CHAPTER 19

Approve, Release, and Prepare

I do the very best I know how—the very best I can; and I mean to keep on doing so until the end.

—Abraham Lincoln

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

—Winston Churchill

What happens when a software project ends? How many times have you heard it isn't over until the paperwork is done? How true that is. There is a lot to consider when completing a project, and how you wrap things up is one of the greatest factors in determining how much of a professional you and your team are.

In this chapter, we teach you how to handle the paperwork as painlessly as possible. We also discuss how to get your BI solution signed off on once the contract is fulfilled, how to announce the release, and then how to release your BI solution, so that your client can finally get the full benefit of all your work. We discuss how to convert your notes into official dev specs and SDK documents as well as how to train your new users how to use their fancy new software by drafting a user manual and help files.

Many of the documents discussed in this chapter are a follow-up to the documents introduced in Chapter 3.

The End of the Cycle

As you can see in Figure 19-1, we have completed all but the last step of our BI solution process, Approve, Release, and Prepare for Next Version. The cycle is in the form of a circle, indicating that the end of one cycle is the beginning of the next. During the planning phase and at other times throughout this book, we discussed data that we chose not to incorporate into the first cycle of the BI solution. Those items can be used as a wish list for possible items to incorporate into a future version. Additionally, as time goes on and your client's company changes, new needs come up that need to be handled. We are never truly finished. And this is a good thing, because it keeps your team employed!

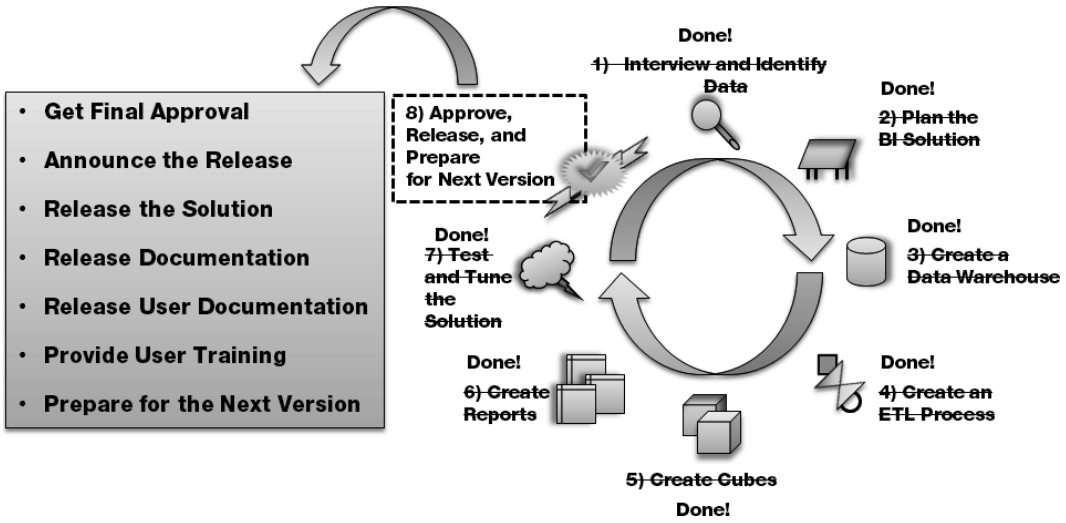


Figure 19-1. *Down to the final step!*

But, before we can begin again, we have some loose ends to tidy up. And we cannot kick off this final stage without the approval from our client, manager, or contract holder. Let's start by getting the needed approval to release their fancy new BI solution.

■ **Note** Several subjects touched upon in this chapter are business practices that will be unique to your company and are implemented according to how you choose to do business.

The Final Approval Process

Why do you need it approved? The answer is simple. It is because you want to get paid! The final approval process begins at the end of each cycle. Many developers are afraid to get sign-off because they are afraid someone will always say, "We missed something." This is all part of preparing for the next cycle, and we will save that topic for later in this chapter.

Who gives the approval is likely whomever you have been answering to all along. It is usually a project manager on your team and a stakeholder who is representing the client receiving the solution. This could also be your supervisor if the BI solution is being developed for the company you work for as an employee. In this case, the approval process is handled according to your company's guidelines.

The Sign-Off Document

In Chapter 3, you created a solution plan, and at that time you agreed to create the project. This agreement might have come in the form of a contract or an accepted solution outline. Once the requirements have been met within the agreement, the approval process can begin.

If the original agreement has been modified, you may need to create a final invoice. This agreement should be handled according to how your company chooses to do business, but the goal is for the invoice to list everything from the original agreement to any new items that were agreed upon. Add lines at the bottom

for proper signatures and dates, include any outstanding balance, and you are good to go! Anything above and beyond the items within the sign-off document can be compiled as a checklist for version 2.

Announcing the Release

One of the biggest problems that can occur within a company when a new BI solution is released is resistance to change. This resistance comes from suspicion that the changes will be more of a hindrance than a help. Informing the company's employees of the changes that are coming with the release of the BI solution can be very helpful in this situation. For larger companies, this can be accomplished in an official manner with a press release.

The benefits of writing a press release are twofold. You give your team credibility, and you get the users on board with the new changes. A good press release can break through the resistance and have users anticipating what is up-and-coming rather than fighting it.

A good press release will include the following:

- A strong title that piques interest
- The announcement of what is about to happen
- How it benefits everyone
- Two quotes from key people of importance who support the changes (this is optional, but it helps support the changes and can generate excitement)
- A boilerplate (fine text) at the bottom that includes your company contact information

The Press Release Title

Many of the news articles you read online when browsing the Internet are written in press release format. A good press release will always have a good title. If the title does not catch your interest, you won't click it to see what it is all about. Additionally, the title should be short. This is particularly true when it is displayed on a website that has limited space for title links.

Your title should be something that catches your client's employee's attention. This will be different for every company. If you find you are at a loss for ideas, you are not alone. Coming up with a good title is often the hardest part of writing a press release. Some companies that publish many press releases on a regular basis hire someone other than the original author to write snappy titles. This is evidenced when a news article is published originally with one title but the title is changed within 24 hours of publication and you find yourself clicking it again not realizing you have read it. Tricky? Maybe. But the purpose is to improve the article by changing the title rather than to trick you.

That's the good news. If you don't like your title, it can always be changed later, or you can let someone else title it for you.

For our example BI solution press release, we might title it something very simple yet informative, like "Publication Industries Revamps Ordering System," or something flashier, like "Publication Industries Computer System Is About to Get Better!" If we were very limited on space, we might simply title it "Upcoming Software Changes," although that title doesn't exactly get people excited.

If it is announced on the company website, sometimes you have room for a small blurb about what those changes may be next to the link. But keep the details in the press release body, rather than in the breadcrumbs that lead them to it.

The Press Release Body

The announcement body should be nothing more than facts that matter without a lot of hype or hot air about how good you are. Press releases are short. They should fit on one page or be fewer than 400 words if it is web based.

Make a list of what you want to say and order the facts in a list of what is most important. Keep in mind that you will want to include how it benefits the reader. You need to tell them who, what, where, how, and why. You do not need to tell them that you are superfabulous for having worked so hard on it for so long and that you are relieved for it to be over. Your readers seldom care. They would rather know how it affects them and why changes are happening.

Once you have your list, consider writing a couple of one- or two-sentence quotes from the top of the food chain in your client's company to add creditability and a show of support to the changes from higher-ups. This is an old trick, but it works!

Compose the press release starting with the most important fact at the top, and go down your list to the least important fact at the bottom. Keep it interesting. If you aren't sure if the fact you want to mention is very interesting, put it at the bottom. The first few sentences should tell nearly the whole story.

The Press Release Boilerplate

The boilerplate is nothing more than your contact information. It lets your readers know who composed the press release, and it also gives you some advertisement to boot.

WRITING A PRESS RELEASE

Practice your writing skills by trying your hand at writing a press release. Be sure to include the following:

- A concise and informative title
- A list of important facts
- A press release body including the facts in order of importance first
- No more than two quotes (you can fabricate these for this exercise)
- A boilerplate

Note: For more information on press releases, search the Web for the term "press release" and look under Images. You will find interesting examples of press releases that give clear examples, templates, and visual styles of many types of press releases that may help with the writing process.

A press release of this type could be published on your client's website, sent via email to your client's employees, or printed out and posted at each place of business where the users operate.

Releasing the Solution

Once you have obtained the sign-off, it is finally time to get that BI solution released to the production server. Releasing the solution involves deploying it to the production server and making it available to the users.

The standard process of deploying the BI solution is to copy the solution objects from a development server to a testing server and finally to a production server. Of course, there are endless variations of this, but this arrangement is common practice.

Now the question becomes how to copy the objects from one place to another with the least amount of effort and errors. First, identify what you are copying and how it can be copied. Table 19-1 provides a list applicable to our current BI solution.

Each of these choices varies slightly, but all involve collecting solution artifacts and using the artifacts for deployment.

Table 19-1. *Copying Objects from a Development Server to a Testing Server*

Object	Locations	Method Used to Copy Object
Data warehouse database	SQL Server instance	Back up and restore database
		Script database and database objects
SSIS ETL package	SQL Server instance	Copy and paste `.dtsx` file
	File system	Deploy using Visual Studio
		Import using SQL Server Management Studio
SSAS database	SQL Server instance	Deploy using Visual Studio
		Back up and restore SSAS database
		Script SSAS database and objects
SSRS reports	SSRS web service	Deploy using Visual Studio
		Import using Report Manager
		Upload using the SSRS web service

■ **Tip** Solution artifacts represent all items produced during the creation of your BI solution. These translate to files you created in your BI projects.

Collecting the Solution Artifacts

You will need to collect all solution artifacts into one location so that they can be used and managed effectively. Ideally this location will be some sort of source control and release management software such as Microsoft's Team Foundation Server, but even a folder on a network share can work.

Collecting all the files you need for deployment into one shared folder is a simple technique that works well enough for small solutions like the example project from the exercises in this book. Each team member will be responsible for placing their files into one central location.

Some larger solutions will have more than one team working on them. As the number of developers involved in creating a solution goes up, so does the need to manage the folder. Developers have been known to be, shall we say, creative as to what and when they upload to deployment folders. Set a schedule to determine when the files must be in place and indicate which files they need to go into. Also, at some point you will need to assign someone as an official release manager. The manager's job is to coordinate with each team to make sure that the release is orderly and on time (aka keep the developers on track).

For the solution created in this book, our network share will look like the one pictured in Figure 19-2. Here we have placed a backup file for the data warehouse, a `.dtsx` file for the ETL process, a backup file for the SSAS database, and a couple of SSRS `.rdl` report files. (Our examples and images are from Randal's RSLAPTOP2 computer, so remember to use your computer name instead.)

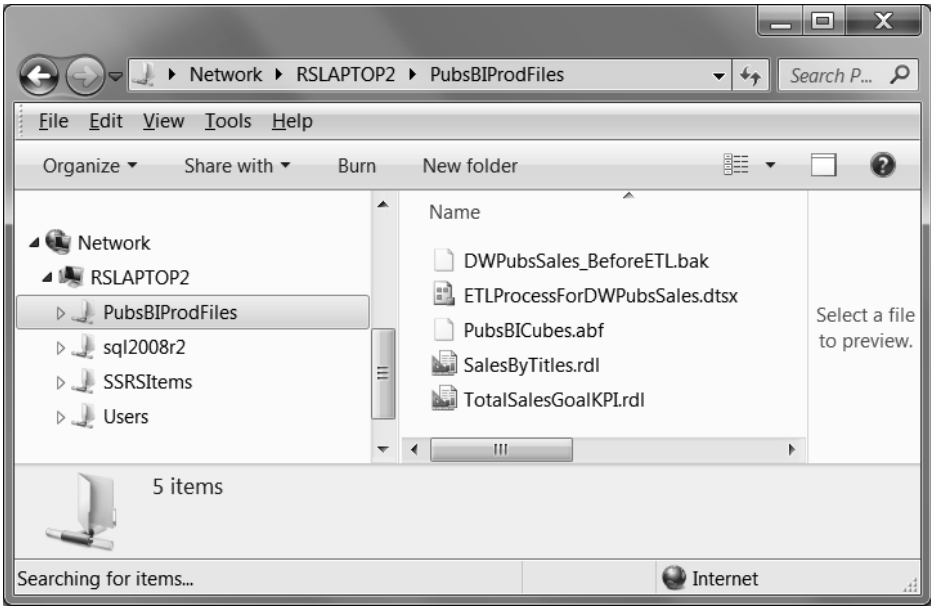


Figure 19-2. *A network share containing solution artifacts*

Deploying the Files

When the files are gathered into one place, the next step is to deploy them to the test or production computers. SQL Server, SSIS, SSAS, and SSRS all have different methods of deployment, making the deployment process very complex. Let's take a look at some foundational things you should know.

■ **Note** This topic quickly gets into advanced administration and programming tasks that may be beyond your current skill set. Just remember that until you become more advanced, you can always deploy your solution manually or hire someone who specializes in release management.

Manual and Automated Deployment

Deploying your files manually involves copying the files to a location that the computer's operating system can use (such as a local hard drive or a network share) and running the SQL code and projects that you have included in your Visual Studio solution. This means that a person is sitting at a computer, running the SQL code in SQL Management Studio, starting the SSIS package, processing the cube, and uploading the reports.

Manual deployment works just fine if you only occasionally deploy the BI solution to new computers. In many test environments, however, the solution may be deployed over and over again on the test servers. In those scenarios, you may want to create some automation code to help streamline the deployment process.

Let's take a moment to give you some examples of how this kind of automation can be accomplished.

Deploying the Data Warehouse with SQL Code

The data warehouse must be in place before SSIS, SSAS, or SSRS can be deployed. Start by restoring a backup file using code similar to that shown in Listing 19-1.

Listing 19-1. Restoring a SQL Server Backup File from a Network Share

```
-- Check to see if they already have a database with this name...
IF  EXISTS (SELECT name FROM sys.databases WHERE name = N'DWPubsSales')
  BEGIN
  -- If they do, they need to close connections to the DWPubsSales database, with this code!
    ALTER DATABASE [DWPubsSales] SET SINGLE_USER WITH ROLLBACK IMMEDIATE
  END

-- Now now restore the Empty database…
USE Master
RESTORE DATABASE [DWPubsSales]

FROM DISK = N'\\RSLAPTOP2\PubsBIProdFiles\DWPubsSales_BeforeETL.bak'

WITH REPLACE
GO
```

Another way to deploy the database is to run a SQL script that creates the database and all its objects, like we did in Chapter 2 (Exercise 2-2). But, restoring the backup includes data, whereas running the script does not. This is both good and bad. It is good because you do not have to run the SSIS process to the data to test your reports and SSAS processing, but it is bad because the backup file may be very large.

You will have to decide what is practical for your situation. Both options work well for deploying most data warehouses, so you really cannot go too wrong choosing one over the other.

You can run SQL code from a command prompt using Microsoft's `sqlcmd.exe` utility. This utility is useful for creating deployment batch files and scripts. For example, the code in Listing 19-2 sets the focus to the directory when the `sqlcmd.exe` utility is located and then executes a SQL file located on a network share called \\RSLAPTOP2\PubsBIProdFiles.

Listing 19-2. Executing a SQL File from a Network Share

```
CD "C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn"
SQLCMD.exe -S RSLaptop2\SQL2012 -E -i \\RSLAPTOP2\PubsBIProdFiles\RestoreDWPubsSales.sql
```

RUNNING SQLCMD.EXE WITH A BATCH FILE

This code can be directly typed into a Windows command prompt or placed into a text file with a `.bat` extension (batch file). The batch file option is preferred, because one batch file can be used for multiple deployments.

Working with the command prompt and batch files was much more common in the early days of PC computing than it is now. We mention batch files several times in this chapter because they are so useful for automating deployments. If you are not familiar with batch files, we recommend searching the Internet for more information; they are really useful!

Deploying the SSIS ETL Process

After the database has been deployed to the production server, the ETL process must be executed by running the SSIS package. This can be done manually using Visual Studio as you have done in this book, or you can execute SSIS code using a command prompt utility called `dtexec.exe`. Listing 19-3 shows an example of how this utility is used.

Listing 19-3. Executing an SSIS File from a Network Share

```
CD C:\Program Files (x86)\Microsoft SQL Server\110\DTS\Binn
DtExec.exe /FILE \\RSLAPTOP2\PubsBIProdFiles\ETLProcessForDWPubsSales.dtsx
```

Combining the SQL Server and SSIS Deployment Code

Release managers will often create a batch file that combines the data warehouse restoration and the SSIS ETL processing by running just one batch file. As an example, the command shell code in Listing 19-4 would be placed in a batch file. Note how it restores the database using `sqlcmd.exe` and then performs the ETL processing using `dtexec.exe`.

Listing 19-4. Combining Multiple Commands into a Batch File

```
REM DeployPubsBISolution

REM Restore the database backup
CD "C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn"
SQLCMD.exe -S RSLaptop2\SQL2012 -E -i "\\RSLAPTOP2\PubsBIProdFiles\RestoreDWPubsSales.sql"
pause

REM Run the ETL process
CD C:\Program Files (x86)\Microsoft SQL Server\110\DTS\Binn
DtExec.exe /FILE \\RSLAPTOP2\PubsBIProdFiles\ETLProcessForDWPubsSales.dtsx
pause
```

To create the batch file, open a simple text editor such as Notepad, type in the code, and save it with a `.bat` extension (for example, `Deployment.bat`).

When the batch file is created, you can execute it by name from a command prompt or by double-clicking the file in Windows Explorer. Either method runs the code contained inside the file as if it had been manually typed at the Windows command prompt.

■ **Tip** If you double-click the file in Windows Explorer, it temporarily opens a command prompt window, runs the code, and immediately closes the command prompt window. By placing the `pause` command on the last line of the batch file, the command prompt window will stay open until you click any keyboard key.

Another advantage to using a batch file in this manner is that you can use the Window's Task Scheduler program to automatically run the file at a designated point in time without the need for human interaction. You can find this program under Administrative Tools in the Windows Control Panel.

Deploying the SSAS Database

An SSAS backup file can be restored using XMLA code. Unfortunately, Microsoft has not provided a command prompt utility to execute XMLA code (although there are some third-party utilities that will). Therefore, it may seem that your only option is to manually restore the database using SQL Server Management Studio or to deploy the SSAS project from Visual Studio as we have done in this book.

You can, however, run XMLA code using an SSIS package that can in turn be automated using batch files. First, create and test some XMLA code that restores the backup file on the network share (Listing 19-5).

Listing 19-5. XMLA Code to Restore an SSAS Database

```
<Restore xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <File>\\RSLAPTOP2\PubsBIProdFiles\PubsBICubes.abf</File>
  <DatabaseName>PubsBICubes</DatabaseName>
  <AllowOverwrite>true</AllowOverwrite>
  <Security>IgnoreSecurity</Security>
</Restore>
```

Next, create a new SSIS package and add an Analysis Services Execute DDL task to it. Finally, configure the task to make a connection to the new SSAS server, and add your XMLA code to the DDL Statements dialog window (as shown in Figure 19-3).

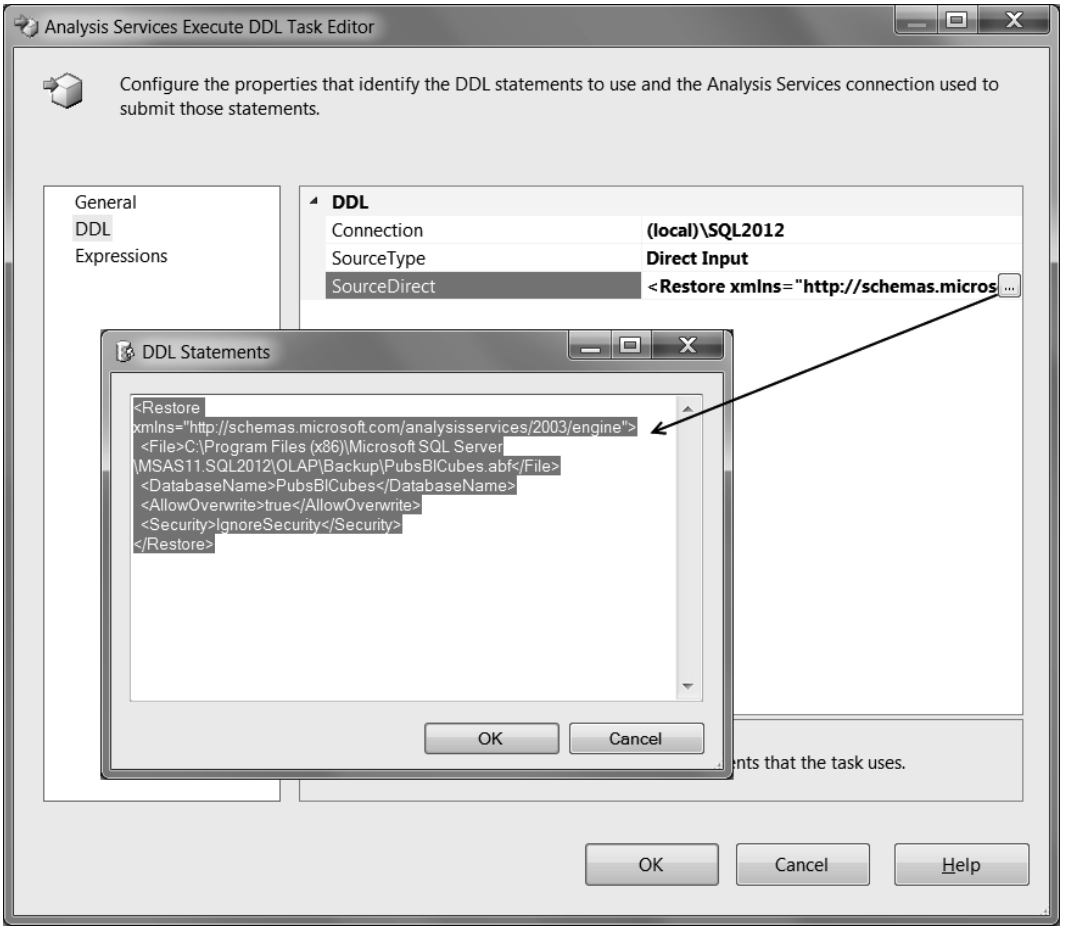


Figure 19-3. *Configuring an Analysis Services Execute DDL task*

Once you have configured the SSIS task, execute it from the command line just as you did the ETL code (Listing 19-6), and of course, you could place this code in the same batch file.

Listing 19-6. Executing the SSIS Package to Restore an SSAS Database

```
CD C:\Program Files (x86)\Microsoft SQL Server\110\DTS\Binn
DtExec.exe /FILE \\RSLAPTOP2\PubsBIProdFiles\RestorePubsBICubes.dtsx
```

Deploying the SSRS Reports

Once again, your choices are to manually deploy the report files to the SSRS web service or to automate the deployment using a command prompt utility. The SSRS reports can be deployed to the web service using a utility called `RS.exe`. Unfortunately, it is the least friendly of the deployment options. The utility requires that you create a VB .NET file that tells the web services what to do. This code is placed in a file with an `.rss` extension. For instance, code used to deploy a report from the network share to Randal's SSRS server looks like the code shown in Listing 19-7.

Listing 19-7. Code to Place in an `.rss` File to Upload an SSRS Report

```
' Deploy SSRS Report from a Command Line:
' RS.exe -i deploy_report.rss -s http://rslaptop2/ReportServer_SQL2012/ReportServer
'

Dim strPath = "\\RSLAPTOP2\PubsBIProdFiles\"
Dim strReportName = "SalesByTitles"
Dim strWebSiteFolder = "/PubsBIReport"

Dim arrRDLCode As [Byte]() = Nothing
Dim arrWarnings As Warning() = Nothing
Public Sub Main()
    Try
     'Read the RDL code out of the file.
     Dim stream As FileStream = File.OpenRead(strPath + strReportName + ".rdl")
     arrRDLCode = New [Byte](stream.Length) {}
     stream.Read(arrRDLCode, 0, CInt(stream.Length))
     stream.Close()

     'Upload the RDL code to the Web Service
     arrWarnings = rs.CreateReport(strReportName, strWebSiteFolder, True, arrRDLCode, Nothing)
     If Not (arrWarnings Is Nothing) Then
         Dim objWarning As Warning
         For Each objWarning In arrWarnings
             Console.WriteLine(objWarning.Message)
         Next objWarning
     Else
         Console.WriteLine("Report: {0} published successfully with no warnings", REPORT)
     End If
    Catch e As IOException
     Console.WriteLine(e.Message)
    End Try
End Sub
```

After you have created the `.rss` file, execute it with the `RS.exe` command prompt code as shown in Listing 19-8. This, too, can be placed in the same batch file that deploys the data warehouse, ETL package, and SSAS database.

Listing 19-8. Code to Place in an `.rss` File to Upload an SSRS Report

```
CD C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn

RS.exe -i \\RSLAPTOP2\PubsBIProdFiles\DeploySalesByTitles.rss
-s https://rslaptop2/ReportServer/ReportServer
```

Using SSIS Packages Instead of Batch Files

SSIS includes the ability to run command prompt code using an Execute Process task. Since running command prompt code is the core of what a batch file does, this feature gives you an alternate way of creating deployment automation.

For example, you can do the following:

1. Create an Execute SQL task to run the code needed to restore the database from the network share.
2. Add an Execute Package task to run the ETL package.
3. Add an Analysis Service Execute DDL task to restore the SSAS database.
4. Add an Execute Process task to call `RS.exe` and upload the report.

Figure 19-4 shows an example of this package.

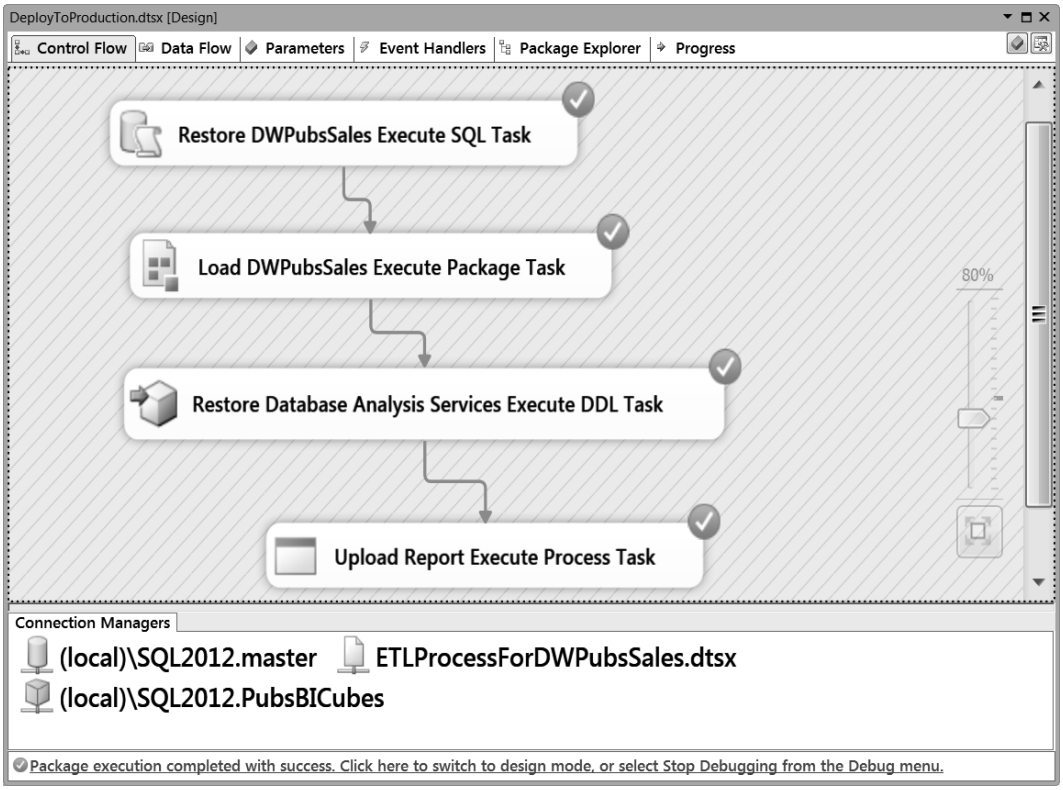


Figure 19-4. *Creating an SSIS package for deploying your BI solution*

No matter which method you choose, make sure you set aside plenty of hours for not only developing the deployment files but testing them as well. This process can be exhausting, because you will have to deal with many different departments to get permission to access the resources you need and many different computer configurations. Plan accordingly!

Release Documentation

Releasing your solution requires more than simply implementing it. Developer documentation such as SDKs and dev specs are also required. Additionally, users need documentation to teach them how to use the software you developed for them.

Programmer writers (for release documentation) and technical writers (for user documentation) who know their job very well will be able handle the majority of the documentation for you.

Ideally, they will have been working with your team for some time and are familiar with the ins and outs of your BI solution. Even better, they have been in on the project from the beginning and have been present for the team meetings that determined the direction that your BI solution would take.

If this is the case, then congratulations, you did it right! You will likely have much, if not all, of your documentation ready to be released at the same time that your software is released. This contributes to your professionalism and helps keep your client happy.

SDKs

If you write your software correctly, it will be usable for many applications. A software development kit (SDK) is a set of documents that includes the following:

- Code examples
- Comments within the code
- Descriptions of procedures
- Parameters
- Return values
- Instructions describing how to use the code
- Links to related SDK subjects (depending upon how they are stored)
- License and legal information declaring how and where it can be used (free or public)

SDKs can also be written for code that is not included in the official release but could be useful for future versions. The more details included, the better.

Typically, these documents are stored together online to form a library. They can also be stored as a simple text document. However you choose to store it, your SDK library should include a complete list of all of the SDKs organized under different topics (such as stored procedures, custom DLLs, or data warehouse tables) for convenience and searchability.

■ **Note** In 2010, the Microsoft Interoperability team released an SDK describing how PHP (an open source competitor to ASP.NET) can interact with Microsoft's SSRS web service. It is a good example of how creating SDKs can enhance your software, because this SDK has gone a long way toward making SSRS the most flexible and robust reporting software in the industry. It is also a good example of what a professional SDK looks like. For more information, go to `http://ssrsphp.codeplex.com`.

SDKs can also include flow charts or visual layouts that demonstrate the structure of a data warehouse, for example. To keep the documentation professional, some companies choose to set a standard for how the documentation is recorded that can be kept on the main page and can help developers understand how the SDK is read. This can include formatting for text or other standardized information. Here are some examples:

- Standard fonts
- **Bold text**
- *Italics*
- Hyperlinks
- Date indicators such as mm/dd/yyyy or yyyy/mm/dd, and so on
- <Inserts>
- Value type indicators such as Bool vs. Boolean
- Separation characters for *either* | *or* choices
- Omitted portions

The following SDK example is created from a stored procedure we used in Chapter 13. Its purpose is to select report data.

AN EXAMPLE SDK

Name: pSelQuantitiesByTitleAndDate

Title: Stored Procedure for Sales Quantities by Title and Date

Description: Reporting stored procedure that returns total quantity, the overall average quantity, and a KPI value based on the comparison between the total quantity and the overall average quantity

Links: `http://MyCompany/BIWebSite/Dev/pSelQuantitiesByTitleAndDateInfo.aspx`

Parameters:

Name	Type	Example	Description
ShowAll	nVarchar(4)	True	Show all rows of data if true
StartDate	DateTime	01/01/1990	Starting date range
EndDate	DateTime	01/01/2100	Ending date range
Prefix	nVarchar(3)	A, AB, or ABC	0 to 3 characters used before wildcard search symbol

Outputs:

Name	Type	Example	Description
PublisherName	nVarchar(50)	New Moon Books	Name of publisher
Title	nVarchar(100)	Life Without Fear	Title of book
TitleId	nVarchar(6)	PS2106	Natural ID used for a title
OrderDate	Varchar(50)	01/13/2011	Date of order
Total for the Date by Title	Int	108	Total quantity based on title and date
Average Qty in the FactSales Table	Decimal or Float	53.512	Total overall average quantity from the entire table
KPI on Avg Quantity	Int	-1, 0, 1	Total overall average quantity compared to the sum of that title (filter by search) and categorized into three groups based on the following: -1 = Quantity is lower than average by 5 0 = Quantity is between 5 less to 5 more of the average, inclusive 1 = Quantity is more than average by 5

Source Code:

```
CREATE PROCEDURE pSelQuantitiesByTitleAndDate
 (
  -- 1) Define the parameter list:
  -- Parameter Name, Data Type, Default Value --
    @ShowAll nVarchar(4) = 'True' -- 'True|False'
  , @StartDate datetime = '01/01/1990' -- 'Any valid date in the mm/dd/yyyy format'
  , @EndDate datetime = '01/01/2100'  -- 'Any valid date in the mm/dd/yyyy format'
  , @Prefix nVarchar(3) = '%' -- '0 or more characters'
 )
AS
BEGIN -- the body of the stored procedure --
  -- 2) Create and Set the @AverageQty to get the overall average of sales qty.
  DECLARE @AverageQty int
   SELECT @AverageQty = Avg(SalesQuantity) FROM DWPubsSales.dbo.FactSales

  --3) Get the Report Data with this select statement
  SELECT
```

```
    [DP].[PublisherName]
   ,[Title] = DT.TitleName
   ,[TitleId] = DT.TitleId
   ,[OrderDate] = Convert(varchar(50), [Date], 101)
   ,[Total for that Date by Title] = Sum(SalesQuantity)
   ,[Average Qty in the FactSales Table] = @AverageQty
   ,[KPI on Avg Quantity] = CASE
    WHEN Sum(SalesQuantity)
     between (@AverageQty- 5) and (@AverageQty + 5) THEN 0
    WHEN Sum(SalesQuantity) < (@AverageQty- 5) THEN -1
    WHEN Sum(SalesQuantity) > (@AverageQty + 5) THEN 1
   END
   FROM DWPubsSales.dbo.FactSales AS FS
   INNER JOIN DWPubsSales.dbo.DimTitles AS DT
     ON FS.TitleKey = DT.TitleKey
   INNER JOIN DWPubsSales.dbo.DimDates AS DD
     ON FS.OrderDateKey = DD.DateKey
   INNER JOIN DWPubsSales.dbo.DimPublishers AS DP
     ON DT.PublisherKey = DP.PublisherKey
   WHERE
     @ShowAll = 'True'
     OR
     [Date] BETWEEN @StartDate AND @EndDate
     AND
     [TitleId] LIKE @Prefix
   GROUP BY
      DP.PublisherName
      ,DT.TitleName
      ,DT.TitleId
      ,Convert(varchar(50), [Date], 101)
   ORDER BY DP.PublisherName, [Title], [OrderDate]
 END -- the body of the stored procedure --
```

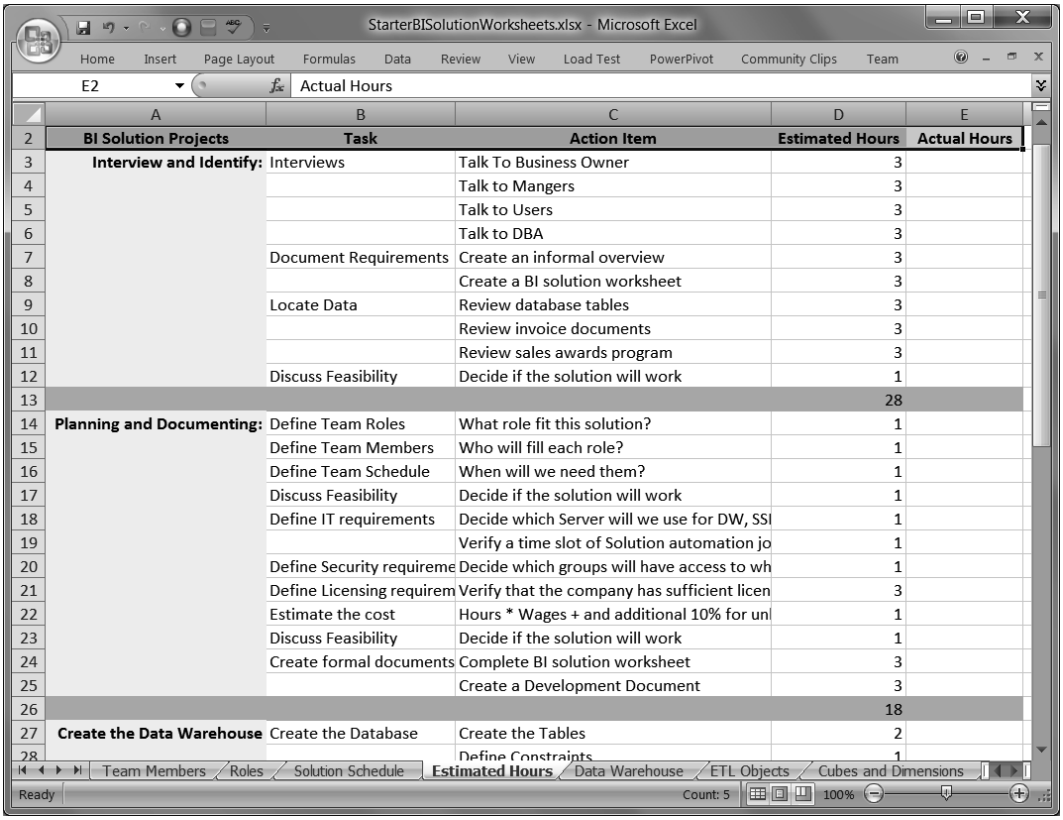
In production code, comments are often removed from within the procedure, but in the SDK these comments can be left in, and additional descriptions, instructions, and comments are written out in full sentences after each section, as you can see in the far-right column of each table.

Objects such as views, functions, and parameters should be stored together in groups and in alphabetical order within those groups. We chose to use a table format for organization, but tables are not required. This stored procedure was already titled pSelQuantitiesByTitleAndDate within the BI solution. But, for better searchability, we included a more human-friendly title: Stored Procedure for Sales Quantities by Title and Date. For additional organization, we suggest listing this type of SDK with other stored procedures that select data within the library.

Any additional information for how this procedure is used including any links to other topics should also be included on the same page. The idea is that if a developer adds to your process or uses these constructs, they are able to see what they can work with.

Developer Specifications

Developer specifications (*dev specs*) are documents that tidy up the completed work by listing what has been done and any important notes that document pertinent information. In Chapter 3 we worked with an Excel spreadsheet titled StarterBISolutionPlan that outlined the official development specifications (Figure 19-5). This is an informal version, and if written well enough, for small projects this may be sufficient for your needs.



	A	B	C	D	E
2	**BI Solution Projects**	**Task**	**Action Item**	**Estimated Hours**	**Actual Hours**
3	**Interview and Identify:**	Interviews	Talk To Business Owner	3	
4			Talk to Mangers	3	
5			Talk to Users	3	
6			Talk to DBA	3	
7		Document Requirements	Create an informal overview	3	
8			Create a BI solution worksheet	3	
9		Locate Data	Review database tables	3	
10			Review invoice documents	3	
11			Review sales awards program	3	
12		Discuss Feasibility	Decide if the solution will work	1	
13				28	
14	**Planning and Documenting:**	Define Team Roles	What role fit this solution?	1	
15		Define Team Members	Who will fill each role?	1	
16		Define Team Schedule	When will we need them?	1	
17		Discuss Feasibility	Decide if the solution will work	1	
18		Define IT requirements	Decide which Server will we use for DW, SSI	1	
19			Verify a time slot of Solution automation jo	1	
20		Define Security requireme	Decide which groups will have access to wh	1	
21		Define Licensing requirem	Verify that the company has sufficient licen	3	
22		Estimate the cost	Hours * Wages + and additional 10% for unl	1	
23		Discuss Feasibility	Decide if the solution will work	1	
24		Create formal documents	Complete BI solution worksheet	3	
25			Create a Development Document	3	
26				18	
27	**Create the Data Warehouse**	Create the Database	Create the Tables	2	
28			Define Constraints	1	

Figure 19-5. *The StarterBISolution Excel spreadsheet*

As the solution planning progresses or as the project grows, this informal Excel file is used as an outline to a more official document called a Solution Development Plan. This plan will become the bible to your BI solution. It is a living document that is updated on a regular basis.

This document is created in Microsoft Word rather than Excel, and it lists more details about the solution objects. This document can be used as a starting point for your dev specs. The difference between the two is that one was made during the development cycle and represents the current state (any time during development) of the solution, while the other represents the final outcome of the solution. Figure 19-6 shows how similar they look, which is intentional.

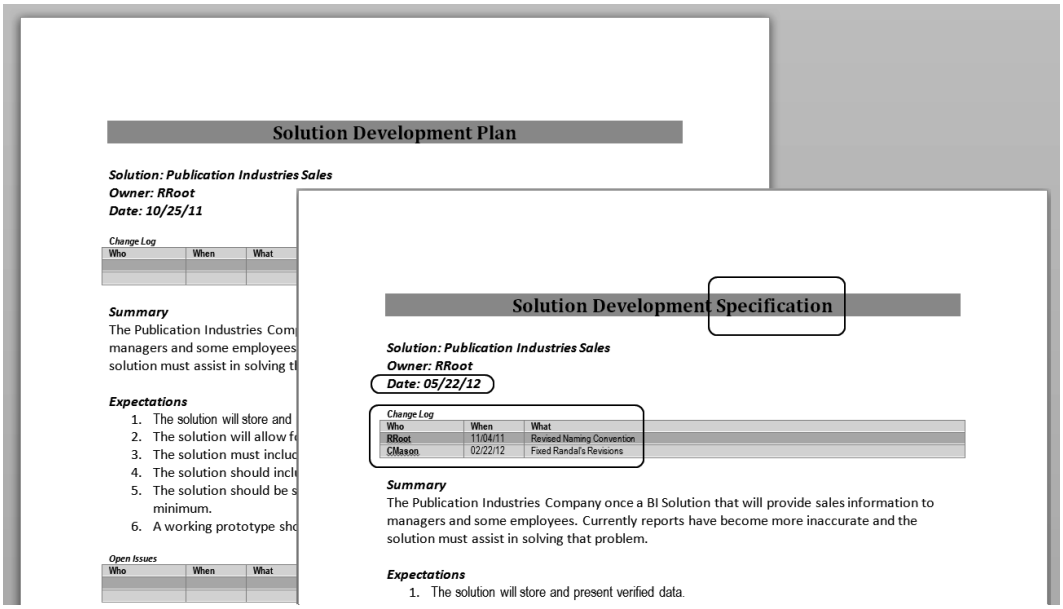


Figure 19-6. *Comparing the Solution Development Plan to the dev specs*

When each version of the solution is complete and you start work on the next version, properly written dev specs will get your team up to speed. Portions of this document can also be included as part of the SDK document, because the audience for the two documents are complementary.

■ **Tip** Some companies fly employees in from all over the world to be trained to create these types of files. You do not have to go that far, but documents of this type are an important indication of your level of professionalism.

Dev specs are a blueprint to what was created and are given to the customer much like a car manual is included with a new car. The SDK is given to other developers who will work with your solution, much like a repair manual is used by mechanics. (You may not want to refer to your SDK as a repair manual! Hot-rod customization tool kit sounds much better.)

User Documentation

With all the documentation that needs to be included, we cannot forget that we have new users who have never seen your BI solution before and need to be instructed on how to use your program.

Just as there is more than one way to peel an apple, there is more than one way to train a user. Some methods include the following:

- Developing a video demo for online training
- Training the managers who in turn are responsible for teaching it to their employees
- Sending someone from your team to set up one or more training sessions

Whichever method you choose, this step is vitally important for the success of your BI solution. User manuals and help documents are to be used as references. Training your users yourself gets the system up and running immediately. The success of the user is what determines the effectiveness of your solution, as well as how likely you will be involved in future versions of the current BI solution you developed.

Style Guides

Regardless of when (or if) you get a professional technical writer involved in the project, your tech writer will need to know what is expected of your documentation, not just how your BI solution works. If your company is large or has been around for a while, you may already have a style guide that standardizes the writing style of your solution. Style guides are highly recommended and can be very helpful in keeping consistency when referencing your company and specific items within each of the solutions you develop.

When you hear the term *style guide*, you may automatically think of commonly known style guides such as *The Chicago Manual of Style*, *The Associated Press Stylebook*, Strunk and White's *The Elements of Style*, and perhaps Microsoft's *Manual of Style for Technical Publications*. This is not what we are referring to when we speak of developing a style guide for your company. We are referring to *house style.* A house style guide describes how items of note are handled within your documentation for your company, particularly when referring to your company name or items within your software that is unique to your company.

This type of guide is much more in-depth than listing the general fonts used within an SDK. It is specific to how your company is presented to the general public for the following:

- User documents
- Company websites
- Email
- Business cards
- Letterhead
- Contact information standards

Fonts and color schemes can also be standardized within your company house style guide.

■ **Tip** Creating your documentation by following a company-specific style guide adds to your final presentation and improves your professionalism to developers, to the client, and to the user.

If you do not already have a style guide, most tech writers can develop one for you while they prepare your documentation. A style guide is an alphabetical list of *how-tos* that include descriptions and examples of how to handle company information. A style guide is another one of those living documents that are ever-changing with your company. And as you change how you do things or develop new ways of keeping consistency, that allows room for new topics to be added at any time.

An added bonus to developing an in-house style guide is that if you work with more than one tech writer, you can keep your documentation consistent with a professional presentation, regardless of who writes it.

User Manuals

The better the instructions on how to use your BI solution, the more successful it will be, which is why the user manual is so important. Keep in mind that your client may hire new employees who were not on the team when

hands-on training became available. Therefore, your user documentation needs to be thorough and accurate and written with very basic terminology that anyone can follow.

Many books are out there that cover precise details on how to write professional user manuals. Your team's technical writer will have this base covered. If, however, your BI solution is created for a small company, the solution itself is very simplistic, or you simply do not have the budget for a large team, then you may not have the luxury of hiring a tech writer to draft a user manual for you.

Let's look at some general guidelines on how to draft a simple user manual that your client can refer to, without going into the extreme depth and detail that typical technical writing books cover.

The Anatomy of a User Manual

The goal of any user manual is to give information that is easy to find and provides a step-by-step guide on how to perform an action. User manuals are usually divided into subjects and often include illustrations and numbered steps to follow.

A user manual is not the same as a trade book, such as the one you are reading now (although the exercises within this book are close to an example of how a user manual should read). Users of instruction manuals will not read each page, paragraph after paragraph. They want to get right to the information, gather it with as little reading as possible, and get on to using the solution itself.

Let's take a look at the anatomy of a user manual. As you can see in Figure 19-7, the layout should be easy to read, and the structure should be consistent. A good user manual will include *signposting* (guides to helping the user get to the information they need) to allow information to be gathered at a glance.

- *Main title*: Clearly indicates which software the manual is written for on the front page or cover.
- *Table of contents*: Gives subject locations at a glance including accurate page numbers that each subject can be found on and sometimes indicates the number of pages a subject spans within the manual.
- *Page numbers*: Even if the manual is short, page numbers are still necessary.
- *Page headers*: Indicates the topic at a glance in a standard place on every page.
- *Spot color, shading, or icons*: Visual aids that draw the eye to specific information.
- *Subject headings*: These must correspond with the titles in the table of contents, must be listed at the top of each page before a new subject, and must accurately define the subject so that users can get to the right place at a glance.
- *Figures (illustrations)*: These must genuinely aid the user rather than confuse them further, and they must be properly numbered and labeled and referred to within the text nearby.
- *Step-by-step instructions*: These must be accurate, written simply with easy-to-understand terms, and highly descriptive, logical, and thorough; they are usually written in the imperative (without using the word *you*).

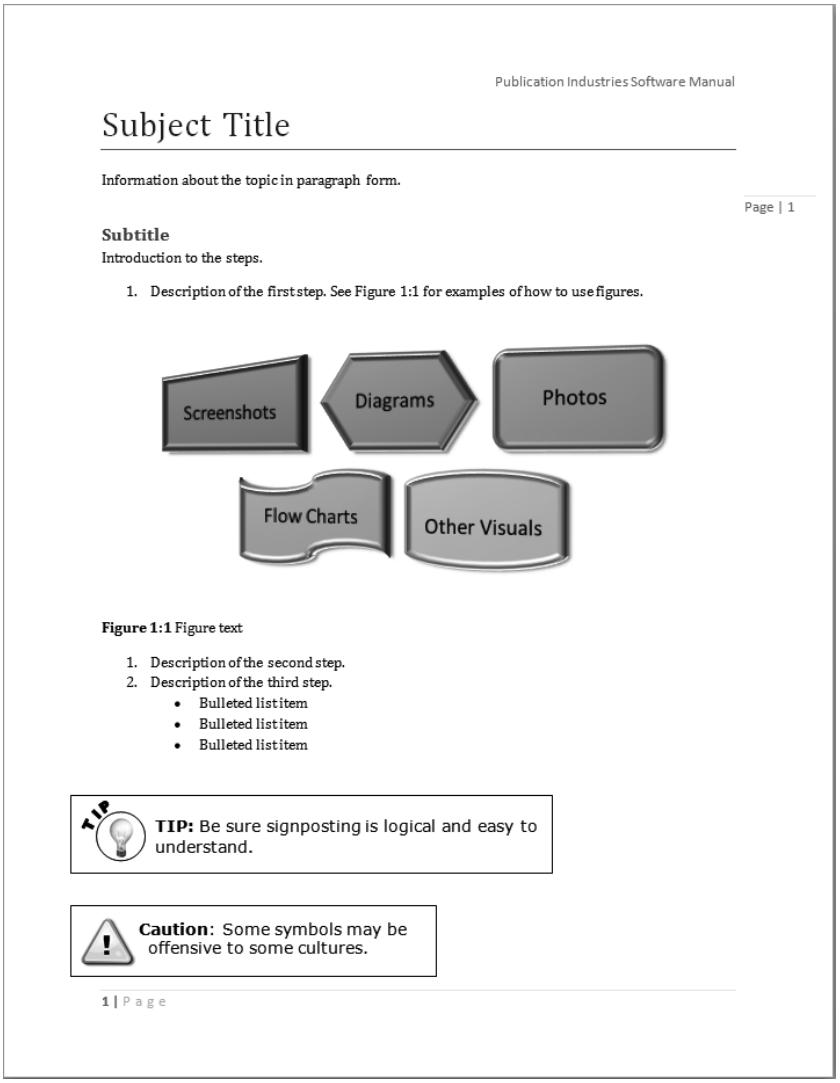


Figure 19-7. *A layout example for a user manual*

Keep in mind that every signpost must have a purpose and communicate the proper information. If it is a warning, put up a proper warning icon. Do not use a rainbow, the Ace of Spades, or Happy Bunny because it looks pretty or adds comedy.

Do not make arbitrary changes in color to specific backgrounds or titles that are not purposeful. This confuses the reader who will spend more time wondering if they somehow stumbled into the wrong section, rather than quickly getting the information and moving on.

Subject Headings

Subjects are given clear headings that are easy to find, and each step or procedure contains a detailed enough explanation so that a new user, who is unfamiliar with the program, can find their way around while following the user guide.

Step-by-Step Instructions

The step-by-step instructions are the largest portion of a user manual. The very first item to consider when writing instructions is to always, always, always consider your audience.

For a BI solution, who are your users? They will typically be your client's employees. These employees are people. How much do you know about these people?

Table 19-2 lists some example items to consider when determining who your audience is. You will likely have more items to add to this list based upon the type of company you are creating your solution for. You may want to ask your client about the type of employees they hire to work for their company.

Table 19-2. *Who Is Your Audience?*

What to Consider	How It May Be Evaluated	How It Affects Your Documentation
Age	Employees may range from teenagers in high school to senior citizens.	Determines the age level the instructions should be written to, and how big the font sizes should be.
Ethnicity	They may speak English as a second language.	Translations may be necessary. Fewer words may be used with more illustrations. Documentation may require using words that are simple and less technical.
Geographic location	Employees may reside in another country.	Translations may be necessary. Specific words or colors can be offensive or have additional meanings to residents within other countries.
Education levels	Your client's company may employ special-needs employees or some who have a lower education level.	Fewer words may be used with more illustrations. Instructions may require using words that are simple, contain fewer letters, and are less technical.
Level of experience with computers	Some may have never used a computer before.	Even the most basic of instructions may be necessary, such as how to close a window.
Color blindness	Some of your signposting may be invisible to 10% of the male population.	Black-and-white versions, versions for color-blind readers, additional signposting, or more specific signposting may be required.

The information within each subject may begin with a brief introduction to a topic before getting to the steps required to perform an action. It must refer to the illustrations where appropriate and optionally end with a brief conclusion of the topic.

The body text of the user manual should not be text heavy, meaning it should not contain paragraph after paragraph of information. Writing a few paragraphs about the program itself can be valuable, but be sure that the majority of the text is broken up into categories such as numbered lists, picture illustrations, bulleted lists, a sidebar, a note that has a different format as the rest of the text, and so on. A very well-written user manual can give the necessary information at a glance.

■ **Tip** At a glance, at a glance, at a glance…notice a pattern? The faster users can glance at the instructions, gain the information, and move on, the more productive they are, the happier your client is, and the more effective your BI solution will be.

You may have heard the *hand* rule. Although this is not the official name for it, the essence of it is this: if you can put your hand on the printed page in any place and your hand does not touch something other than paragraphs of text, then your text is too heavy. The reason why this is important is because users cannot find information easily in a block of heavy text. When the text is broken up, your users can navigate through the instructions much more easily, enabling them to visually skip around on the page and gather information faster and more readily find what they are looking for.

As we stated earlier, our exercises are similar to how a user manual's instructions should appear. The voice of most of our exercises, however, is not always in the imperative in this book, which is the recommended voice for a user manual's instructions. (One exception to this is the next exercise. Exercise 19-1 is written in the imperative and has the proper format and voice for a user manual.)

Here is an example of an imperative voice and a nonimperative voice:

- *Imperative voice:* "Click the OK button."
- *Nonimperative voice:* "After that, you can then click the OK button."

The imperative voice begins with a verb, is in the form of a command, and the word *you* is implied. The second example is simply a statement and is less effective for instructional documentation.

Keep the instructions precise and properly spell the titles of the items you are referring to within your program, paying attention to capitalization. If the button title is OK, for example, then the instructions should not say, "Click the Okay button." (In other words, Okay should not be spelled properly when the *OK* on the button itself is not spelled out properly.) Nor should the instructions be "click ok" or "click Ok" when both letters of *OK* are capitalized within the program. This may sound nitpicky, but you would be surprised how users can get lost when the instructions within your manual are not capitalized exactly the same way that they are within the program. (You may not be so surprised by this, however, if we have made that mistake within this book and you found yourself lost for that very reason!)

Your instructions should also be free of filler words. Descriptive words are helpful, but get to the point of what needs to be said as quickly and concisely as possible without losing the reader.

Figures

Whoever said a picture is worth a thousand words had the soul of a writer—perhaps even a technical writer. Figures are the illustrations or pictures in your manual and can be one of the most important signposts you use. They save the writer a lot of words explaining what to look for within the program. Figures help tell the story and aid the user by giving clear and precise visual cues as to what they are to click or where an item within your program can be found.

If you have ever shopped at Ikea, you may have noticed that some assembly-type user manuals do not contain words. The illustrations themselves are the instructions. Sometimes these illustrations are easy to follow, and sometimes they are worthless and get thrown away with the box the product came in. Make sure that the

pictures you include are genuinely helpful to the user, instead of confusing them further. Sometimes this means that sections of the illustration should be highlighted and/or numbered to properly indicate what is intended by the figure.

Additionally, figures should be numbered accurately and referred to properly within the text. Figures should never be pictured without written instructions indicating the purpose of the illustration, unless, of course, you are writing instructions for Ikea.

Figure Captions

Illustrations should always be captioned with a figure number as the title and a brief description of what the figure is indicating. The description of the figure is typically more like a label, and not a full sentence, so it does not need a period at the end of the caption.

Figure numbers are a very important aspect of your user manual's instructions. Each figure should be given two numbers. The first portion of the number indicates whether it is the first, second, third, and so on, subject within the manual. The second portion indicates the order in which the figure appears.

Take a look at the format used for the figure numbers in this book. It does not matter if you format them exactly as you see here or if you use a dash or another commonly used format.

■ **Tip** If you gave every figure a single number starting with 1, by the time you get to the very end of your user manual, you may then realize you need to add a new figure somewhere near the beginning of your text. If so, you will be forced to renumber every single figure within your entire manual after that change. When you give your figures two numbers, the first for the chapter (or subject) and the second for the sequence within that chapter or subject, you only have to renumber the rest of the figures within the chapter (or beneath that subject).

User Manual Testing

Just as your BI solution needs to be tested to be sure it works, your user manual needs to be tested to make sure it works as well. If you are the original writer of your user manual, you are unlikely to recognize missed steps, wrong information, or confusing instructions. This is because you already know how it all works, and your mind fills in the blanks.

Because of this, we highly recommend that you have your documentation tested by someone who is unfamiliar with your program and is detailed minded enough to point out where they got lost and even make suggestions on how to fix it. And, just like your BI solution, user manuals can benefit from user feedback after the initial release so that new versions can be drafted with more precision.

All of this information about how to compose a user manual may sound simple enough, but at this point it is still just hypothetical. To truly gain an understanding, you need to try it.

This next exercise is a really fun, out-of-the-box way of learning how to effectively compose written instructions. It is based on an exercise from a college-level technical writing course.

EXERCISE 19-1. CREATING A CHILD'S PLAY USER MANUAL

This fun and simple exercise is a hands-on example of how to create a user manual and have it tested by a team of users for accuracy and simplicity.

As humorously as this exercise may be written, the understanding that is gained from performing it is invaluable for learning how to communicate instructions accurately and effectively. This exercises will not

only aid in creating user manuals, but is helpful for all types of written instructions including help files, SDKs, and other types of instructional documentation.

Required items to perform this exercise:

- One beginner-level children's construction set such as a magnetic building set, a set of building blocks that contains various shapes (not just cubes), a set of Tinkertoys, or a set of LEGOs that contains various shapes (not just cuboids).
- A digital camera (or a camera phone)
- Several pens (or pencils) for each of the testers
- Several sheets of paper
- A computer with word processing software
- A printer
- A small group of detailed-minded testers with a willingness to follow instructions and give written feedback
- A door (to leave one's ego behind)

Build Something

Build something interesting and creative with the construction set.

1. Select 10 to 15 pieces from the construction set. Include at least three different shapes and sizes.
2. Build a unique structure using all of the selected construction set pieces.
3. Give the structure a name.
4. Take one or more photos of the structure with the digital camera or camera phone.

Develop the Instructions

Create documentation with instructions on how to re-create the structure.

1. Hand-write a list of the items required to create the structure with a pen (or pencil) and paper. Include names of each item (you may need to make up a name) and proper descriptions, such as length, shape, width, color, or whatever else is required to describe the piece. (Do not cheat and look up this information on the Internet or in the toy's instruction manual.)
2. Hand-write detailed instructions on how to build the structure. Be sure to number each step, use correct grammar, and punctuate sentences properly.

Create the User Manual

Compose a simple instruction manual and print out several copies for the testers.

Note: Do *not* include photo illustrations within the instructions, the items list, or anywhere else within the Child's Play User Manual.

1. Create a professional-looking document with the computer and word processing software. Refer to the text within this chapter for help if necessary.
2. Title the documentation with a simple yet descriptive title that includes the name of the structure. Example: How to Create a <insert structure name here>.
3. Write a simple introduction that describes the structure to be built.
4. Type the hand-written instructions as accurately as possible. Leave space between each of the instructions for notes to be recorded by the testers.
5. Optional: Write a concluding sentence or paragraph (that is not numbered) after the instructions.
6. Proofread the instructions, and verify that they are accurate. Make a printed master copy of the instructions and print one extra copy for each tester.
7. Disassemble the structure.
8. Combine the pieces required to build the structure with the rest of the pieces in the construction set. If the original construction set is not a beginner level and contains thousands of pieces with many different shapes and sizes, combine the pieces with approximately 20 additional pieces from the original set.
9. Place the combined pieces in a container.

Note: Some construction sets come with illustrations of the pieces contained within their original containers. If so, place the pieces in a different container so that the testers are not given extra clues on how to construct the structure.

Test the Instructions

Acquire testers who have never seen the final structure.

Note: One tester may be enough, but a small group of testers are likely to make this exercise more fun and will be able to give more perspective on the experience, exponentially increasing the learning experience. The testers can work together or individually if time permits.

1. Give the container containing the unconstructed pieces to the testers.
2. Give the testers each a copy of the instructions and a pen or pencil to take notes.
3. Leave the room.
4. Give the testers about 20 minutes to construct the structure and to take notes on the experience. (For individual testers, allow each to proceeded through the Test the Instructions steps individually.)

The Moment of Truth

Check out the final results.

1. Place any residual ego behind the door when returning to the testers.
2. Compare the structure that the testers built with the photo image.

Note: If the photo image is different from the tester's structure, consider this exercise to be an important learning experience rather than a failure.

3. Review the notes that the testers made and discuss their experience in creating the structure.

This exercise was an outside-the-box skill-building activity for developing user instructions and having them tested for accuracy. For some, this may have been a very humbling experience.

Help Files

Help files (aka help docs) may be included as part of the solution. Not every BI solution is complex enough to require a help file, but depending upon how complex your BI solution is, it is something you may want to consider.

Your help documents should contain four key items:

- A hyperlinked table of contents
- An index
- A search feature
- Specific instructions for each topic

Help files can be very simple to develop. The topics that are covered often correspond with the user manual, but they are often written with more brevity. The subjects for your user manual are a good place to start when developing help files.

PDF Files

One way of handling help files is to save them in a PDF file. PDF files are a format that is easy for users to navigate but not necessarily easy for them to change. Many companies work with specific programs to aid in the development of PDF files that can simplify (or possibly complicate) the process. But this can get expensive if you are working on a budget and do not already own this software.

Simple PDF files can be created in Word 2010. To do so, write up the help instructions as necessary; then access the File tab of the Ribbon and select the Save As option. This will bring up the Save As dialog window. Save the document as a PDF file using the Save File Type dropdown box (Figure 19-8).

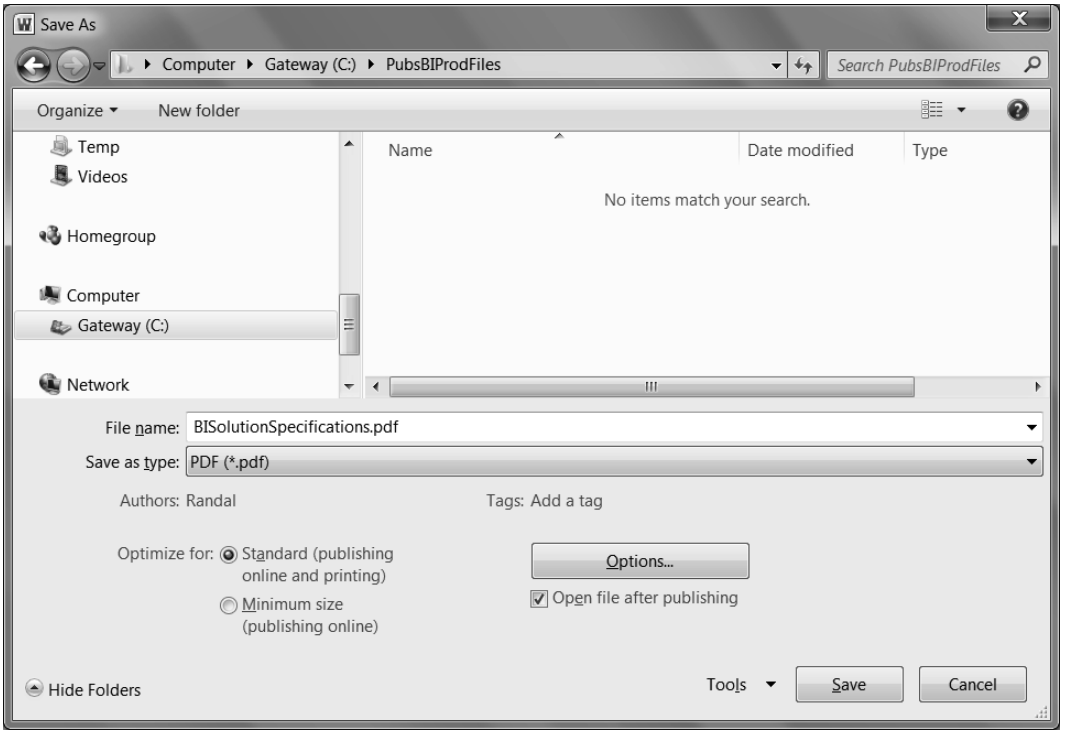


Figure 19-8. *Saving a Word document as a PDF file*

HTML Files

Microsoft Word also allows you to save a help document as an HTML file. The HTML file that is created is not the most compact or even most compatible HTML code you will ever find but is simple and effective enough for small solutions.

When you elect to save the Word document as an HTML file, you have the option to save the document as a single `.mhtml` file that includes embedded images or as a standard HTML file with all of the images placed in a separate subfolder.

User Training

A manual is good to have, but hands-on training has proven to be more effective with some users. Often this is done in person, but it can also be done by creating a video demonstrating how to use the solution.

Training your users gets the system up and running immediately, while user manuals provide a reference to use after training. The effectiveness of your user is what determines the success of your solution.

When training your users, it is helpful to develop an outline of what needs to be covered so that nothing that is vitally important is missed. Be sure to keep in mind that this system is extremely familiar to you, the developer (or project manager or whatever your title is), but it is new and very unfamiliar to your user. If you are there in person, be sure to allow for time to answer questions. Keep your instructions as simple as possible and perhaps show them where they can find this information in the user manual later, when you are no longer with them to guide them through.

While developing and uploading videos are outside the scope of this book, we recommend that if you choose this method, also ensure that the audio is clear and that the steps you take are easy to follow. Keep in mind that you may want to keep your videos short to solve connection speed and playback issues.

Say Thank You

One last personal touch that we recommend to help you stand out is to send a thank-you to your client. This can be simple and generic and very cheap, such as an email or a thank-you card. If you really want to keep working for that client and prefer to make it more special, you may want to step it up and give your client a gift such as a gift basket, some chocolates, or even a bottle of wine.

Order it in advance with your company name and/or logo, and not only do you have an additional professional touch, but the expense is now a tax write-off!

Moving On

The final step to completing your BI solution is to take what you have learned from the current version and apply it to your future versions. Make sure you consider items such as testing bug reports, questions that users are asking, and any other feedback that you have collected.

In this book, you have seen how to design and implement a BI solution from start to finish. Along the way, you have been introduced to a number of technologies that are included with Microsoft SQL Server. What happens now is up to you.

You might decide to take what you know about SQL Server, SSIS, SSAS, and SSRS and expand that knowledge by reading the books that we have referenced in each chapter. There certainly is a lot more to learn about each one of the servers, and most developers will find that specializing in only one of them keeps them quite busy.

You might realize that you are more interested in the overall process of creating and managing a BI solution and want to hire developers to help you with the implementations. If that is the case, you now know the basics about how the different servers operate, which will make you more effective in working with those developers.

You also might find that you still need more practice before you venture out into the real world of BI solutions. In that case, make sure you try the "Learn by Doing" exercises we have included on the companion website: `www.NorthwestTech.org/ProBISolutions`.

Be aware that the world of business intelligence is one of constant change. Just as you have completed one version of a BI solution, you will find that the needs of the user, and possibly the technology that supports the solution, have changed. Although this can be one of the most frustrating aspects of being a BI solutions developer, it is also the most exciting. In the world of BI, the development cycle never truly ends.

What's Next?

After reading this book and performing its exercises, you now have enough knowledge to create your own BI solutions. If you have not done so already, we recommend working through the "Learn by Doing" exercises included at the end of each chapter.

It can be difficult to break into a new area without experience, but you cannot get experience if you cannot perform the work! For this reason, we recommend gaining experience and recognition by working on department-level BI solutions. A department-level BI solution is a smaller solution specific to a single department or a small company that you already work for, or it may be a gift to a local social group, organization, or church. This type of experience is often enough to get you started. This is especially true if you are not currently considered a BI professional.

No matter what your situation, to become more proficient at creating BI solutions, you must continue building them.

It can be difficult to know where to start, particularly if your work is volunteer, or the company or group you are creating it for does not know how BI solution reports can benefit them. This unknown is likely to take you back to the interview process at the beginning of this book. If that is not possible or if it adds too much pressure to what you intend to do, perhaps you can simply consider their needs on your own. You can create reports for nearly anything. Get creative. Ideas on what to create reports on may include the following:

- Attendance at meeting and events
- Frequency of member visits to needy or elderly members of a church
- Animal rescues with an organization like the Humane Society
- Customer age groups for advertising purposes within your current company
- River pollution levels for an organization like American Rivers
- Who has received training on the newest technology your company is promoting

Your options are unlimited. There are many reporting opportunities to choose from in our everyday lives all around us. Pick one or more items that interest you, and then open the book to Chapter 1 and apply the processes we have outlined to your own BI solution (Figure 19-9).

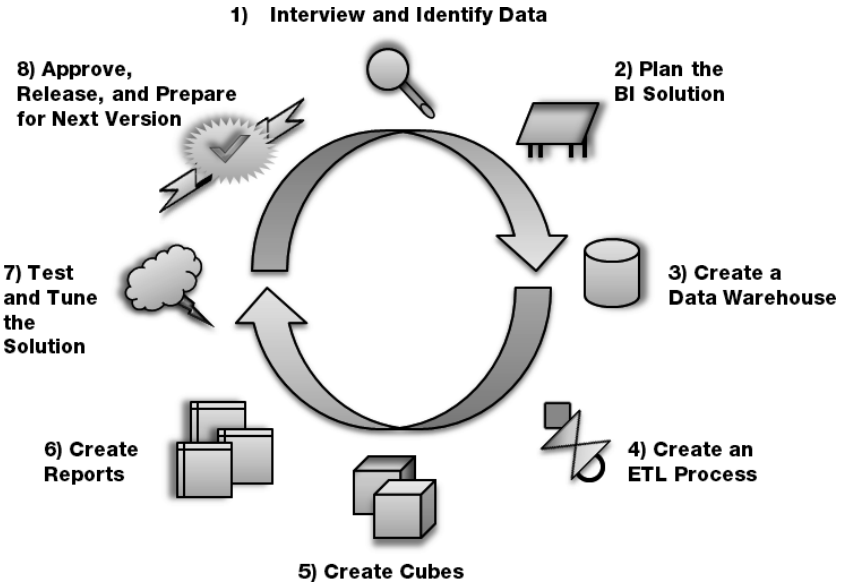


Figure 19-9. *Restarting the BI solution cycle*

With only a few successes, you will not just hope to be a BI professional, you will be one! And in the meantime, you help your co-workers and community by providing the benefits of a simple, quick, and effective BI solution.