Deployment Manual

For

Meghna Solar

(01/02/2015)

**Deploy ASP.NET Web Application on server**

Deployment refers to the process of copying an asp.net web application from the development system to the server on which the application will be run. There are several way we can deploy our web application

We can deploy ASP.NET Application in 3 different ways-

* **Leasing And Integrated Accounting System Deployment**
* **Precompiled Deployment**
* **Web Setup Project**

The choice of best deployment alternative depends upon particular need of each application. Leasing and Integrated Accounting System deployment is the easiest, and it is often used during development to create copies of application n different servers for testing purpose. For small application Leasing and Integrated Accounting System deployment may be the best choice.

Precompiled deployment has several advantages over Leasing and Integrated Accounting System deployment. Eg. Precompiled deployment is always gives better performance for the first users of the site at the same time it is more secure as we don’t need to copy our source code files on to server. If our application deployed on one or few servers then precompiled deployment is usually best choice.  
  
When we are going to deploy our application on number of servers then creating a setup program is a very handy tool. Although creating this setup program is much tedious and involves considerable working, the deployment from this setup program becomes very easier.

**Leasing and Integrated Accounting System Deployment**

To manually copy the files of an asp.net web site to a server. We can use the Leasing And Integrated Accounting System command from a command prompt. Then we can use IIS’s (Internet Information Server management console t o create a virtual directory that’s mapped to the directory that you copied the web site to.  
  
It is easier to create a batch file for the Leasing and Integrated Accounting System command. Then after we can run that batch file at given time we make changes to the application and want to deploy the updated code.  
  
We can also perform Leasing and Integrated Accounting System deployment from visual studio by using copy website command.

To perform Leasing and Integrated Accounting System we use copy web site command. This command lets us to copy website to file system, local IIS, FTP or remote IIS website. At the same time we can copy all or selected files.  
  
**How to use this command**

1. In visual studio open the website to be deployed and choose the website copy web site command.
2. Then click the connect button to display an open website dialog box that lets to choose the destination to copy the web site to.
3. Click the arrow buttons to copy the files from the source web site to remote web site.

**Publish Web Site**

The publish web site command lets us to precompiled an asp.net application and copy the precompiled assemblies to a target server. This is the easiest way to use the precompiled deployment feature.

* Deploys precompiled assemblies to the specific server
* Lets us to deploy the web site without source code files
* It is done either from publish web site command or from command prompt using the aspnet\_compiler command

**Advantages**

* Avoids delays caused by compiling web pages when they are first accessed by a user.
* Finds compile errors before the site is deployed.
* Can copy just executable files and not the source files to the server.

The build -> publish web site command compiles all of the files makes up an asp.net application, and then deploys the compiled assemblies to the location we specify.

**Publish Web Site**

If we check allow precompiled site to be updatable box, the source files are deployed along with the executable files. If we uncheck then source code files aren’t deployed.

**The syntax of aspnet\_compiler command** The aspnet\_compiler command is located in the asp.net framework directory, **c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727**

**aspnet\_compiler –v virtual-directory [-u] [-d] [-f] [target-directory]**  
  
**where**

* -v name of virtual directory of existing web site
* -u precompiled website will be updated
* -d debug information is included in compiled assemblies
* -f overwrite target directory if exist

**Setup Deployment**

* It uses a web setup project to build a windows setup program used to deploy website onto server.
* Useful for deployment on multiple servers.
* Can be used to deploy precompiled assemblies and can be configured to include or omit the source code.
* The installed application can be removed by using add or remove programs dialog box from control panel

**How to create a setup**

Choose the file -> add -> new project command to display the add new project dialog box. Then, choose setup and deployment. Select web setup project as template, enter a name for websetup project click ok.

**Add project to Output Group**

In the solution explorer right click the web setup project and choose the add-> project output command to display the add project output group dialog box, then click ok to add the content files from you website to the web setup project

Use the buttons that are displayed at the top of the solution explorer when web setup project is selected to access setup editors that lets us to customize various aspects of the web setup project.

**Add setup prerequisites**

Visual studio creates files named setup.exe and setup.msi. setup.exe is the file we will run to install the application and setup.msi contains all of the files to be installed.

The setup.exe and setup.msi files are stored in the web setup project’s debug or release folder.

**Deploy SQL Server Database project’s with SqlPackage.exe**

To create and deploy a database that uses a SQL Server common language runtime (SQL Server CLR) assembly, you create a database project and a SQL Server CLR project. When you build and deploy the database project, you automatically deploy the SQL CLR assembly that it references at the same time.

In this walkthrough, you will perform the following tasks:

* **Create a simple database project.**
* **Create a SQL Server CLR project and define a new data type.**
* **Use the type that you defined in the SQL Server CLR assembly.**
* **Build and deploy the database project**.

## [Prerequisites](javascript:void(0))

To complete this walkthrough, you must have installed Visual Studio 2010 Premium. You must have access to an instance of SQL Server 2005 or SQL Server 2008. You must have installed the code samples for SQL Server from the [CodePlex Web site](http://go.microsoft.com/fwlink/?LinkId=122896). You must have permissions to deploy a database to your database server.

## [Create a Database Project](javascript:void(0))

### To create a database project

1. On the File menu, choose New, Project.
2. Under Installed Templates, expand the Database node, and choose the SQL Server node.
3. In the list of templates, choose SQL Server 2008 Database Project.
4. In Name, enter SQLClrDatabaseSandbox.
5. Select the Create directory for solution check box if it is not already selected.
6. Clear the Add to Source Control check box if it is not already cleared, and choose the OK button.
7. The empty database project appears in Solution Explorer.Next, you create a project for the SQL Server CLR assembly.

## [Create a SQL Server CLR Project](javascript:void(0))

### To create a SQL Server CLR project

1. On the File menu, choose Add, Project.
2. In Project types, expand the Database node, and choose SQL Server.
3. In the Templates list, choose either Visual Basic SQL CLR Database Project Server Project or Visual C# SQL CLR Database Project.
4. Choose .NET Framework 3.5 in the list at the top of the dialog box. (If you choose the default value, .NET Framework 4.0, and deploy to SQL Server 2008or SQL Server 2008 R2, deployment will fail.)

|  |
| --- |
|  |
|  |

1. In Name, enter SupportingSQLClrObjects.
2. Choose the OK button to accept the default location, close the dialog box, and create the project.
3. In the Add Database Reference dialog box, choose Cancel.By choosing Cancel, you enable the database project to determine where the assembly will be deployed.

Next, you create a custom data type in the SQL Server CLR assembly.

### To define a custom SQL Server CLR data type

1. On the Project menu, choose Add User-Defined Type.
2. In Name, enter MyCustomType.cs.The code editor opens and displays the contents of MyCustomType.cs. (At this point, you should modify the default user-defined type to provide an implementation that meets your requirements. For this walkthrough, you will accept the default implementation. For more information about how to define custom types and other SQL Server CLR objects, see this topic on the Microsoft web site: [Using CLR Integration in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkId=122897))
3. On the File menu, choose Save MyCustomType.cs.

### To configure and build the SQL CLR assembly

1. On the Build menu, choose Configuration Manager.
2. In Project contexts, clear the Deploy check box in the row for the SupportingSQLClrObjects assembly.
3. Choose the Close button.
4. In Solution Explorer, open the shortcut menu for the SupportingSQLClrObjects project, and choose Build.The assembly builds without any errors.

Next, you reference the new assembly and use the type that you defined in that assembly.

## [Use the Type Defined in the SQL Server CLR Assembly](javascript:void(0))

To use the type that you defined in the SQL Server CLR assembly, you must add a reference from your database project to that assembly. You then define the data type in your database project so that you can use that new data type in object definitions.

### To add a reference to the SQL Server CLR assembly

1. In Solution Explorer, expand the SQLClrDatabaseSandbox project, open the shortcut menu for the References node, and choose Add Reference. The Add Reference dialog box opens.
2. On the Projects tab, verify that the SupportingSQLClrObjects project is highlighted, and choose the OK button.
3. In Solution Explorer, expand the References node, choose the SupportingSQLClrObjects node, and choose F4.
4. Verify that Copy Local is set to True and that Permission Level is set to Safe.
5. In the Assembly Name property, replace SqlClassLibrary with SupportingSQLClrObjects.
6. In the Owner property, enter dbo.

You could specify a different owner.

1. On the File menu, choose Save All. The properties that you specified control the CREATE ASSEMBLY statement that is created when you build the project.

Next, you define a data type that uses the assembly that the SQL Server CLR project produces.

### To define the new data type

1. On the View menu, choose Error List, Database Schema View.
2. Expand the SQLClrDatabaseSandbox node, expand the Schemas node, expand the dbo node, and expand the Programmability node.
3. Open the shortcut menu for the Types node and choose Add, User-defined Type CLR.The Add New Item dialog box appears.
4. In Templates, verify that User-defined Type (CLR) is highlighted.
5. In Name, enter MyCustomType, and choose Add.The Transact-SQL editor appears and displays the definition for the type. In the Error List window, the following error appears:

SR0029 : Microsoft.Validation : User Defined Type: [dbo].[MyCustomType] has an unresolved reference to Assembly [assembly\_name]

1. In the Transact-SQL editor, update the code to match the following:
2. CREATE TYPE [dbo].[MyCustomType]
3. EXTERNAL NAME [SupportingSQLClrObjects].[MyCustomType]
4. On the File menu, choose Save MyCustomType.udtclr.sql.

In the Error List window, the error disappears because you have updated the type definition to refer to a valid assembly and class.

Next, you define a simple table that uses the new data type.

### To use the new data type in a table definition

1. In Schema View, open the Schemas node, and open the dbo node.
2. Open the shortcut menu for the Tables node and choose Add, Table.
3. In Templates, verify that Table is highlighted.
4. In Name, enter SimpleTable, and choose Add.
5. In the Transact-SQL editor, update the code to match the following:
6. CREATE TABLE [dbo].[SimpleTable]

(

column\_1 int NOT NULL,

column\_2 [dbo].[MyCustomType] NULL

)

1. On the File menu, choose Save dbo.SimpleTable.table.sql.

Next, you configure the database project and then deploy it to an isolated development environment.

## [Build and Deploy the Database Project](javascript:void(0))

### To configure build settings

1. In Solution Explorer, choose SQLClrDatabaseSandbox.
2. On the Project menu, choose SQLClrDatabaseSandbox Properties.
3. Choose the Build tab, and review the default settings. In this walkthrough, you can use the default settings.

Next, you configure deployment settings.

### To configure deployment settings

1. Choose the Deploy tab.
2. In the Deploy Action list, choose Create a deployment script (.sql) and deploy to database.
3. In the Configure deployment settings for list, choose My isolated development environment.
4. In Deployment script name, accept the default value.
5. In Target Database Settings, choose Edit to specify a target connection.
6. In the Connection Properties dialog box, specify a connection to the server and database where you want to deploy the database project, and then choose the OK button.
7. In Target database name, accept the default value.
8. In the Deployment configuration file list, choose Properties\Database.deploymentconfig. By default, your isolated development environment is not associated with a set of detailed deployment settings. When you specify this file, you will use the deployment settings that are associated with the database project. You could also create a separate configuration file and modify the settings for your isolated development environment.
9. In the SQL command variables file list, choose Properties\Database.sqlcmdvars.
10. On the File menu, choose Save All.

Next, you build the database project.

### To build and deploy the database project

1. On the Build menu, choose Build Solution. The database project and the SQL Server CLR assembly project build successfully.
2. On the Build menu, choose Deploy Solution. The database project and the SQL Server CLR assembly are deployed to the target server and database that you specified in the deployment settings. In the Output window, the following message appears:

========== Deploy: 1 succeeded, 0 failed, 1 skipped ==========.

The database project was successfully deployed. When you deployed the database project, you also deployed the SQL Server CLR assembly. The SQL Server CLR assembly did not have to be deployed separately, which is why it was skipped when you deployed the solution.

# Deploying a .NET Application with Crystal Reports

Crystal Reports often cause problems in .NET deployment package. Errors like “Can't find keycode32.dll” are likely to occur if the license key is not provided during creation of deployment package. To include license information, use the following procedure.

Create a Sample Deployment project.

Type the name of your project in the Name: Field “Setup” as in our example.

Now click the Application Folder from the left Pane (“File System on the Target Machine”) , then right click anywhere on the right yet empty pane and select Add > File from shortcut menu.

Go to the bin directory of your desired .NET application (which is already built) and select the \*.exe file. This will add necessary references of all the assemblies required.

Now go to Solution Explorer and right click on Setup and select Add > Merge Module…

From Add modules dialog box, select the first four files namely Crystal\_Database\_Access2003.msm, Crystal\_Database\_Access2003\_enu.msm, Crystal\_Managed2003.msm and Crystal\_regwiz2003.msm and click Open.

Now again from Solution Explorer, right click on crystal\_regwiz2003.msm and select properties. Set license key value under MergeModuleProperties > License key as “AAP50-GS00000-U7000RN”.

You can find license key information through the about box.

By following the above process you can easily deploy dot net application with crystal report