A

Setup Instructions

This appendix provides detailed setup instructions for labs and sample code referenced throughout this book. Each lab will specifically indicate which sections of this appendix must be completed prior to or during the lab. Code samples for each chapter in the book also rely on you completing the setup instructions for the labs in that chapter. A cross-reference will be provided for samples that require setup, referencing this appendix. The setup instructions discussed in this appendix include:

- Instructions for installing database scripts
- Configuration settings for SQL Server
- Instructions for configuring web applications for the ASP.NET membership and role provider
- Instructions for generating X.509 certificates
- Instructions for installing X.509 certificates
- Instructions for installing an SSL certificate with IIS

Database Setup

Each lab will have its own instructions for installing specific database scripts, or for configuring connection strings. In this section I'll review some general guidelines for database setup, script installation, and connection string formats.

Choosing a Database Engine

You have to make a decision about the database engine to use for samples that rely on a database or that use ASP.NET membership and roles. Your choice of database engine affects how you configure connection strings for labs and samples.

Your options are as follows:

- Microsoft SQL Server 2000 or 2005

- Microsoft SQL Server 2005 Express Edition (SQL Express)

If you don't have access to SQL Server 2000 or 2005, you can download SQL Express from the following location: http://msdn.microsoft.com/vstudio/express/sql/download/.

Installing Database Scripts

To install database scripts for book labs and samples, you'll need a management tool. If you installed SQL Server 2000 or 2005, you can use SQL Server Management Studio to administer any database. Otherwise, you can install SQL Server Management Studio Express from here: http://msdn.microsoft.com/vstudio/express/sql/download/.

To install a database script, open the database management tool, and then open and run the *.sql* script.

Database Connection Strings

Labs and samples in this book that rely on a database will require a connection string setting in the application or web configuration file.

The following illustrates a connection string for a SQL 2000 or 2005 installation on the local machine:

```
<connectionStrings>
  <add name="PhotosDatabaseConnection" connectionString="data
source=localhost;Initial Catalog=Photos;Integrated Security=True;"
providerName="System.Data.SqlClient" />
</connectionStrings>
```

By comparison, the following illustrates a connection for a SQL Express database:

```
<connectionStrings>
  <add name="PhotosDatabaseConnection" connectionString="data
source=.\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=Photos"/>
</connectionStrings>
```

Labs and samples are configured to use SQL 2000 or 2005, but you can refer to the differences here if you need to change those settings to work with SQL Express.

Database Security

In this book you will be working with web site projects that run with the ASP.NET account. That account is either ASPNET (for Windows XP) or NETWORK SERVICE (for Windows Server 2003 and Windows Vista). In some labs and samples you may be asked to give the ASP.NET account access to a particular database. This section will provide instructions for this based on Microsoft SQL Server 2005.

First, add the ASPNET or NETWORK SERVICE account to the database instance Logins as shown in Figure A-1. Right-click on the Login node and select New Login. Click Search and add the account as the login name. Click OK to complete the process.

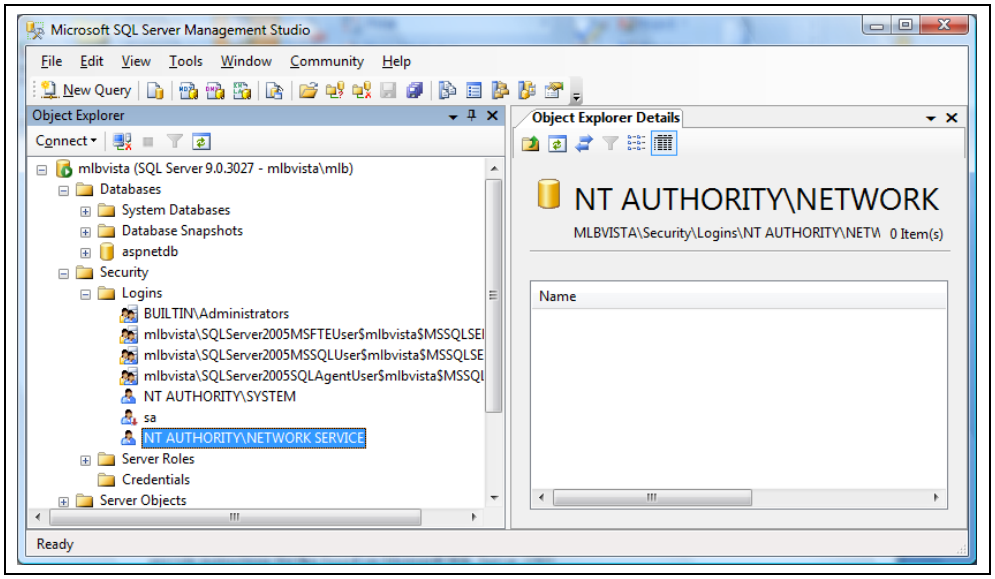

Figure A-1. Adding logins to SQL Server 2005.

Now, expand the Databases node and find the database you want to allow this account access to. Beneath the Security node for the database (Figure A-2) right-click on the Users node and select New User. Provide a name for the user, and beside Login name, click the browse button (…) and find the account. In the Database role membership section, check the db_datareader and db_datawriter role members (Figure A-3) and click OK to complete the process.

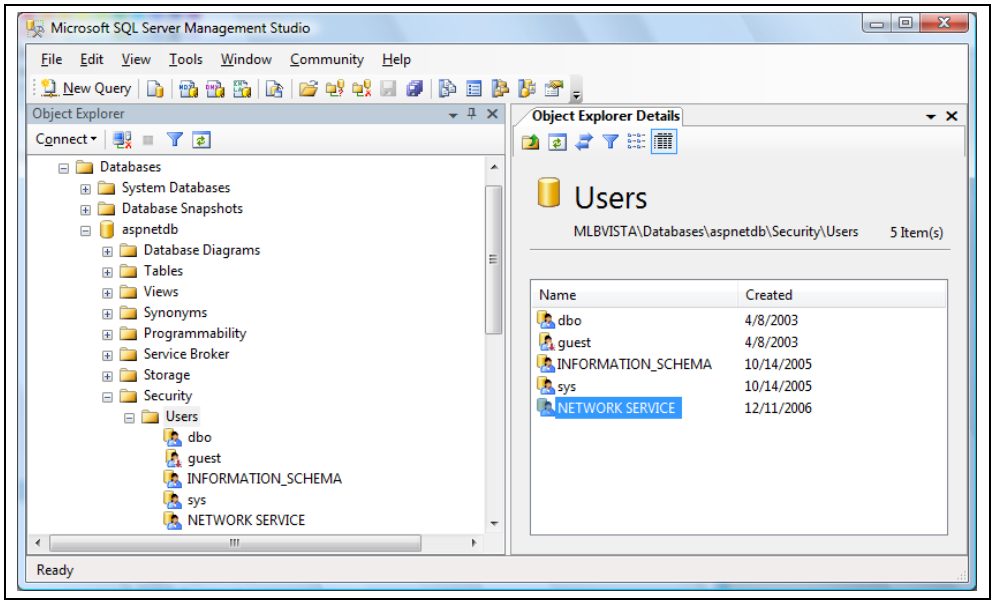

Figure A-2. Account access to a particular database

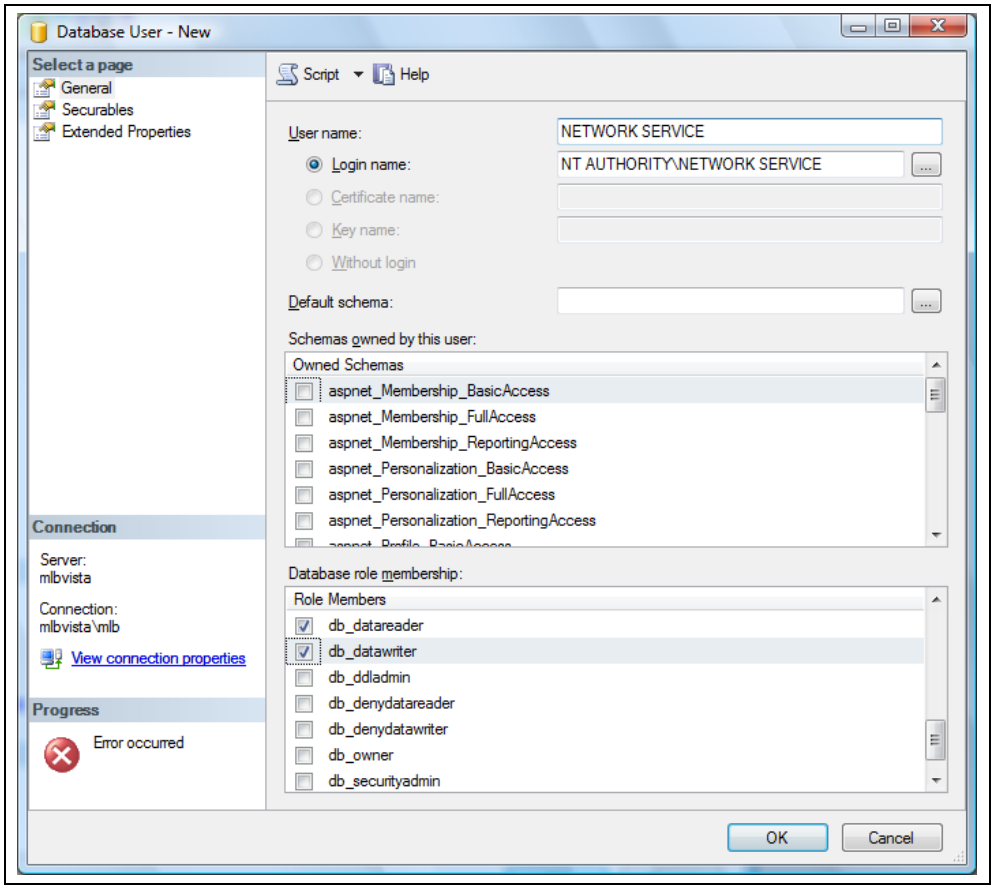

Figure A-3. Adding an account to a particular database

The new user should now appear in the list of users (Figure A-2).

ASP.NET Provider Model Setup

This section summarizes the database configuration settings required to use the ASP.NET membership and roles provider model with your WCF services. You'll learn how to override the default settings in the machine configuration file, how to select the right database connection for your environment, and how to initialize tables for the built-in ASP.NET providers including those for membership and roles.

Generating ASP.NET Provider Tables

For ASP.NET providers to work the database must be initialized with the ASP.NET provider tables.

If you are using SQL Express, a new database is created for you when you use the ASP.NET web site configuration tool. This database file has an *.mdf* extension, and is generated in the web site *\App_Data* directory.

If you are using SQL Server 2000 or 2005, or you would like to use the central SQL Express database instead of individual files, you must run the ASP.NET SQL registration utility — *aspnet_regsql.exe* by following these instructions:

1. From the Visual Studio 2005 command line, type *aspnet_regsql.exe*. The wizard will launch; skip to the second page.
2. From the "Select a Setup Option" page, select "Configure SQL Server for application services".
3. From the "Select the Server and Database" page, enter your server name, usually your machine name. If you are using SQL Express, type your machine name followed by "\SQLEXPRESS" (*[machinename]\SQLEXPRESS*). If you forget to type SQLEXPRESS, you will be attempting to connect to a SQL Server 2000/2005 default instance which may or may not be installed on your machine.

   Provide credentials for the database on this page as well. If you installed the database to support Windows authentication (recommended) you can click Next to continue. Otherwise provide your credentials to log in.
4. The summary page will state that it will create a database called `aspnetdb` for you. This will have numerous tables and supporting code for the ASP.NET provider model. Complete the wizard and you are ready!

To view the `aspnetdb` tables you just created, use SQL Server Management Studio or SQL Server Management Studio Express. If you're using the former, be sure to connect to the SQL Express instance by supplying the machine name followed by \SQLEXPRESS as shown in Figure A-4.

Figure A-4. Connecting to the default SQL Express instance

After connecting, you will be able to expand the Databases node to see the provider model tables generated for ASP.NET (see **Error! Reference source not found.**).

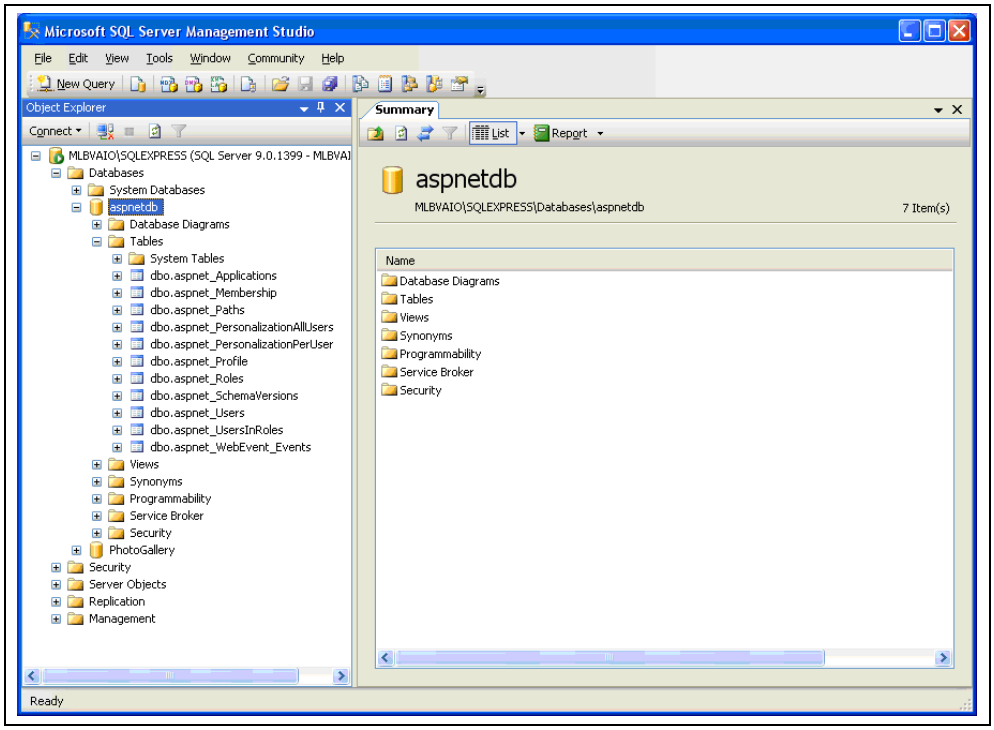

Figure A-5. ASP.NET provider model tables

ASP.NET Provider Connection Strings

Any lab that uses the ASP.NET provider model will expect you to put a connection string appropriate to the database you are using in your configuration file. The lab will tell you when to do this — this section explains which connection string you should use, and where to put it.

The default connection string in the *machine.config* file for ASP.NET provider is shown in Example A-1.

Example A-1. Default setting for ASP.NET providers in the machine.config

```
<configuration>

  … other settings

  <connectionStrings>
    <add name="LocalSqlServer" connectionString="data
source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true"
providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

If you have installed SQL Express, this connection string will expect a unique database instance (*.mdf* file) for each web site, beneath the *\App_Data* directory. This will create unnecessary work for you to complete each lab. Instead, if you plan to use SQL Express, I recommend that you override this setting in your application configuration file for the

lab by removing and re-adding the `LocalSqlServer` connection string as shown bold in Example A-2.

Example A-2. Using the default Express instance for ASP.NET providers

```
<configuration>
  <connectionStrings>
    <remove name="LocalSqlServer"/>
    <add name="LocalSqlServer" connectionString="data
source=(local)\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=aspnetdb"/>
  </connectionStrings>
</configuration>
```

If you are not using SQL Express, you'll still need to change this `LocalSqlServer` setting to a different connection string to support SQL Server 2000 or 2005. Add the bold configuration code from Example A-3.

> You can also configure ASP.NET providers to use a different connection string, by a different name, but this requires additional work. The easiest way to go is to remove and re-add the `LocalSq1Server` connection.

Example A-3. SQL Server 2000 or 2005 setting for ASP.NET providers

```
<configuration>
  <connectionStrings>
    <remove name="LocalSqlServer"/>
    <add name="LocalSqlServer" connectionString="data source=localhost;Initial
Catalog=aspnetdb;Integrated Security=True;" providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

The connection string should be set to the correct value before you use the web site configuration tool for a web project — for example when you are generating new users and roles. Each lab that relies on the ASP.NET provider model will tell you when to update the configuration file to reflect one of these settings.

Creating Sample Users and Roles

Several labs and samples rely on the ASP.NET provider model to provide authentication and authorization. This section will walk you through creating sample users and roles.

1. From Visual Studio, create a new web site.
2. From Solution Explorer, open the *web.config* file and provide a `<connectionStrings>` section for `LocalSqlServer` as discussed in the previous section. Save this change.

   > You must do this before you configure users and roles, unless you want to use the default SQL Express file-based database.

3. Open the ASP.NET Configuration utility. From the Web Site menu, select ASP.NET Configuration. This launches a browser with the configuration tool.
4. Click on the security link.
5. Configure the application for Internet security, and enable roles.

6. Now it's time to create some roles and users. Click the "Create or Manage Roles" link and from this page add three roles: `Administrators`, `Users`, and `Guests`.
7. Return to the main Security configuration page and select the "Manage Users" link. Click "Create new user" and from the Create User page enter values for the first user in Table A-1. Check the roles according to the Roles column in the table. Repeat this for each user in the table.

> The email address is not important to the labs, so feel free to use any email address.

Table A-1. Sample user accounts for labs

User Name	Password	E-mail	Question	Answer	Roles
Admin	p@ssw0rd	admin@thatindigogirl.com	NA	NA	Administrators, Users
User	p@ssword	user@thatindigogirl.com	NA	NA	Users
Guest	p@ssword	guest@thatindigogirl.com	NA	NA	Guests

8. Test the accounts by attempting to log in to the application. Add a Login control to the default page and run the web site in a browser instance (F5). Try to log in with one of the accounts. If you created the accounts correctly, you'll be authenticated and no exceptions will be presented. Test each account to be sure it works.

Certificate Setup

This section describes how to install sample certificates to support code samples and labs. Labs that require certificates will include a note requesting that you complete this section before you begin.

Sample Certificates

When you import certificates to the certificate store, you must choose a particular certificate store. In the labs and code samples for this book I am using the `LocalMachine` as the "server" store for certificates, and `CurrentUser` for "client" certificates. Table A-2 provides a list of certificates used by the code in this book, and the appropriate location for installation.

Table A-2. List of certificates and locations for installation

Certificate Filename	Subject Key	Description	LocalMachine	CurrentUser
RPKey.pfx	CN=RPKey	Private key pair for the relying party (target services).	Personal	NA
RPKey.cer		Public key for target services.	TrustedPeople, Trusted Root Certification	Personal

Certificate Filename	Subject Key	Description	LocalMachine	CurrentUser
			Authorities	
IPKey.pfx	CN=IPKey	Private key pair for identity provider services.	Personal	NA
IPKey.cer		Public key for identity provider services.	TrustedPeople, Trusted Root Certification Authorities	NA
SubjectKey.pfx	CN=Subject Key	Private key pair for client applications.	NA	Personal
SubjectKey.cer		Public key for client applications.	Personal, TrustedPeople, Trusted Root Certification Authorities	NA
LocalHost.pfx	CN=LocalHost	Private key pair to use import as the SSL certificate in IIS.	Personal, Trusted Root Certification Authorities	NA

All certificates are located in the *<YourLearningWCFPath>\SupportingFiles\Certificates* directory.

Certificates Console

If you are new to certificates you may also need some instructions on using the Certificates MMC snap-in. Here are some instructions for setting up the snap-in.

1. From the Start menu, select Run and type *mmc.exe*. Click OK to launch the MMC console.
2. From the File menu, select Add/Remove Snap-in. Click Add from the dialog and select Certificates from the Add Standalone Snap-in dialog.

   Click Add, select `My` user account, and click Finish.

   Click Add, again and select Computer account. Click Next and then Finish.

   Close the Add Standalone Snap-in dialog. Click OK on the Add/Remove Snap-in dialog.
3. Save the Certificates snap-in settings to a file. From the File menu, select Save. Name the file *certificates.msc*. You should see the console shown in Figure A-5.

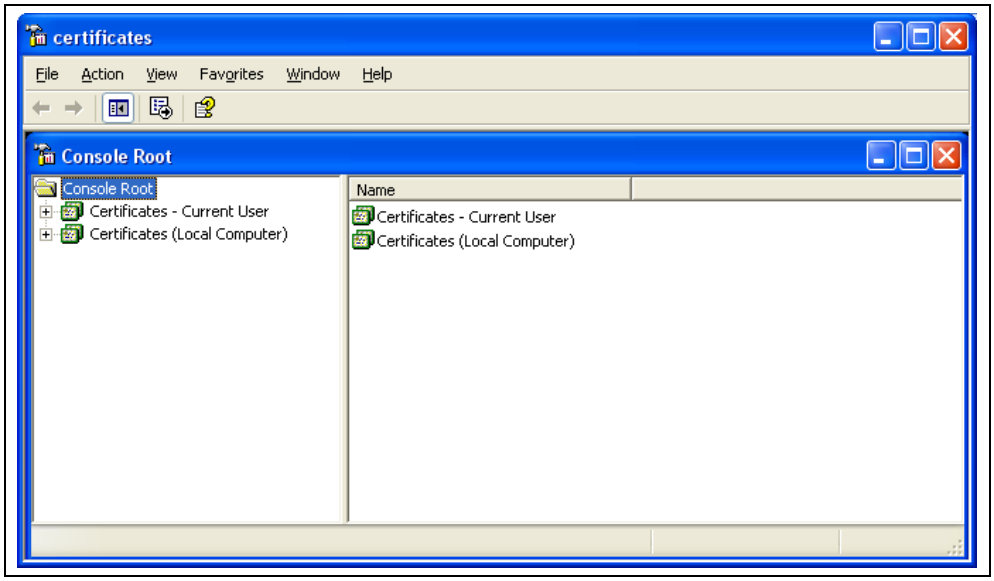

Figure A-5. Certificates MMC console

4. To re-open this console, repeat Step 1 and open the *certificates.msc* file.

You will use this console to import and export certificates.

Importing Certificates

You can use the Certificates console to install certificates listed in Table A-2 to the appropriate certificate stores under `LocalMachine` or `CurrentUser`. Each certificate in Table A-2 indicates exactly which stores it must be installed in and you must pay careful attention to complete the installation for each certificate to exactly that list of certificate stores in order for all labs and samples in this book to work properly.

To import certificates do the following:

1. Expand the certificate store and the folder specified in Table A-2.
2. Right-click on the Certificates folder in the hierarchy and select All Tasks→Import (see Figure A-6).
3. Follow the wizard steps. Browse for the certificate file in the *<YourLearningWCFPath>\ SupportingFiles\Certificates* directory. When browsing for *.pfx* files, change the "Files of type" selection to **.pfx* instead of **.cer*.

   Provide the password "indigo" for private key pairs and do not mark the keys as exportable.

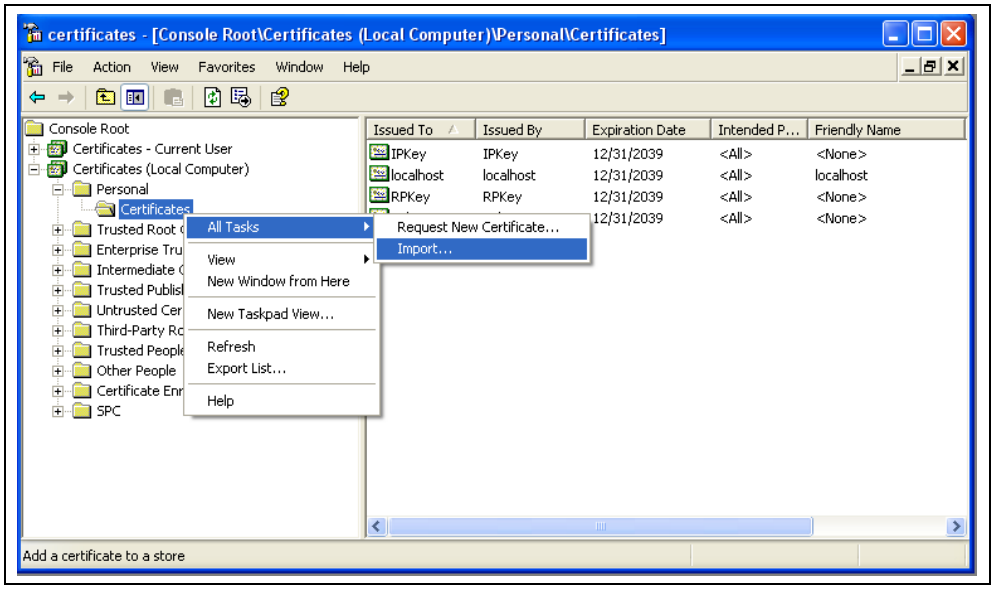

Figure A-6. Importing certificates

Certificate Security

The executing process identity must be given access to private keys installed into a certificate store For example, when hosting services in IIS, the ASP.NET identity (ASPNET or NETWORK SERVICE) must be granted access to the certificate file.

To grant a Windows account access to a particular certificate, follow these instructions:

1. The Windows SDK for .NET 3.0 includes samples for WCF in the zip file *WCFSamples.zip*. If you have unzipped this file, beneath the *\WCFSamples* directory you'll find sample code for the `FindPrivateKey` tool beneath *\TechnologySamples\Tools\FindPrivateKey*.

   As an alternative you can download the individual sample from here: *http://msdn2.microsoft.com/en-us/library/aa717039.aspx*.

   Compile this sample.

2. *FindPrivateKey.exe* is a command-line tool that can be used to find the directory and file that contains a particular key. For example, to find the full path to a key file from the `LocalMachine My` certificate store, type the following command:

   ```
   Findprivatekey.exe My LocalMachine -n "CN=RPKey" -a
   ```

   Table A-2 includes the subject key name for each certificate.

3. Once you have the path to the file, you can navigate to it using File Explorer as shown in Figure A-7. Right-click on the key and select Properties. Select the Security tab from the Properties dialog as shown in Figure A-8.

   From this tab you can add the appropriate users or groups to the list. On Microsoft Vista machines, click Edit, and then Add to add new users or groups. On Windows XP/SP2 machines, click Add.

   After adding ASPNET or NETWORK SERVICE to the list, save the changes.

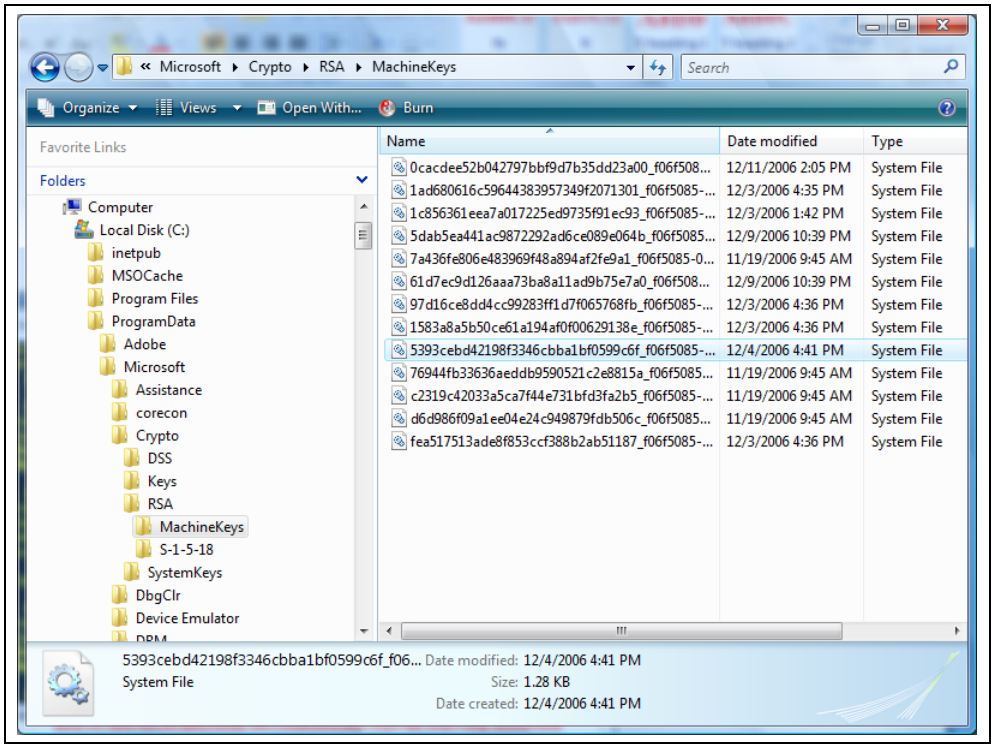

Figure A-7. Using File Explorer to browse certificates

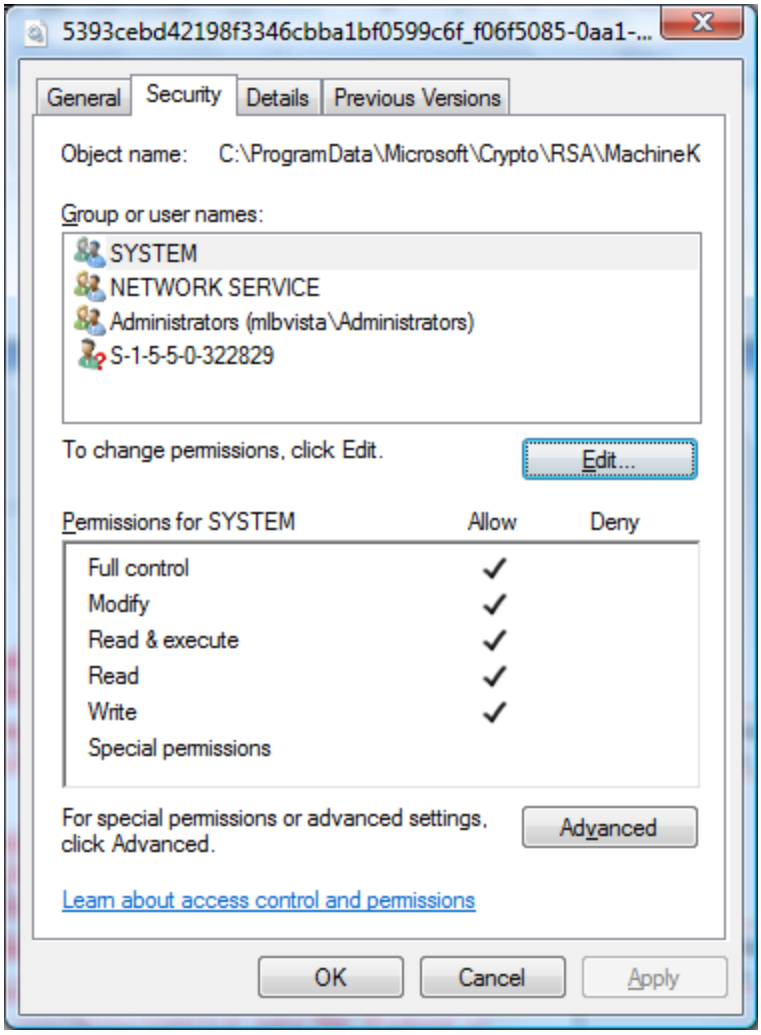

Figure A-8. Editing security for a directory or file

Setting up an SSL Certificate

Some labs and samples rely on an SSL-enabled web site. This requires you to attach a certificate to your default web site in IIS. This section discusses how to configure the `localhost` certificate from Table A-2 as the SSL certificate for the default web site. First I'll walk you through instructions for IIS 5.1 and 6.0, followed by instructions for IIS 7.0.

> Before you continue, follow the instructions in "Importing Certificates" and install the `localhost` certificate.

Instructions for IIS 5.1 and 6.0

The following are explicit instructions for installing an SSL certificate in IIS 5.1. IIS 6.0 follows similar instructions.

1. Import the *localhost.pfx* certificate as instructed in earlier sections.
2. Open the console for IIS from Control Panel→Administrative Tools.
3. Right-click on the default web site node and select Properties.
4. From the Directory Security tab, select Server Certificate (see Figure A-9).

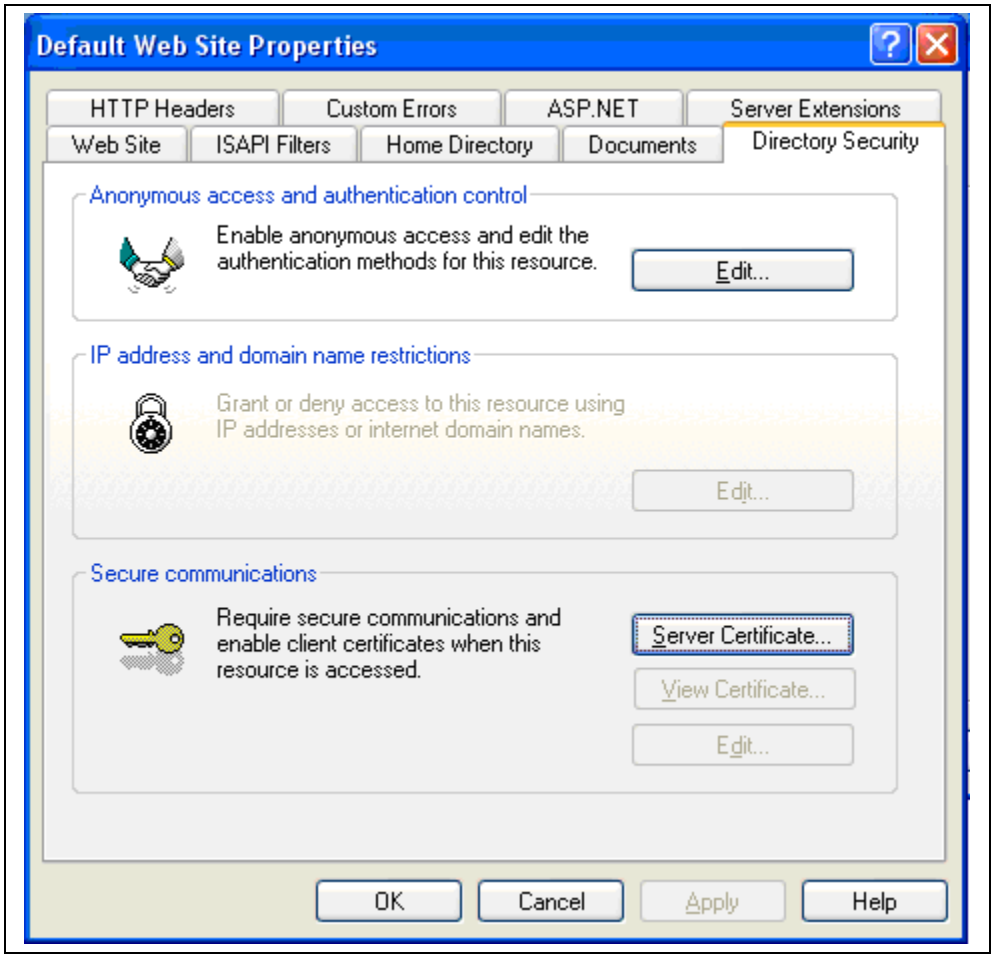

Figure A-9. Directory Security settings for the default web site

4. Select "Assign an existing certificate" (Figure A-10) and click Next.

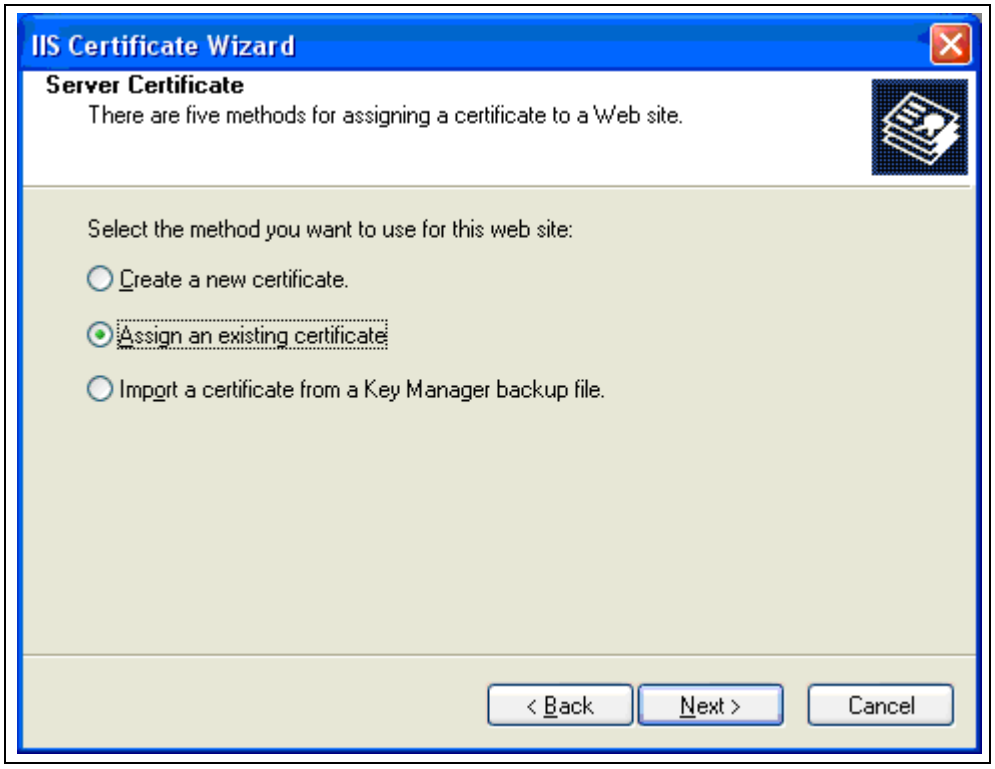

Figure A-10. Assigning an existing certificate to enable SSL support

5. Certificates installed to the LocalMachine→Personal store will be presented. Select `localhost` and continue (Figure A-11).

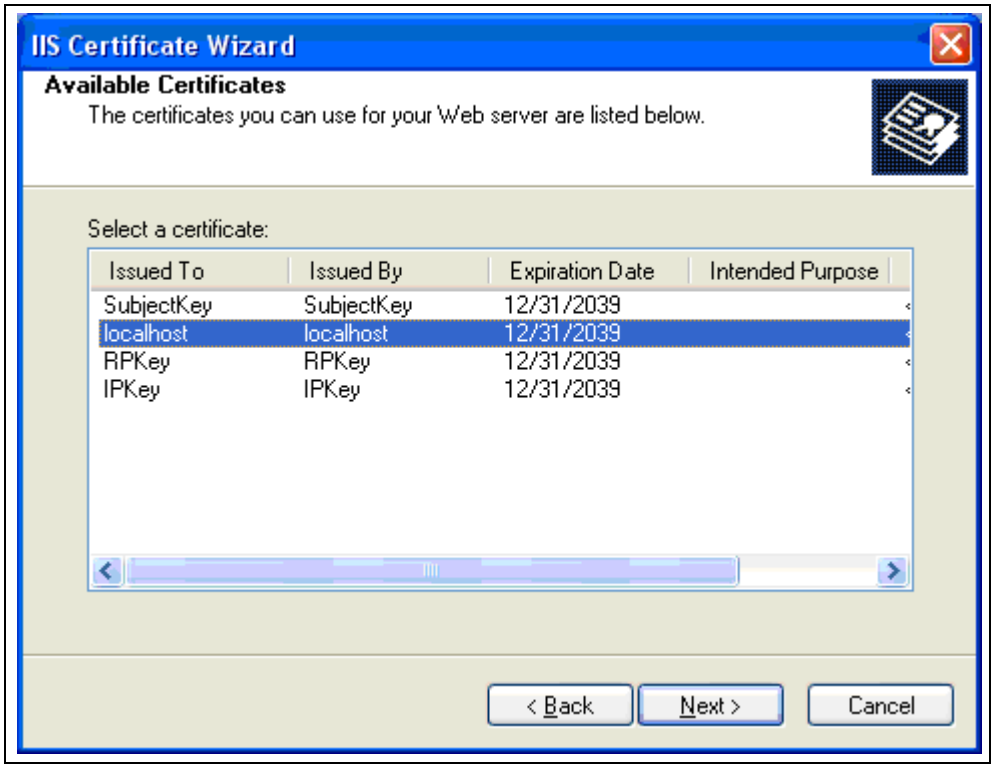

Figure A-11. SSL certificates are installed to the LocalMachine→Personal store

Instructions for IIS 7.0

The following are explicit instructions for installing an SSL certificate in IIS 7.0.

1. Open the console for IIS from Control Panel→Administrative Tools.
2. Select the root node and find the Server Certificates feature shown in Figure A-12. Select this feature, and verify that the `localhost` certificate is listed, as shown in Figure A-13. If not, import *localhost.pfx* from here — this will import it into the `LocalMachine` certificate store.
3. Enable HTTPS protocol for the default web site. This is done using the following *appcmd.exe* command line instruction:

   ```
   appcmd.exe set site "Default Web Site"
   -+bindings.[protocol='https',bindingInformation='*:443']
   ```

   I have provided a batch file with this instruction: *<YourLearningWCFPath>\Samples\Security\enablewashttps.bat*.
4. Now you can associate the `localhost` certificate with the default web site for HTTPS protocol. Return to the IIS console, select Default Web Site and find the SSL Settings feature as shown in Figure A-14. From the Actions pane, select Bindings from the Edit Site section.

   In the Web Site Bindings dialog you should see HTTPS protocol listed (see Figure A-15). Select the HTTPS protocol item from the list and click Edit.

From the Edit Web Site Binding dialog, drop down the list for SSL Certificate – you should see localhost listed if you have installed the certificate per instructions so far. Select localhost and save these changes.

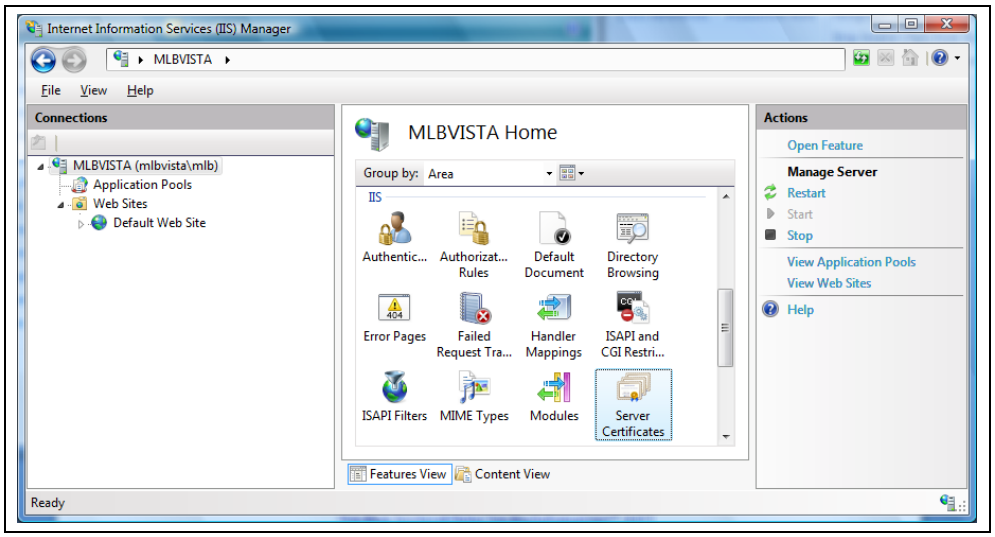

Figure A-12. Server Certificate feature of IIS 7.0

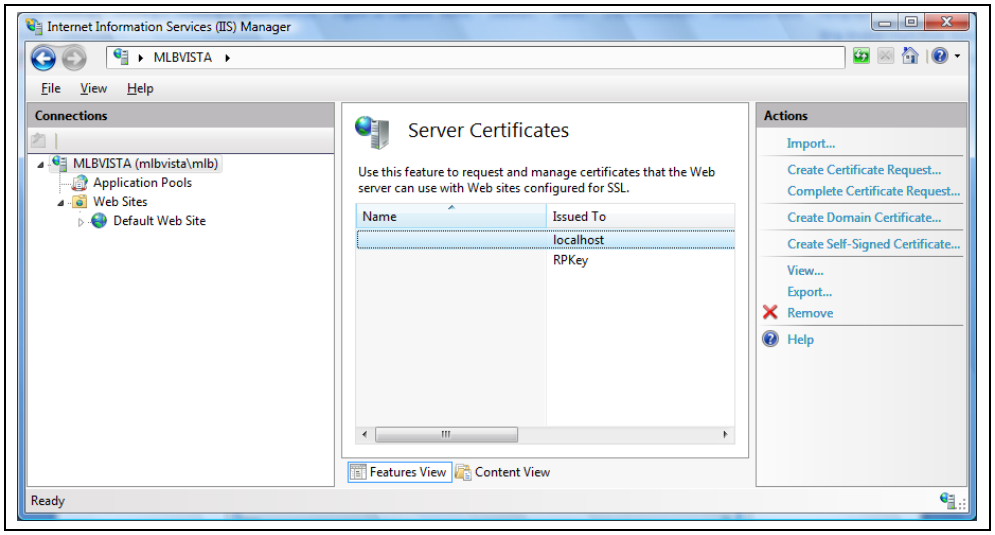

Figure A-13. Importing SSL certificates in IIS 7.0

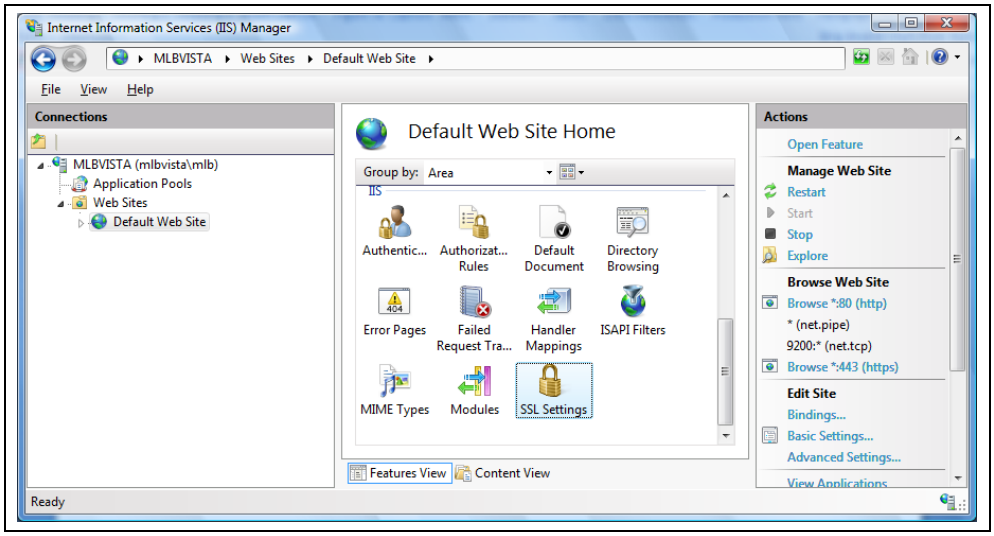

Figure A-14. SSL settings feature for the default web site

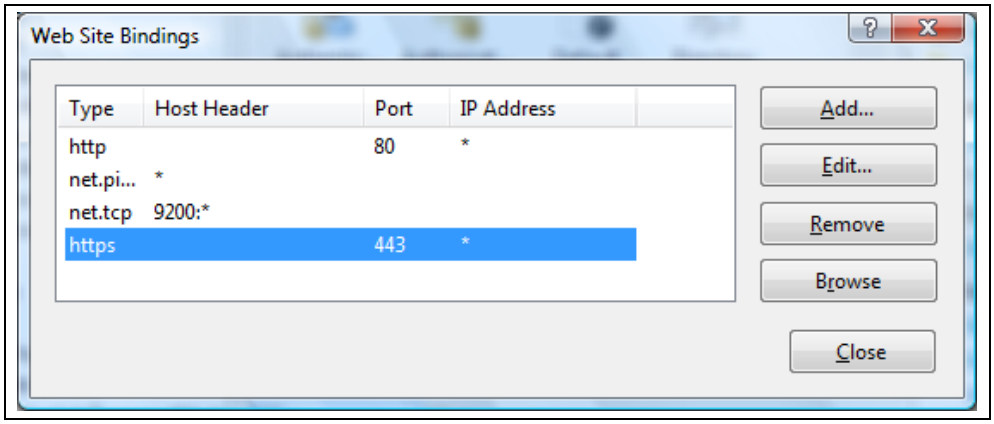

Figure A-15. Editing bindings for the default web site

Generating Certificates

The labs and samples expect you to be using the certificates provided. Still, you may want to create your own certificates for future work, or you may need to create a test certificate for SSL that matches your machine or the domain name of your default web site.

These instructions explain how you can use *makecert.exe* to create test certificates.

1. Launch the Visual Studio command line and type these instructions to generate a certificate named `localhost`:

```
makecert -r -pe -n CN=localhost -ss my -sr currentuser -sky exchange -sp "Microsoft
RSA SChannel Cryptographic Provider" -sy 12 c:\localhost.cer
```

> Naming the certificate `localhost` allows you to use it for local testing on your machine. The SSL certificate must be named the same

as the web site. You can rename `localhost` to your machine or domain name if necessary.

The certificate will be generated in the `CurrentUser` store, but you will install it in the `LocalMachine` store for SSL.

2. Since the certificate is generated to enable exporting the private key, you can export it using the Certificates console. Expand the CurrentUser→Personal store, and selecting All Tasks→Export (see Figure A-16).

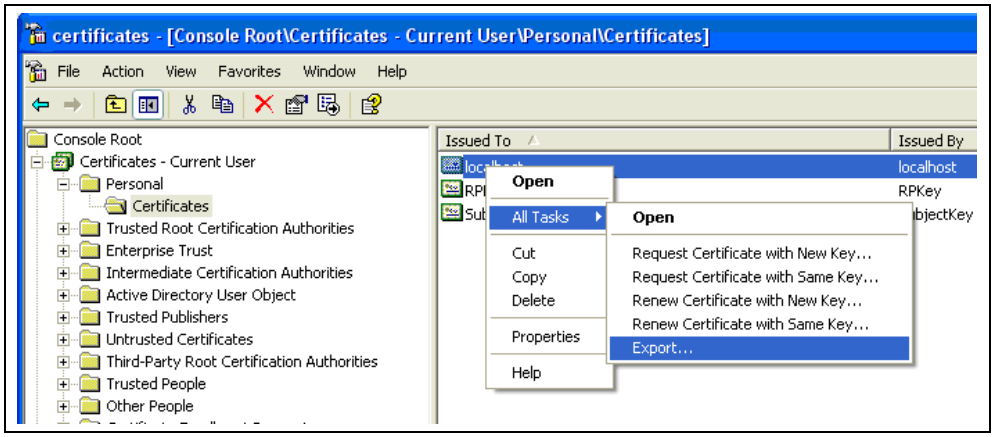

Figure A-16. Exporting a private key certificate

3. From the wizard, select "Yes, export the private key" (Figure A-17). Click Next.

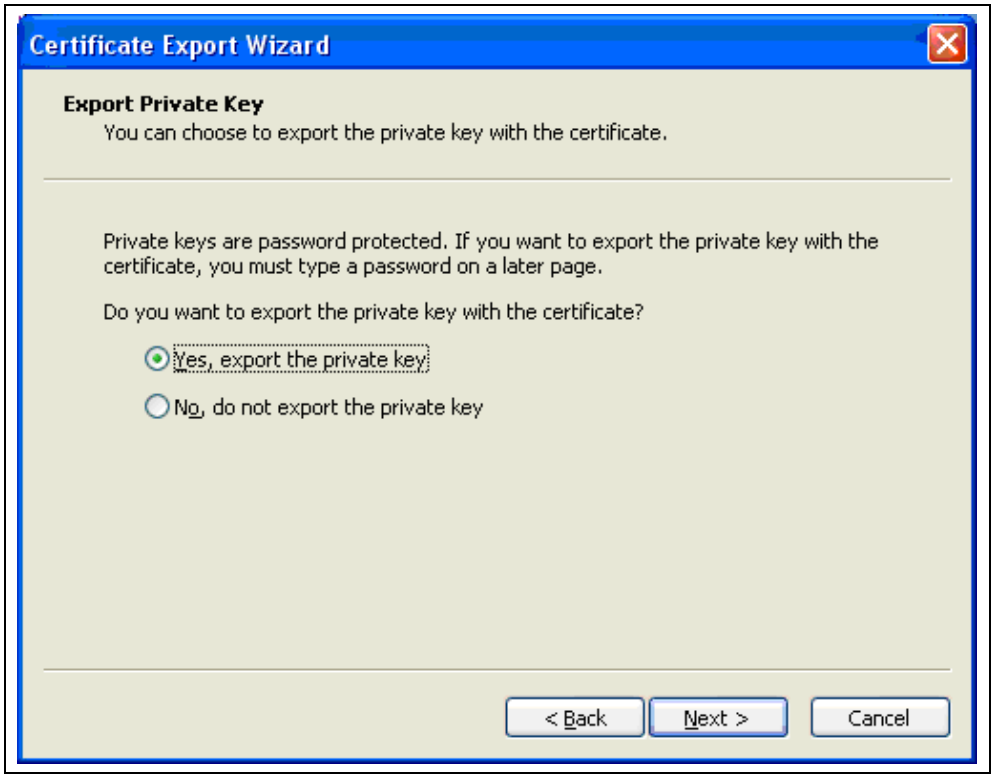

Figure A-17. Export wizard options for exporting private keys

4. Select "Delete the private key if the export is successful" (see Figure A-18). Click Next.

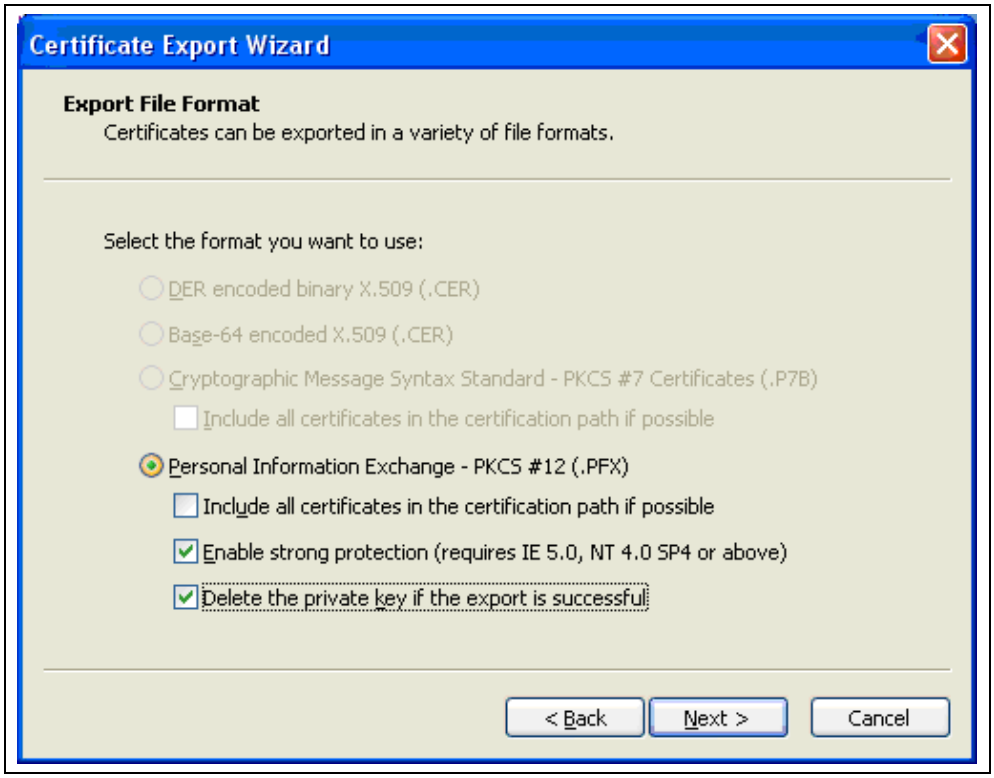

Figure A-18. It is best to remove the key when exporting, in particular since the new key did not have password protection and supports exporting private keys

5. Provide a password to protect the key (Figure A-19). Click Next.

Figure A-19. Private keys require a password

6. Select a filename for the key pair with a *.pfx* extension (Figure A-20).

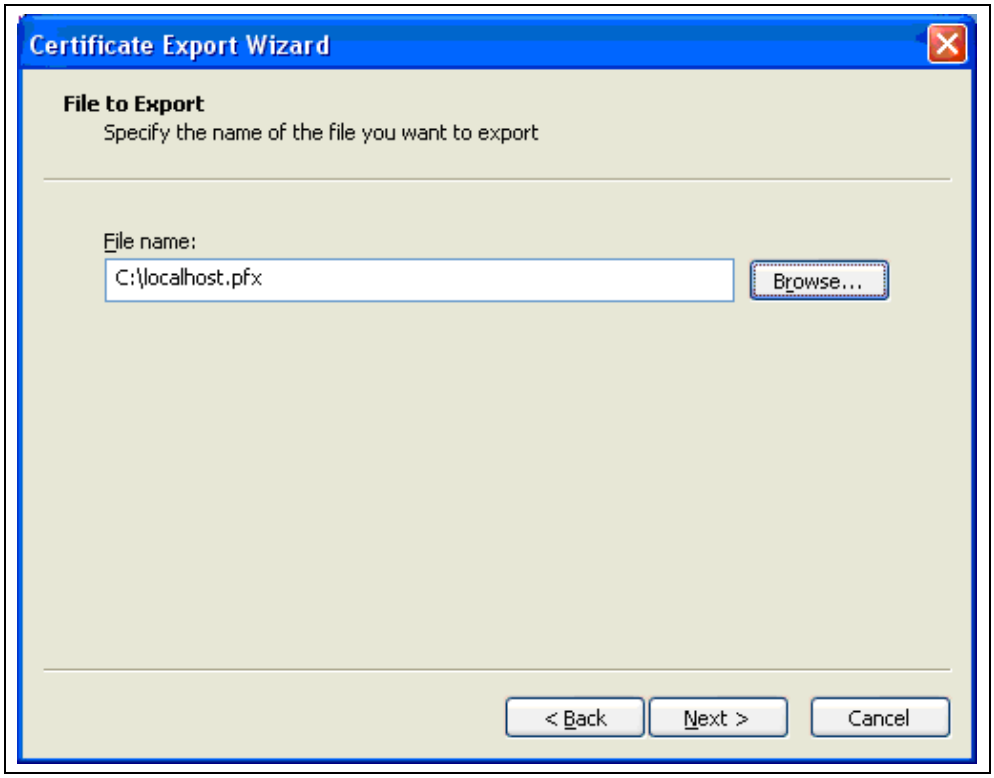

Figure A-20. Selecting the private key certificate file.

IIS Application Directories

Labs and samples that require you to set up an IIS application directory (also called a virtual directory) will provide you with a *.vbs* script to run, usually in the root of the solution. In some cases, however, you will create a new web site in IIS using Visual Studio. Depending on your default settings in IIS, it is possible that Windows Integrated Authentication may not be enabled for the application directory. This section explains how to accomplish this for IIS 5.1/6.0 and IIS 7.0.

Instructions for IIS 5.1 and 6.0

The following are explicit instructions for enabling Integrated Windows Authentication for virtual application directories in IIS 5.1. IIS 6.0 follows similar instructions.

1. Open the console for IIS from Control Panel→Administrative Tools.
2. Select the web site or application directory to configure, and edit its properties.
3. From the properties dialog, select the Security tab.
4. Check the Integrated Windows Authentication checkbox and save the changes.

Instructions for IIS 7.0

Follow these instructions to enable Integrated Windows Authentication for virtual application directories in IIS 7.0.

1. Open the console for IIS from Control Panel→Administrative Tools.
2. Select the web site or application directory to configure, and find the Authentication feature.
3. From the Authentication feature, enable Windows Authentication.