal for similar content or a snippet of some kind to reference or incorporate into your own document. Not only could you search for content within documents and lists, but also the network shares and user profiles. Office 2007 has improved upon this by allowing you to create your own task panes and make them look professional using your own custom interface. You could even enhance the interface with Microsoft Expression, providing not only a professional-looking interface, but a striking professional user interface. What's really surprising is that it's so simple to do—even for a non-artist like myself.

### **10.1.1 Overcoming an everyday problem with OBA**

OBA isn't just about custom task panes. OBA is a bunch of Office 2007 services, brought together with the aim of cutting down on the number of manual processes that we carry out. Those services include workflow, Business Intelligence, BDC, Excel Services, and so on.

An example of a manual process that OBA could improve upon would be creating a proposal in Word. This is something that I do on a daily basis without any automation. Even though I'm using SharePoint and Office 2007, I still carry out several manual steps that could be made a lot easier and less time-consuming if I adopted an OBA. When I create a proposal, I need to obtain the customer's name and address. I do this by copying the information from a database once I've retrieved the correct customer. Once I have the customer details, I sometimes need to search for the product to find the correct price to quote. This involves another manual step, with room for error because I'm flipping between applications and using copy and paste. Upon saving the document to a SharePoint document library, I take a local copy of the document and attach it to an email, which is then sent to the customer's email address.

This system is a manual process, despite having sophisticated applications at my fingertips. So how can that process be improved upon? First, in Word, we could create a custom task pane that hooks into the Business Data Catalog. This custom task pane will allow us to perform a search on a customer, be it via company name, ZIP code, or whatever column we like, as long as we've set up the ADF correctly with filters on the required columns. After providing a search, summary information about the customer could be displayed, such as the customer's order history or any outstanding invoices. Using content controls, we can then place the BDC data into the Word document in the appropriate location. For example, the Product column in an invoice can be a drop-down list of product names, and then upon choosing the product name, we populate other content controls with the price and availability of that item.

Custom ribbons also help navigate our OBA by providing buttons to display or hide our custom task panes, email the proposal, or trigger a specific workflow. Everything that forms part of the proposal-sending process could be available on our custom ribbon. The nice thing about this solution is that we wouldn't have to Alt+Tab out of Word to obtain information. It would also mean that information workers can carry out simple tasks such as sending a proposal without the need to train that information worker on the LOB system.

What's the situation when the proposal is ready to be sent to the customer? All we'd have to do is save the document to the document library, where a workflow will be triggered. This could be a simple SharePoint Designer workflow or a more complicated Visual Studio workflow that emails the completed document to the customer. Workflow is another part of the Office-based system, as it's incorporated into SharePoint.

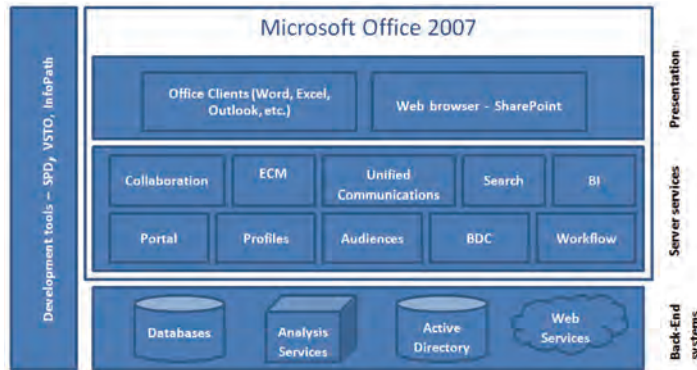
Business Intelligence would also be available to this solution, because the proposal is already stored in a document library with metadata associated with it. The customer name, address, proposal amount, and so forth is all available to sort, filter, and create views upon within the document library. If we have MOSS 2007 Enterprise, we can then display that information in the form of KPIs (key performance indicators) on a dashboard page within the Reports Center. This would allow us to predict the sales forecast for the following month and whether we're likely to hit target.

### **10.1.2 Collaboration of applications**

So really this is what we mean by OBA: we have Office functionality such as Business Intelligence, workflow, communication, collaboration, Business Data Catalog, search, Excel, Word, Outlook, SharePoint Designer, Excel Services, and InfoPath all working together in one solution. The OpenXML file format of Office 2007 has made this possible by allowing us to manipulate and create Office documents more easily through the use of XML and Visual Studio.

There are many different applications and ways in which you can create an OBA. One way is to use InfoPath, SharePoint Designer, and Visual Studio. What we're going to concentrate on here is Visual Studio 2008, and how it can be used to help us build an OBA that integrates our line-of-business data with Word 2007. The diagram in Figure 10.1 displays the many different services at presentation, server, and back-end system level that all contribute to an OBA.

In the next section of this chapter, we'll explore some uses of OBA and BDC that will provide food for thought on how you can build your applications within your own organizations.



**Figure 10.1** The services within the Microsoft Office system that play a role in OBA, including the tools that can be used to develop within OBA

## 10.2 Where does the Business Data Catalog fit in?

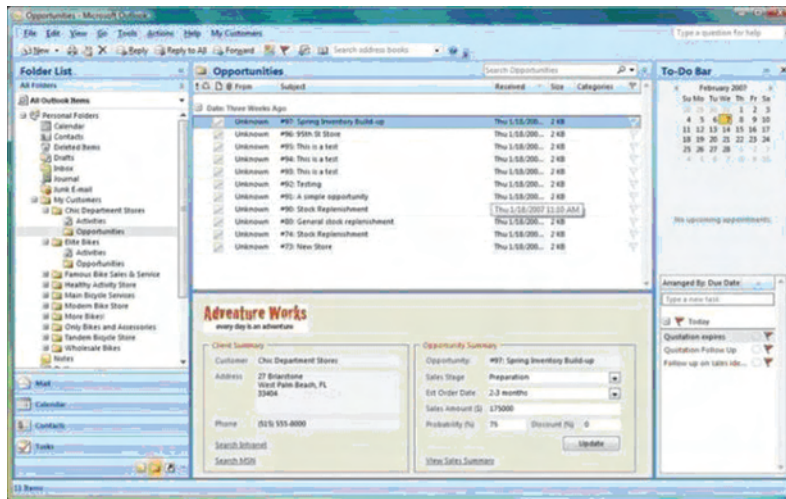
Everything we've been doing with the BDC up to this point is (kind of) already an OBA. Presenting line-of-business data in a dashboard inside SharePoint is an OBA. Using the business data inside the Document Information panel or creating a workflow on BDC metadata is also an OBA. But what we hope to demonstrate here are some real-world uses of line-of-business data within the Office client. Before we start developing, we'll explore some opportunities to use line-of-business data in Outlook, InfoPath, Word, and Excel. InfoPath is also discussed in chapter 11, under the heading of writing back to the line-of-business system.

### 10.2.1 Outlook

Custom form regions in Microsoft Outlook 2007 provide the opportunity to customize the existing forms and also create new forms. An example would be a contact information lookup from your CRM application on the Contacts form. You could even use the customer information from BDC in your email messages. The form regions can replace an entire form or be part of the form. You can also use multiple form regions within one form. Figure 10.2 displays a form region allowing customer and order information to be displayed for a particular customer.

Form regions can be created from within Outlook 2007 or by using Visual Studio 2008. Using Visual Studio 2008, you can create task panes and form regions as add-ins. Form regions are often more beneficial if you have a lot of information, because there's more space available. Task panes are useful for making small panes that blend nicely with out-of-the-box Outlook. Custom ribbons can also be





**Figure 10.2** A custom form region replacing the existing Inspector window. This example is taken from the SDK.

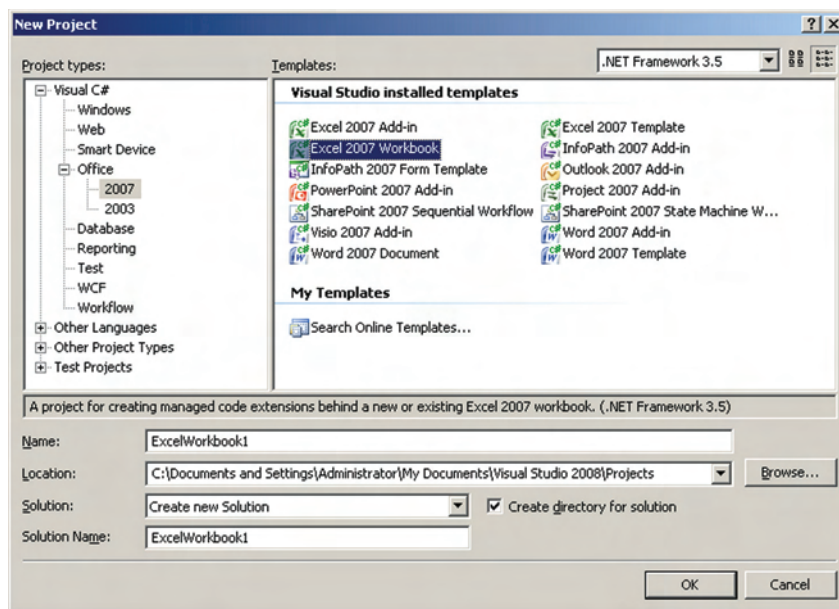
created for Outlook, as well as command bars. Command bars are still used in some areas of Outlook, as with SharePoint Designer and InfoPath.

### 10.2.2 Excel

Microsoft Excel 2007 has the ability to obtain external data from legacy systems without needing the Business Data Catalog. You can connect to data sources directly, such as Microsoft Access, SQL, ODBC, and also data sources that are exposed as a website. Once the data is imported or linked into Excel, it can be manipulated with the usual Excel functionality, such as pivot tables, charts, aggregate functions, and calculations. What's even better is that it can then be displayed in an Excel Web Access Web Part by publishing the workbook to a document library enabled for Excel Services. This allows you to view data in the form of a list, pivot table, or chart via a simple web part on a reporting page that can also be filtered by other Filter Web Parts. The problem with this is that configuring the data source connection has to be carried out by the person who created the workbook. This person doesn't always have a full understanding of the back-end data, especially how to go about connecting to that back-end data. Therefore, through the use of the Business Data Catalog and VSTO, a workbook or workbook template can be created that houses a ListObject content control. This ListObject control is an Excel list, allowing you to bind data to it. The data is usually stored in a data table

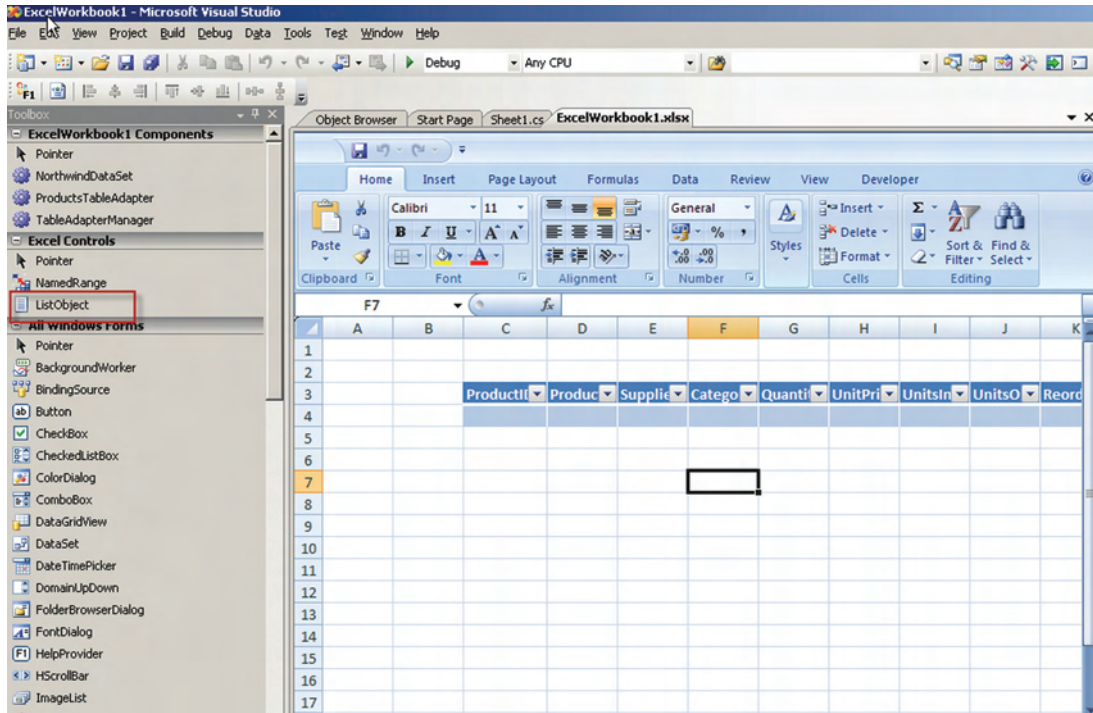
or data set, and can then be bound to the list within Excel. We can then simply provide our information workers with an Excel Workbook with business data available via the Business Data Catalog. It's recommended that you expose the data to Excel via a web service similar to the one we created in chapter 8. The following is an example of what this can look like and how it can be created.

To begin creating such a solution, you need to start Visual Studio 2008 and create a new project. Under (language of choice), then under Office in the New Project dialog box, you'll see some project templates for Word, Excel, Visio, Project, InfoPath, and Outlook. One of the Excel options is an Excel 2007 Workbook, as illustrated in figure 10.3.



**Figure 10.3** The available project types with Visual Studio 2008 for Microsoft Office 2007

Once you've created the workbook within Visual Studio 2008, you'll be able to start adding controls and adding some code. Ideally, you'll reference your web service that returns the data from your entity, and then instantiate the web service and use the results to populate the ListObject control within the workbook. Figure 10.4 displays the Excel controls within Visual Studio 2008, with the ListObject added to the workbook sheet. The ListObject is then bound to the BDC data returned by the web service.



**Figure 10.4** The ListObject control in an Excel Workbook Document project type within Visual Studio

When you press F5 to run your application, Excel is launched and you can see that the data is retrieved at runtime. We ran the code to bind the ListControl in the `ThisWorkbook_startup` event. Figure 10.5 shows the result from running the OBA. The business data is retrieved and displayed in the list. We then configured a chart on the data, which will be used within an Excel Web Access Web Part. Figure 10.5 illustrates the result of using the list object with the Business Data Catalog.

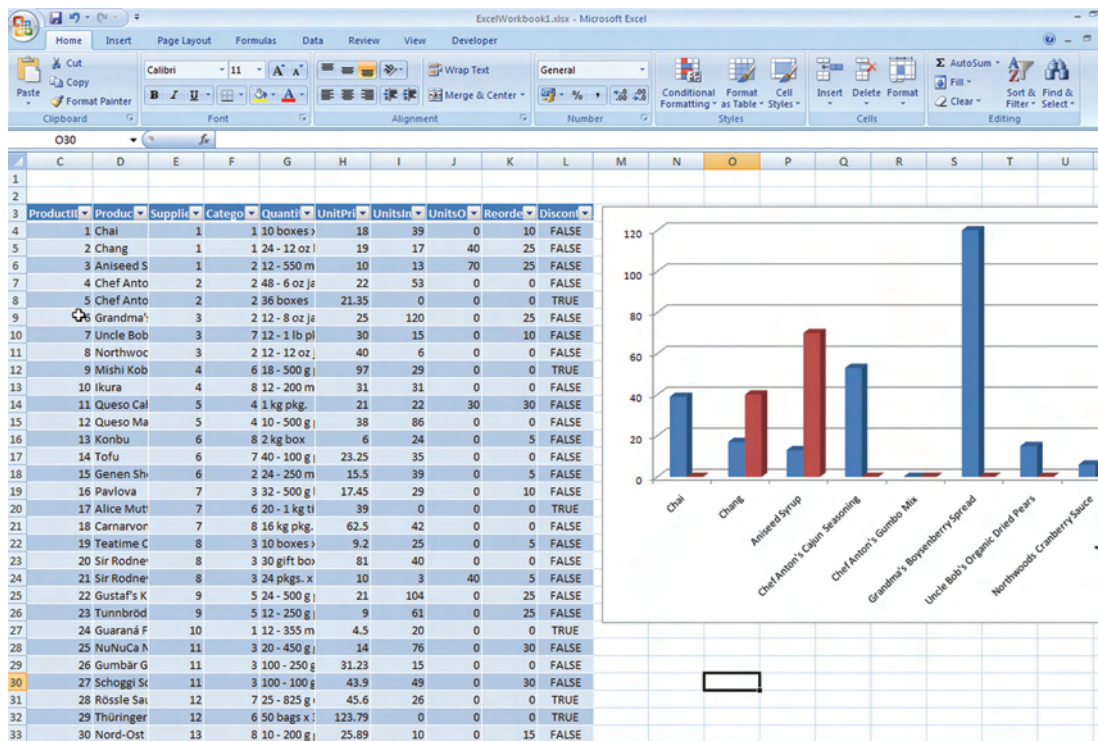


Figure 10.5 The runtime of the Excel document

Once you have the BDC data within the Excel Workbook and you've manipulated it, you can publish the workbook to Excel Services. The advantage of doing so is that any named object within the sheet can be displayed within the Excel Web Access Web Part. To publish the workbook, click the Office button and then choose Publish, Excel Services, as shown in figure 10.6.

When you're publishing the workbook, you can either publish everything or publish specific objects. In figure 10.7, you can see that we're publishing both the list and the chart. These objects can then be displayed selectively via Filter Web Parts in a SharePoint dashboard page. Figure 10.7 demonstrates how to choose which options to include.

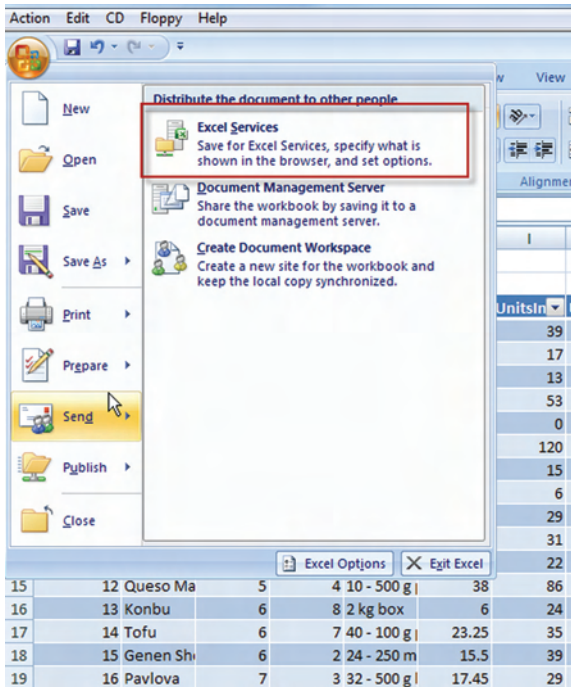


Figure 10.6 The Publish to Excel Services option in Microsoft Excel 2007

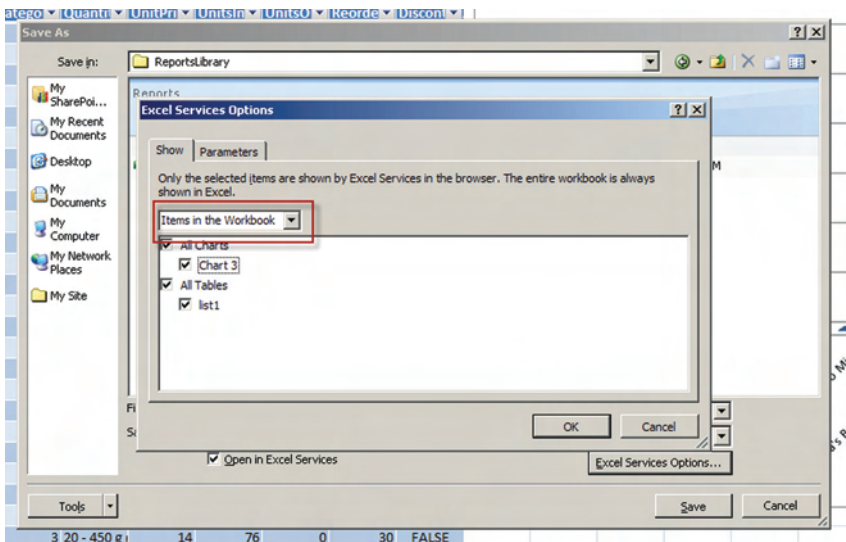


Figure 10.7 Excel Services options, set to publish only certain items within the workbook



Once the workbook has been published, you can set up the Excel Web Access Web Part in a dashboard to display either of the two objects. As you can see, the Excel Web Access Web Part has been configured to display the list1 object. The toolbar in the web part gives you the ability to flip back and forth from displaying the list or the chart. Of course, it's possible to display more than one web part so that both objects are displayed at the same time on the dashboard, or I could add a Filter Web Part that would allow the user to choose. Figure 10.8 illustrates the Excel Web Access Web Part in use on a dashboard page.

Excel Web Access - ExcelWorkbook1

Open Update Find View: List1

Product	Product	Supplier	Category	Quantity	Unit Price	Units in Stock	Units on Order	Reorder Level	Discontinued
1	Chai	1	1	10 boxes	18	39	0	10	FALSE
2	Chang	1	1	24 - 12 oz l	19	17	40	25	FALSE
3	Aniseed Syrup	1	2	12 - 550 ml	10	13	70	25	FALSE
4	Chef Antoinette	2	2	48 - 6 oz jar	22	53	0	0	FALSE
5	Chef Antonette	2	2	36 boxes	21.35	0	0	0	TRUE
6	Grandma's	3	2	12 - 8 oz jar	25	120	0	25	FALSE
7	Uncle Bob	3	7	12 - 1 lb pkgs	30	15	0	10	FALSE
8	Northwoods	3	2	12 - 12 oz jar	40	6	0	0	FALSE
9	Mishi Kobayashi	4	6	18 - 500 g	97	29	0	0	TRUE
10	Ikura	4	8	12 - 200 ml	31	31	0	0	FALSE
11	Queso Caliente	5	4	1 kg pkg.	21	22	30	30	FALSE
12	Queso Mascarpone	5	4	10 - 500 g	38	86	0	0	FALSE
13	Konbu	6	8	2 kg box	6	24	0	5	FALSE
14	Tofu	6	7	40 - 100 g	23.25	35	0	0	FALSE
15	Genen Shoyu	6	2	24 - 250 ml	15.5	39	0	5	FALSE
16	Pavlova	7	3	32 - 500 g	17.45	29	0	10	FALSE
17	Alice Mutt	7	6	20 - 1 kg tin	39	0	0	0	TRUE
18	Coriander	7	6	16 - 1 kg tin	22.5	40	0	0	FALSE

**Figure 10.8** The configured Excel Web Access Web Part displaying the list of BDC data from the workbook

Now that we have the workbook published to a document library in SharePoint, I can configure some key performance indicators on the data within the workbook. Within the Reports Center in MOSS, you'll already have a KPI list created called Sample KPIs. This KPI list gives you the option to create KPIs on Excel Workbooks, SharePoint lists, SQL analysis services, or on manually entered data. We've configured it to look at our published workbook, and have set up a KPI to warn us if the stock level falls below the reorder level. Figure 10.9 demonstrates that you can then create KPIs on the published workbook.

Home > Reports > Sample KPIs > New Item

## Sample KPIs: New Item

**Name and Description**

Enter the name and description of the indicator.

The description explains the purpose or goal of the indicator.

Name

Description

**Comments**

Comments help explain the current value or status of the indicator.

Comments

**Indicator Value**

Select the workbook that contains the information for the indicator value.

Select the cell in the workbook that contains the indicator value.

The cell address can be any valid Excel cell address for the selected workbook such as Sheet1!\$A\$1 or the name of a cell such as "Total".

Workbook URL

Examples:

http://portal/reports/workbook.xlsx

or /reports/workbook.xlsx

Cell Address for Indicator Value

Example: Sheet1!A1 or Total

**Status Icon**

The status icon rules determine which icon to display to represent the status of the indicator.


Values can be either numbers, or valid Excel workbook cell addresses such as: Sheet1!\$A\$1 or Sheet1!A1.


For some indicators, such as "The percentage of tasks completed", better values are usually higher.


For other indicators, such as "The number of active tasks", better values are usually lower.

Status Icon Rules:

Better values are

Display  when has met or exceeded goal

Display  when has met or exceeded warning

Display  otherwise

**Figure 10.9** The KPI configuration page in the Sample KPIs list

Once that list item has been created in the sample KPIs list, the KPI will automatically display in the Sample KPIs Web Part on the dashboard page. Of course, it's possible to configure your own KPI Web Part rather than reusing the Sample KPI Web Part, as shown in figure 10.10.

We hope this has given you some idea of how BDC, Excel, Excel Services, and Business Intelligence can work together to create an OBA with your line-of-business data using Excel. Next, we'll look at Word and will create a custom task pane to display business data, a custom ribbon, and a Word document using content controls.

Sample KPIs

Show Only Problems

Indicator	Status
Morale	
Productivity	
Expenses	
Product Stock	

**Figure 10.10** The Sample KPI Web Part showing the Product Stock KPI that was configured in the KPI list

10.2.3 Word

There are many ways to use the Business Data Catalog with Word. Through the use of content controls, we can select line-of-business data within documents such as invoices or proposals. What we’re going to achieve initially in this demonstration is similar to a mail merge. The document is more interactive, allowing you to actually select data from inside the document itself. So every time you create a proposal or an invoice, you can choose the customer that will receive the document, and your code fills in the other details such as the customer’s address. Note that the Customer section in figure 10.11 is just text copied and pasted from a CRM application. This takes a lot of time to flip between applications, find the right customer, and then copy and paste the address.

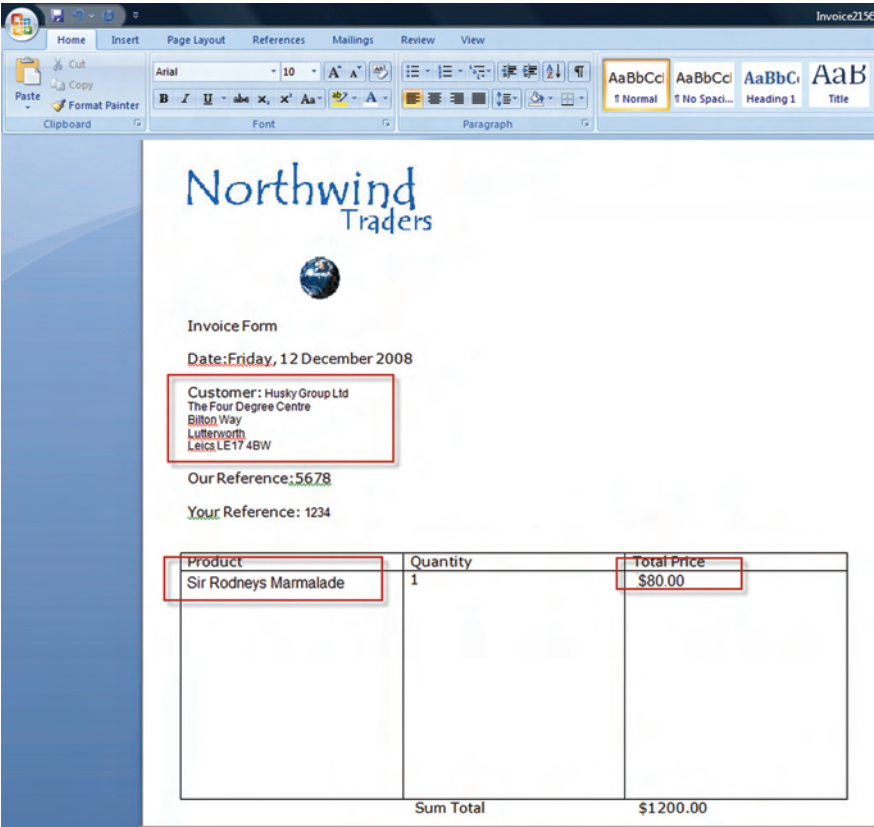
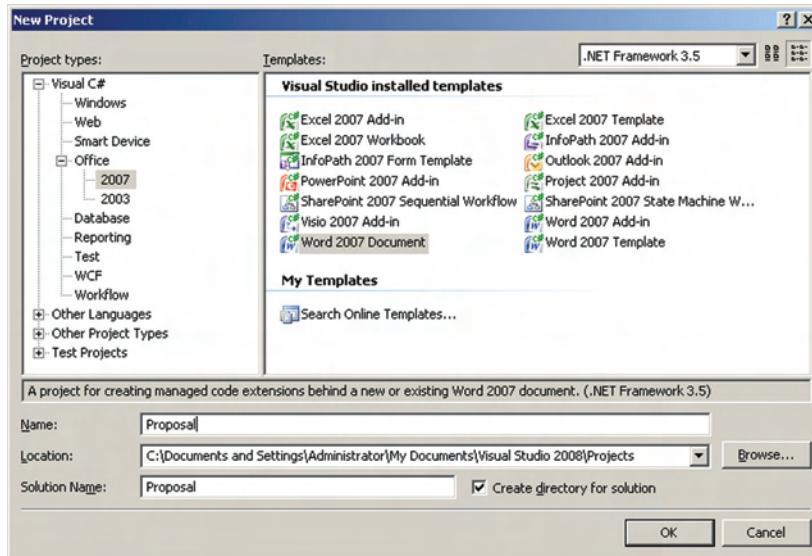


Figure 10.11 The problem in Word with completing the address details



So rather than using copy and paste, we decided to use the Business Data Catalog to allow us to select our customer from the CRM application. To do this:

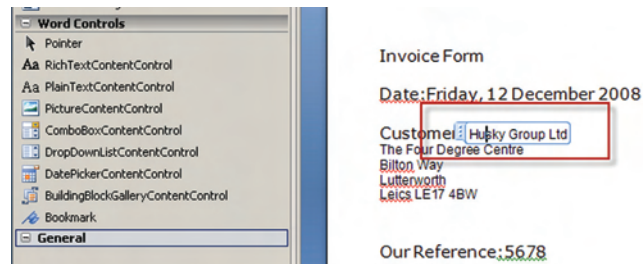
- 1 Start Visual Studio 2008.
- 2 Choose Visual C#, Office, and then Word Document from the New Project dialog box, as shown in figure 10.12.



**Figure 10.12** The New Project dialog box with Word Document selected

- 3 Name the project and click OK.
- 4 On the next dialog box, you can either choose to create a brand new document or open an existing document. In our case, we opened an existing invoice document.
- 5 The document then displays in Visual Studio 2008. Click the Toolbox icon to display the tools if they're not already showing. Then add a `DropDownListContentControl` next to the Customer: text in the Word document, as shown in figure 10.13.
- 6 Immediately below the `DropDownListContentControl`, add a `RichTextContentControl` for the address details. You could do the same thing for the product if you wanted to.
- 7 Add a web reference to your web service that returns the entity data to your Visual Studio project. This can be done by right-clicking References, and

**Figure 10.13** The toolbox of controls available in Visual Studio 2008



then choosing Service Reference. On the Advanced tab of the Service Reference dialog box, you can choose to add a .NET Framework 2.0 web reference if you prefer.

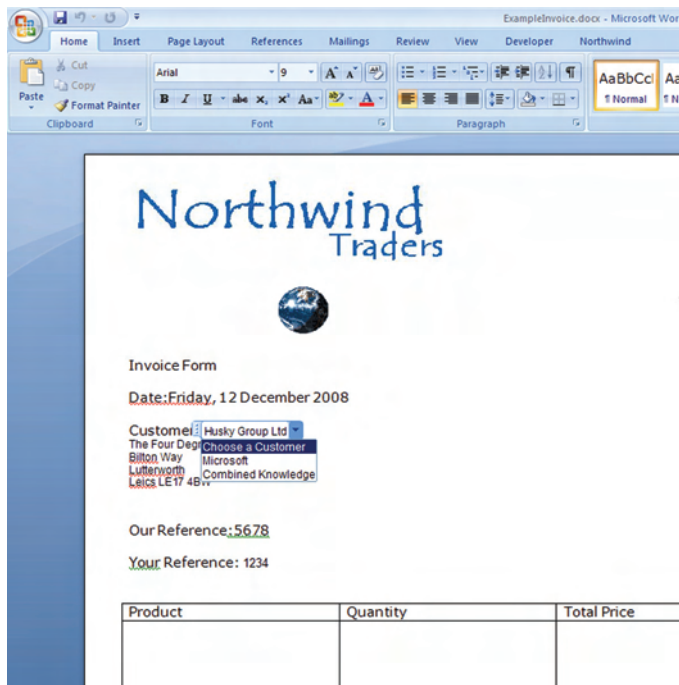
- 8 Double-click your `DropDownListContentControl` to access the code window. Add the code to populate the `DropDownListContentControl` with data returned from a web method to the `ThisAddIn_Startup` event, so that your control is populated with choices upon creating the document.
- 9 Add code to the `DropDownListContentControl_Exiting` event to populate the `RichTextBoxContentControl` with the remainder of the address upon selection of a customer. The code in listing 10.1 makes a call to the WCF web service to populate the `DropDownList` control.

#### Listing 10.1 `DropDownListContentControl` web service call

```
private void dropDownListContentControl1_Enterung(object sender,
➤ Microsoft.Office.Tools.Word.ContentControlEnteringEventArgs e)
{
    CustomerService.CustomersServiceClient proxy = new
➤ CustomerService.CustomersServiceClient
➤ ("BasicHttpBinding_ICustomersService");
    string[] customernames = proxy.ListCustomers();

    int x = 1;
    foreach (string CustomerName in customernames)
    {
        x += 1;
        dropDownListContentControl1.DropDownListEntries.Add
➤ (CustomerName.ToString(), CustomerName.ToString(), x);
    }
}
```

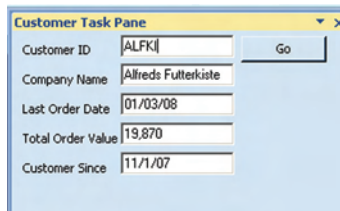
- 10 Test the document by pressing F5 on your keyboard. The result should look like figure 10.14.



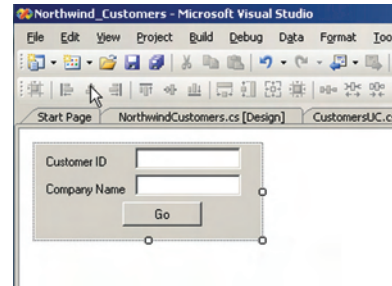
**Figure 10.14** The result of customizing the content controls to allow a customer to be chosen from the `DropDownListContentControl`

In addition to creating content controls, you can create custom task panes. Figure 10.15 shows an example of the kind of task pane that would be useful to sales representatives when sending out quotes. Users can look up customers easily to find out information about them.

- 1 To create a custom task pane, launch Visual Studio 2008.
- 2 From the New Project dialog box, select Word 2007 Add-In from the C# Office section.
- 3 Add the web reference to your BDC web service as we did in the previous exercise.
- 4 Right-click your project and add a UserControl.
- 5 Design your task pane interface on the UserControl, as shown in figure 10.16.



**Figure 10.15** A custom task pane showing useful data about a customer from the LOB system



**Figure 10.16** Designing the custom UserControl

- 6 To populate the controls, you need to call the method that exposes your specific finder from your web service. Pass through the Customer ID, and then assign the company name to the `CompanyName.txt` field. Soon, we'll use a similar piece of code to create the ribbon.
- 7 Open the `.cs` file for your add-in class.
- 8 In the Startup event for the add-in class, add the code from listing 10.2 to instantiate the `UserControl`. Position it correctly within Word by setting the dimension properties of the task pane, as shown in the listing.

#### Listing 10.2 The add-in startup event

```
private void ThisAddIn_Startup(object sender, System.EventArgs e)
{
    CUC = new CustomersUC();
    myCustomTaskPane = this.CustomTaskPanes.Add(CUC,
        "New Task Pane");

    myCustomTaskPane.DockPosition =
        Office.MsoCTPDockPosition.msoCTPDockPositionFloating;
    myCustomTaskPane.Height = 500;
    myCustomTaskPane.Width = 500;

    myCustomTaskPane.DockPosition =
        Office.MsoCTPDockPosition.msoCTPDockPositionRight;
    myCustomTaskPane.Width = 300;

    myCustomTaskPane.Visible = true;
}
```

- 9 Press F5 to test your solution.

You should now be able to search for customers using the custom task pane, which is loaded during the startup of the document. Next, we'll create a custom ribbon that uses an alternate method to look up customer details.

- 1 Right-click your project and choose Add, New Item.
- 2 Select Ribbon (Visual Designer).
- 3 From the Office Ribbon Toolbox controls, drag a toggle button into the first group on your ribbon.
- 4 In the Properties pane, set an image for your button, along with a tooltip and a label. The label should say "Display Customers."
- 5 Double-click the button to open the Code view.
- 6 Add the code to the button, as shown in listing 10.3.

#### Listing 10.3 Customer information lookup ribbon button

```

    CustomerService.CustomersServiceClient proxy = new
➤ CustomerService.CustomersServiceClient
➤ ("BasicHttpBinding_ICustomersService");
        string[] customernames = proxy.ListCustomers();

        gallery1.Items.Clear();

        RibbonDropDownItem item = new RibbonDropDownItem();
        foreach (string CustomerName in customernames)
        {
            string customeraddress =
➤ proxy.GetCustomer(CustomerName).ToString();
            RibbonDropDownItem dditem = new RibbonDropDownItem();

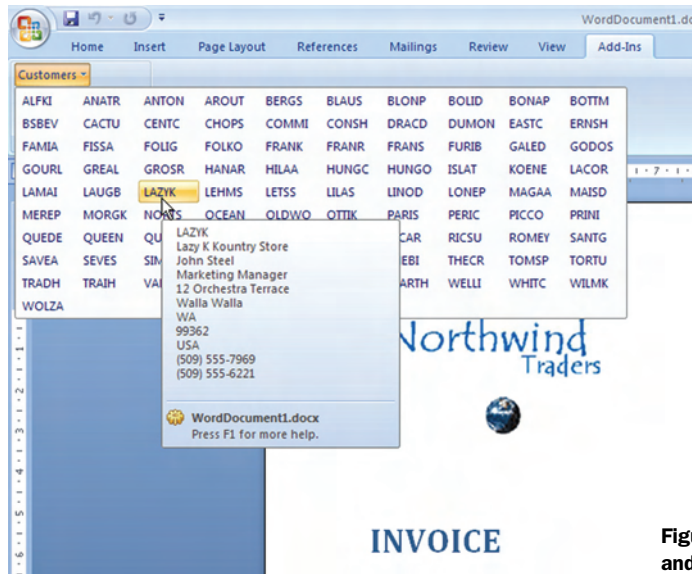
            dditem.Label = CustomerName.ToString();
            dditem.SuperTip = customeraddress.ToString();

            gallery1.Items.Add(dditem);
        }

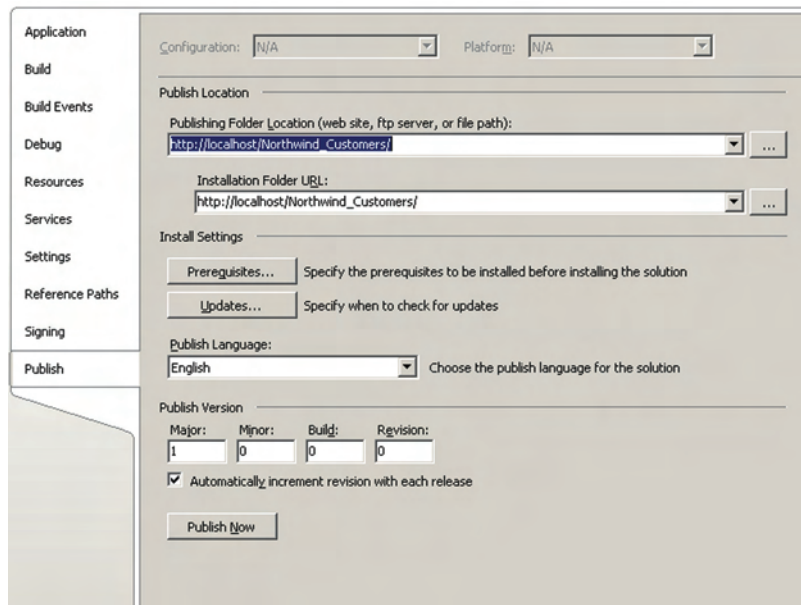
```

- 7 Press F5 to test the solution.
- 8 The finished ribbon should look like figure 10.17 in Word.

Once you've finished your custom document with the task pane, ribbon, and content controls, you'll want to publish the document so that it can be used within a SharePoint content type. The first thing you'll need to do is set up the document's Publish location. This can be done in the Project properties and is usually set to a file share, as shown in figure 10.18.

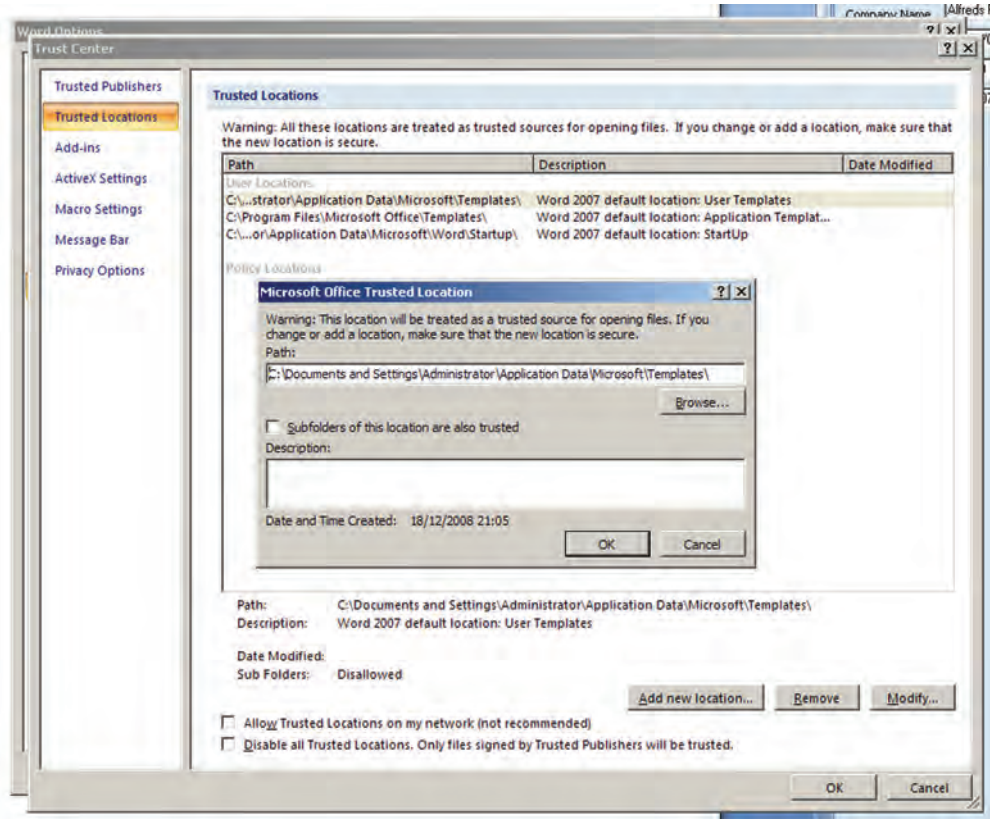


**Figure 10.17** The custom ribbon and ribbon button in Word 2007



**Figure 10.18** The Publish settings in the Project properties

You'll also need to configure the Trust settings. The Trust is configured in Word Options. To set this, you need to trust the location where the document is stored. Click the Office button, then Word Options. Under the Trust Center, click the Trust Settings button. You can then add the Publish location as a trusted location, as shown in figure 10.19.



**Figure 10.19** The Trust Center settings within Word

Finally, you can upload this document as a template within your content types. To create a content type:

- 1 Choose Site Actions, Site Settings from your team site.
- 2 Choose Site Content Types from the Galleries section.
- 3 Click Create to create a new content type.

- 4 Give the content type a name such as Proposal.
- 5 Inherit from Document, which is in the Documents group.
- 6 Click Advanced Settings.
- 7 Choose Upload a New Document Template and browse to the share where your document is published.
- 8 Add the content type to the document library where users will create invoices.

You've now successfully created an OBA using Visual Studio.NET 2008, Word 2007, and SharePoint 2007. Combining these products with the Business Data Catalog provides some powerful solutions.

### 10.3 *Summary*

---

Within this chapter, we took a brief look at Office Business Applications. Having written this chapter, we're inspired to write a whole book on Office Business Applications and the Business Data Catalog. There are so many solutions that could be created! We utilized the web service that we developed in chapter 9 of this book within OBA. We explored OBA solutions within both Excel and Word. We looked in detail at how to create custom task panes, custom ribbons, and content controls. We discussed mixing OBA and BDC, and also how information could then be used with other services such as Excel Services. Given more time and more pages, we could've also explored InfoPath and how that can be used in an OBA.

In the next chapter, we'll explore how to use InfoPath and Visual Studio.NET code to write back to the LOB system. BDC Meta Man can be used to generate Insert and Update Web Parts. In chapter 11, we'll explore exactly how this is achieved.



## SHAREPOINT 2007 DEVELOPER'S GUIDE TO **Business Data Catalog**

Brett Lonsdale • Nick Swan

**T**he data locked in your organization's systems and databases is a precious—and sometimes untapped—resource. The SharePoint Business Data Catalog makes it easy to gather, analyze, and report on data from multiple sources, through SharePoint. Using standard web parts, an efficient management console, and a simple programming model, you can build sites, dashboards, and applications that maximize this business asset.

**SharePoint 2007 Developer's Guide to Business Data Catalog** is a practical, example-rich guide to the features of the BDC and the techniques you need to build solutions for end users. The book starts with the basics—what the BDC is, what you can do with it, and how to pull together a BDC solution. With the fundamentals in hand, it explores the techniques and ideas you need to put BDC into use effectively in your organization.

Knowledge of SharePoint Server and WSS is required.

### **What's Inside**

- The BDC Object Model
- How to build BDC applications
- BDC-driven search
- Integrating with Office, CRM, and InfoPath

**Brett Lonsdale** and **Nick Swan** are the founders of Lightning Tools, a UK-based SharePoint consulting company specializing in the SharePoint Business Data Catalog.

For online access to the authors, code samples, and a free ebook for owners of this book, go to: [manning.com/lonsdale](http://manning.com/lonsdale)



**"This book is an absolute must-have!"**

—Christina Wheeler, SharePoint Consultant, Summit 7 Systems

**"... from experts who know the BDC inside and out."**

—Monty Grusendorf, Senior Web Developer, Bantrel

**"An excellent guide for working with the BDC."**

—Darren Neimke, Author of *ASP.NET 2.0 Web Parts in Action*

**"A one-stop guide for SharePoint BDC developers."**

—Prajwal Khanal  
Senior Software Engineer  
D2HawkeyeServices Pvt. Ltd.

ISBN 13: 978-1-933988-81-8  
ISBN 10: 1-933988-81-9  
5 5 4 9 9



9 781933 988818