

uniGUI Developer's Manual

© 2014 FMSoft Co. Ltd.

Table of Contents

Foreword	0
Part I Installation	3
1 Requirements	3
2 Installation Instructions	4
Part II Developer's Guide	9
1 Web Deployment	9
Sencha License Considerations	9
Adjusting Paths	10
VCL Application	11
Standalone Server	12
ISAPI Module	14
IIS 5	15
IIS 6	19
IIS 7	27
Apache 2.2.....	34
Windows Service	34
SSL Configuration	35
Obtain a SSL Certificate from an Authority.....	36
Generate a Self-Signed Certificate.....	37
Configure SSL Parameters.....	40
Index	43

1 Installation

1.1 Requirements

uniGUI does not require any special system configuration. A typical uniGUI installation will occupy 100-150MB of HDD space.

Note: Requirement for runtime environment can be very different and will be discussed under deployment topic.

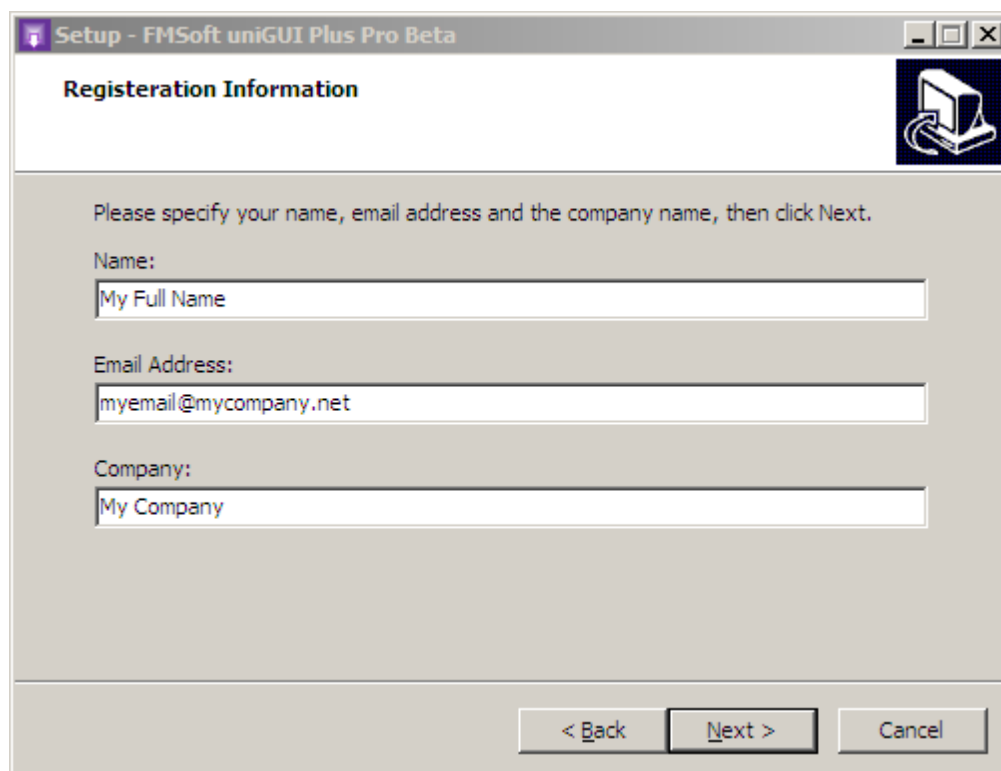
1.2 Installation Instructions

Installation instructions for uniGUI (Delphi and C++ Builder**)

- Before installing a new version remove all design packages from Delphi and uninstall uniGUI from Windows Program Add/Remove.
- After re-compiling an application with this new version, "ext" folder must be re-deployed to PCs running new version of your application or you can simply re-install the newly introduced Ext JS runtime package which can be downloaded from Downloads page.

****Note for C++ Builder:** You need **RAD Studio IDE** to install **uniGUI** for **C++ Builder**.

- 1) Download the latest uniGUI Setup from customer portal.
- 2) Enter required information. Make sure entered email address is same as email address registered in customer portal.



Setup - FMSoft uniGUI Plus Pro Beta

Registration Information

Please specify your name, email address and the company name, then click Next.

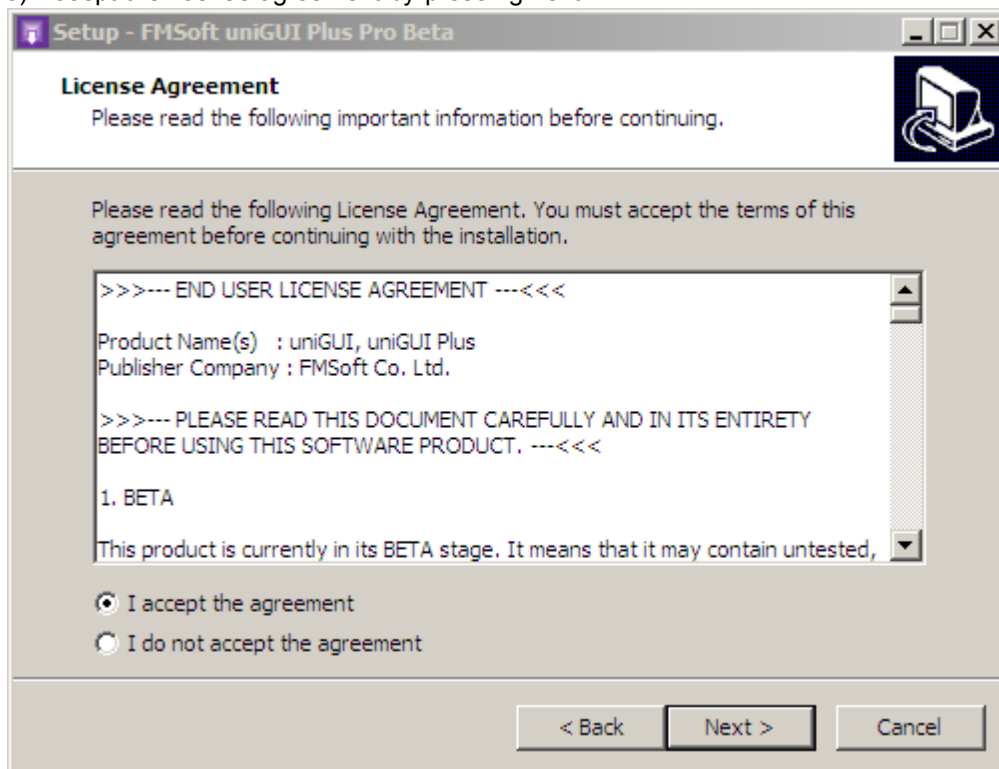
Name:

Email Address:

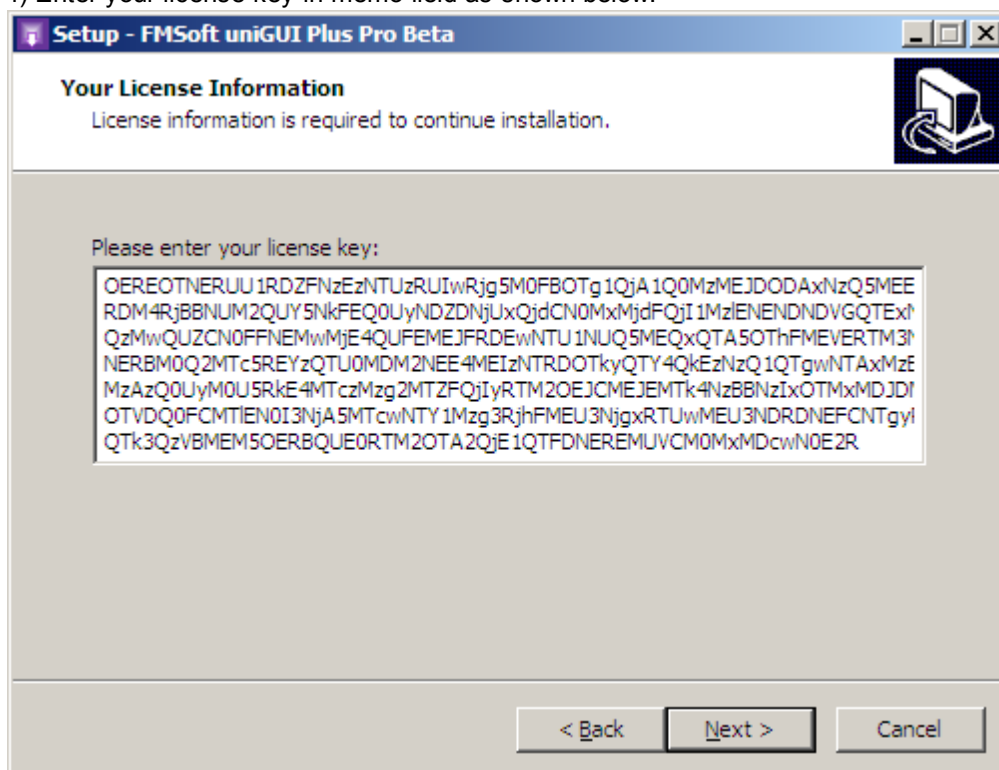
Company:

< Back Next > Cancel

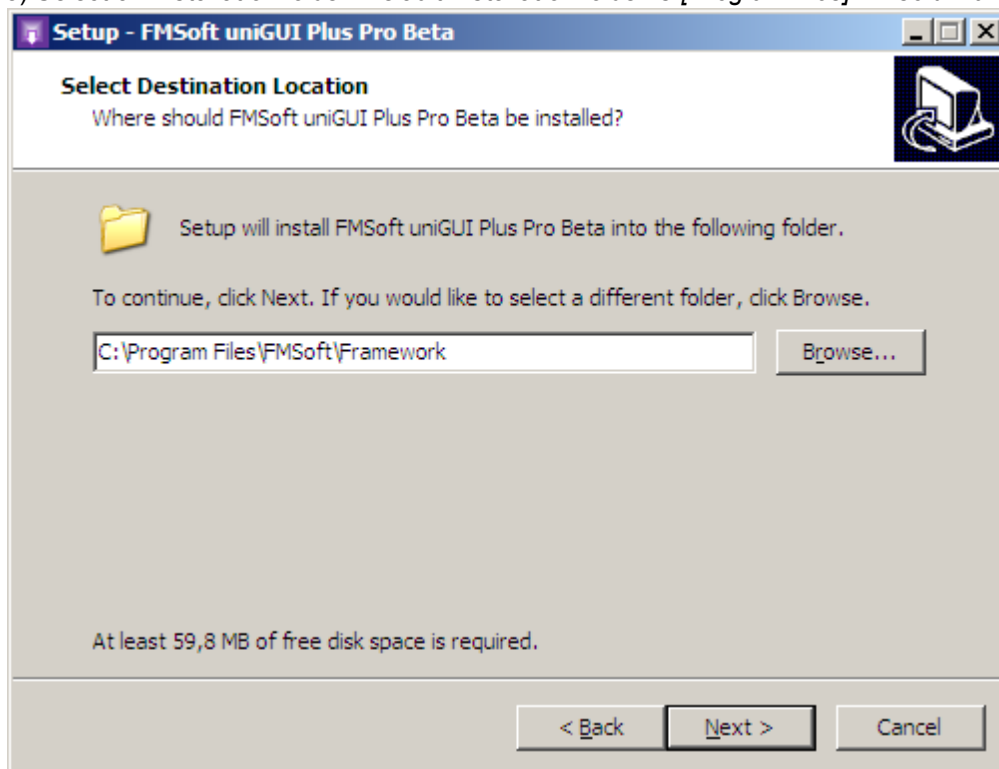
3) Accept the license agreement by pressing next.



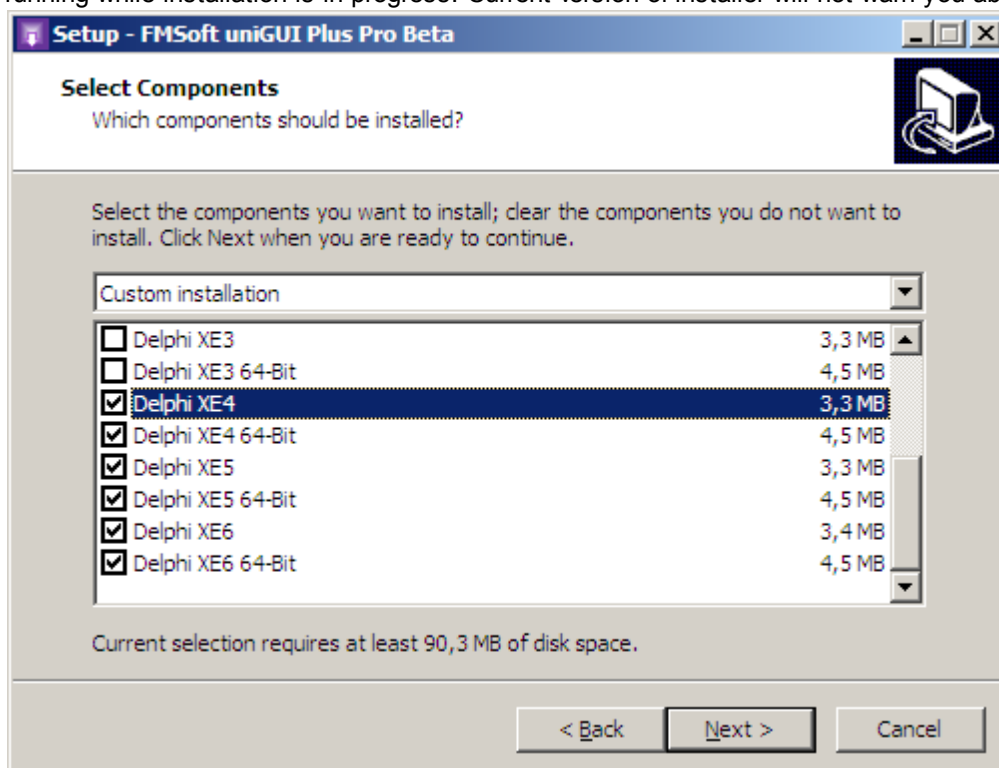
4) Enter your license key in memo field as shown below.



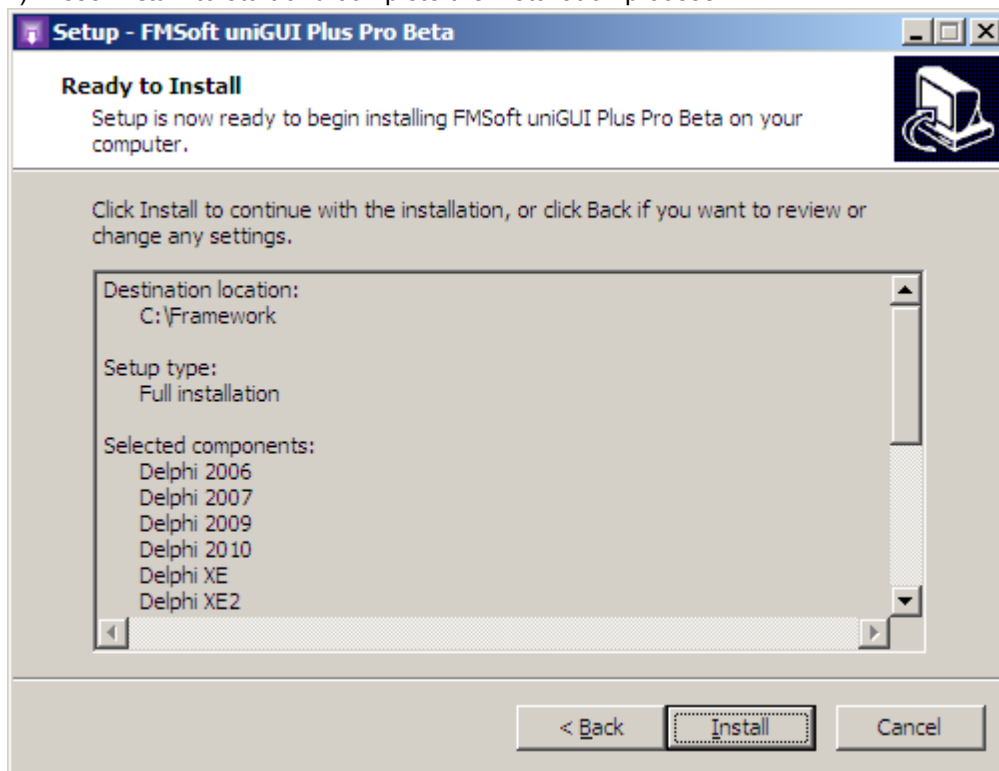
5) Select an installation folder. Default installation folder is *[ProgramFiles]\Fmsoft\Framework*.



6) Select Delphi version(s) you want to install uniGUI library. You must be sure that Delphi is not running while installation is in progress. Current version of installer will not warn you about this.

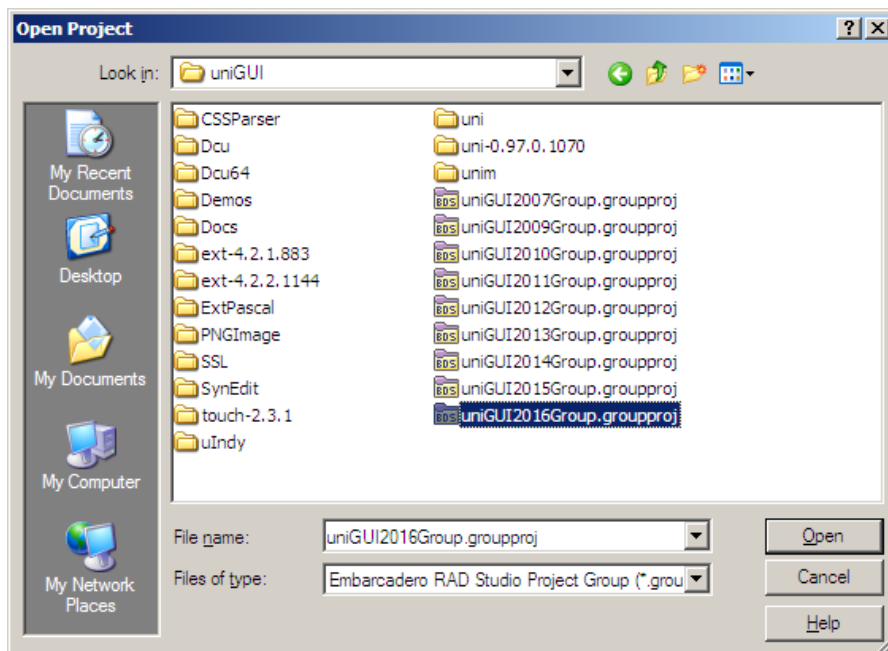


7) Press **Install** to start and complete the installation process.



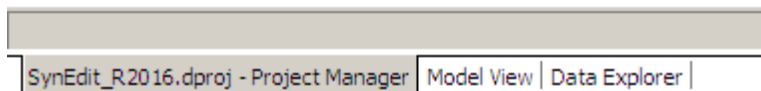
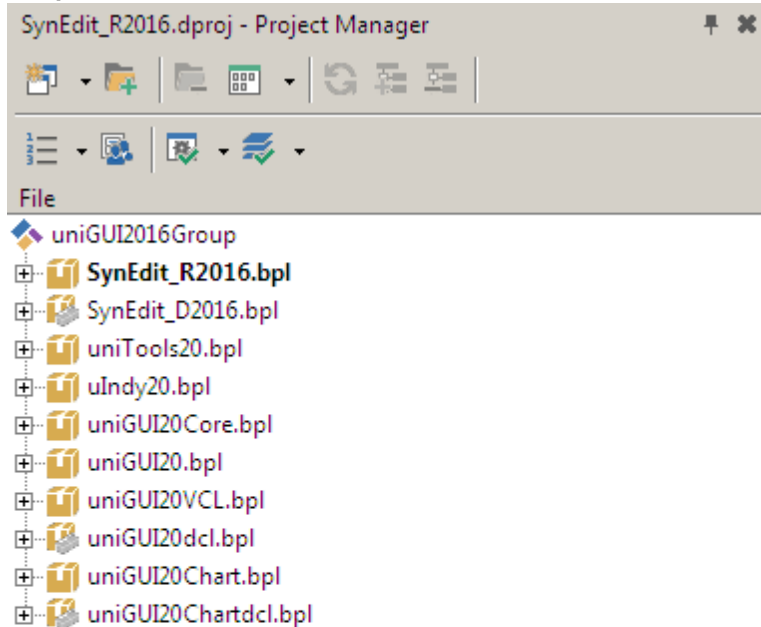
8) Start Delphi and open the project group for your Delphi version. e.g. **uniGUI2016Group (Delphi XE6)**.

****Additional step for C++ Builder:** Instead of Delphi IDE open project in **RAD Studio IDE**.



9) In project manager there are 11 Delphi packages. Build all packages starting from **SynEdit_Rxxxx.bpl**.

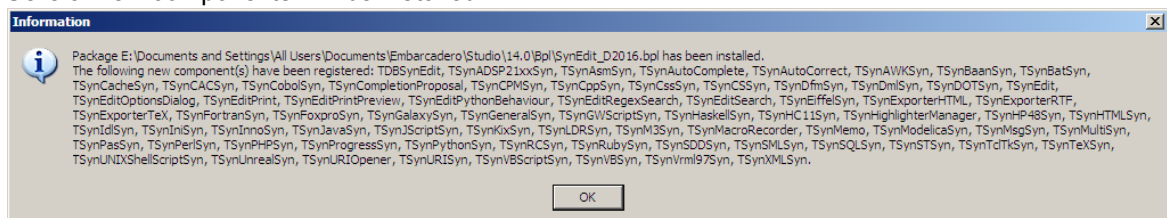
****Additional step for C++ Builder:** Before building packages, for each individual package please go to **Options->Linker** and Select/Set **Generate all C++Builder Files**.

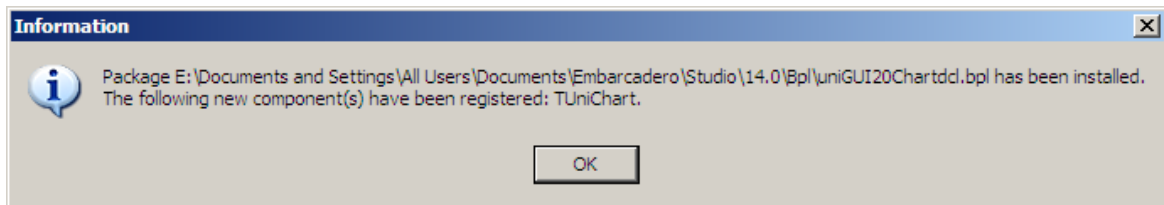
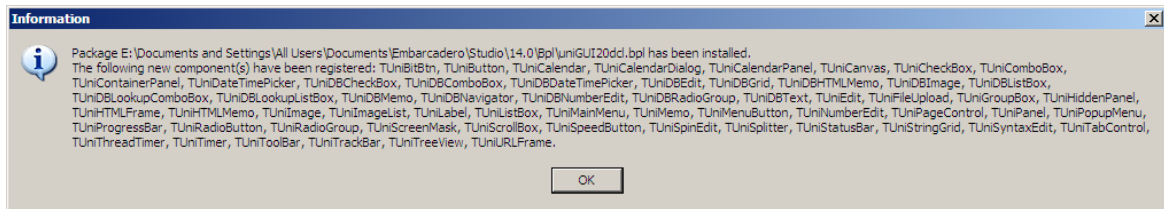


10) After building all packages install design time packages by right-clicking and selecting **Install** in following order:

- **SynEdit_D20xx.bpl**
- **uniGUIxxdcl.bpl**
- **uniGUIxxChartdcl.bpl**

Several new components will be installed:





****Additional notes for C++ Builder:**

- After starting a new C++ project you must disable **Linker->Dynamic RTL**.
- Currently building with Run Time Packages doesn't work properly. You must statically link all libs and create a single EXE. (Simply unselect **Build with runtime packages** option.)
- New C++ projects are created without a resource (.RES) file. As a result project has no default icon. This issue will be fixed in next releases.
- Combo VCL/ISAPI projects are not supported for C++ Builder.

2 Developer's Guide

2.1 Web Deployment

Currently there are four options available for deploying your uniGUI project to the Web .

[VCL Application](#)

[Standalone Server](#)

[ISAPI Module](#)

[Windows Service](#) (not implemented in this version)

2.1.1 Sencha License Considerations

FMSoft Co. Ltd. is an official partner of [Sencha Inc.](#) and is granted to distribute **OEM** copies of **Sencha Ext JS**.

UniGUI distributes a partial copy of **OEM Sencha Ext JS**. Your **uniGUI** license grants you using and deploying **Sencha Ext JS**. However, your **OEM** copy of **Sencha Ext JS** can only be used combined with **uniGUI**.

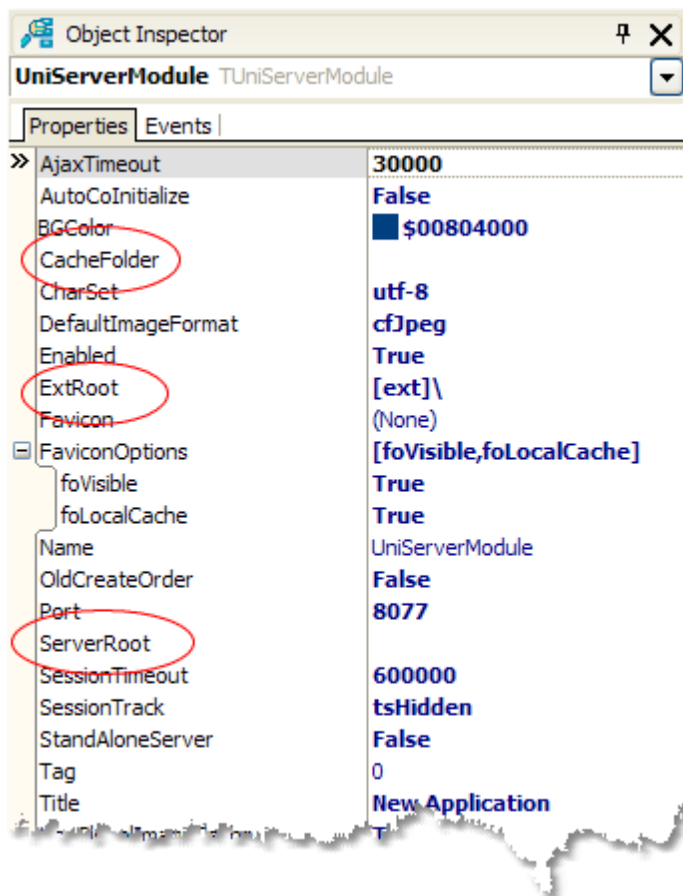
You may not use or distribute included **OEM Sencha Ext JS** for any other purpose other than using it to develop and deploy **uniGUI** projects. This library can not be treated as a standalone library and used to create independent software products based on it.

Please keep in mind that you get **Sencha Ext JS** as an integrated part of **uniGUI** package. You do not get **Sencha Ext JS** in form of a separate product and you do not own a separate license for it.

2.1.2 Adjusting Paths

There are some essential paths for a **uniGUI** application which must be adjusted before you deploy it to the outside world.

First of all, you must be sure that your Web Application knows where **Ext JS** files are located. For this, in your Application ServerModule you must assign a proper path to **ExtRoot** property. Default value of **ExtRoot** is "[ext]\\" which means Ext JS files are located under *<InstallFolder>\FMSoft\Framework\uniGUI\ext*



Since you will not install **uniGUI** library on target PC, you must assign a full or a relative path to **ExtRoot**. If you assign a relative path it will be relative to **ServerRoot** and you can use the "...\\..\\..\\path" notation.

The easiest method is to set the **ExtRoot** to blank and copy the "ext" folder to same folder your Web application executable/module resides. (A blank value is automatically translated to 'ext\') However, for security reasons it is better to put "ext" folder in another folder and deploy all "ext" files as read-only. Under **IIS** you must be sure that your application has enough credentials for a read-only access to "ext" folder and its files.

In **ServerModule** there are two other path related parameters, **ServerRoot** and **CacheFolder**.

ServerRoot:

ServerRoot defines root path for all relative paths. A blank value points to application startup folder.

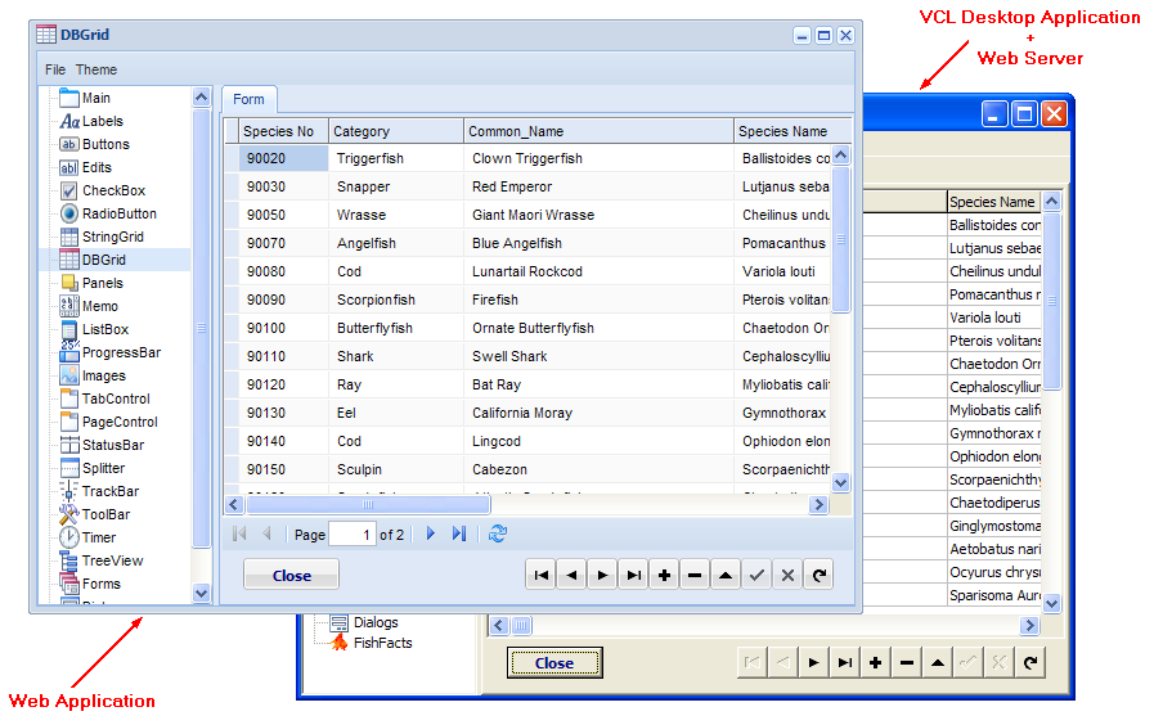
CacheFolder

A **uniGUI** server needs a folder to create temporary files. Normally, it is a folder named **Cahce** created under same folder your module exists. You can change this by assigning a different path to **CacheFolder** parameter. Under **IIS** you must be sure that your application has enough credentials for a full access to **CacheFolder**.

2.1.3 VCL Application

One of the unique features of uniGUI is its ability to create VCL applications which are a web server at the same time. This means that each VCL Application created with uniGUI library contains a Web Server. This web server allows multiple sessions of same application served through the Web using a regular browser. By this mean, you can simultaneously test your application inside your desktop and in a browser window. While your desktop application is running, you can run a visually intact copy of it inside a browser window.

It is the default mode selected in project wizard when your start a new uniGUI project and probably you will not use this mode for serious web deployment unless it is a small application which needs a temporary web gateway. Needless to say that closing the VCL application also terminates the web Server, so web gateway is available as long as VCL application is running. It is not a reliable solution when there is a need for a permanent 7x24 web server.

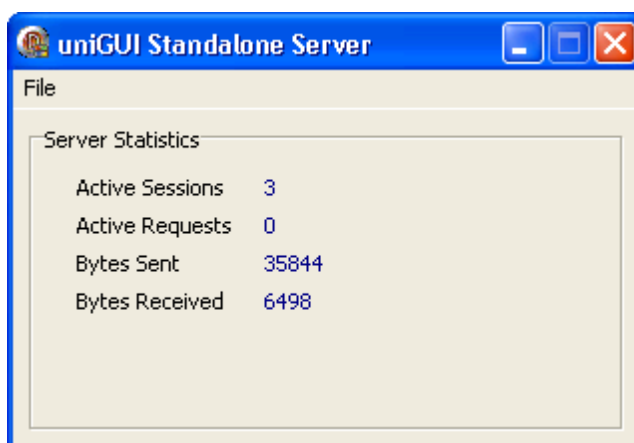


2.1.4 Standalone Server

Standalone Server mode is very similar to VCL Application with some differences which makes it a better option for Web deployment. In this mode your application main form is no longer visible on the desktop, instead an icon is placed in Windows taskbar.



Double-clicking on this icon will open application control panel. Below you see an initial version of control panel.

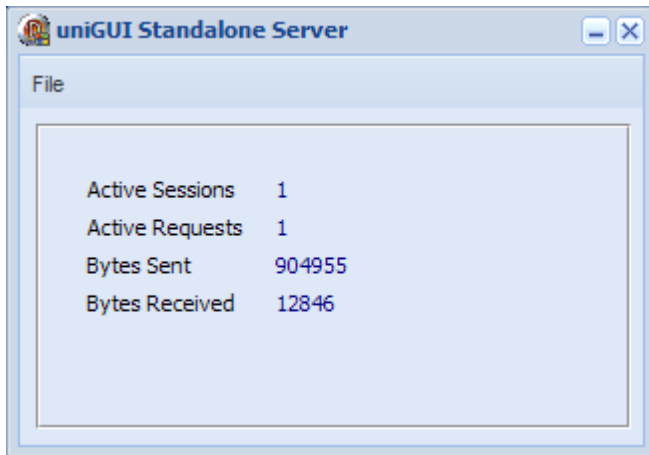


You can shutdown the server by right-clicking the icon and selecting **Shutdown** menu item.



One of the advanced features in uniGUI is accessibility of the Control Panel from web. You can access control panel from this URL:

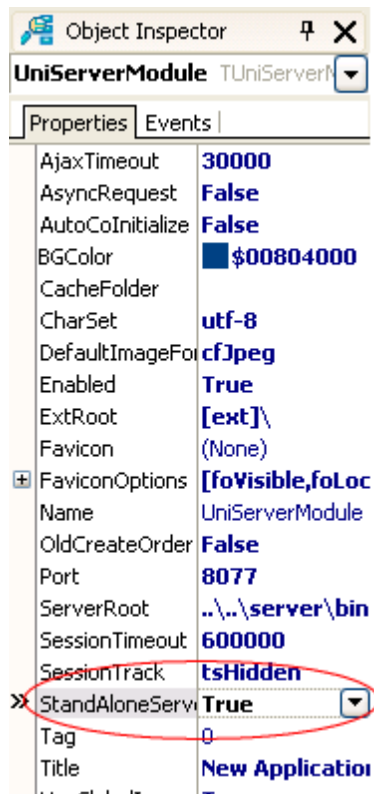
<http://mysite:port/server>



Default Icon can be customized by either Changing the Delphi Application Icon or assigning a new Icon to ServerModule->Favicon

Standalone Server is a good option when you need a web server where server availability is not very important. To automatically start the server you must place a shortcut to server executable in Windows **Startup** folder. No need to mention that server will not start until a Windows user logs in.

In order to turn your application to a Standalone Server simply open the **ServerModule** Unit and set the **StandAloneServer** parameter to **True**.



2.1.5 ISAPI Module

Deploying your Internet application as ISAPI module probably is best method of deployment. You can run several modules together without a need to dedicate a TCP port to each application.

You can use all Web Servers which support ISAPI extentions. uniGUI generated modules are tested with IIS 5.1, IIS 6.0, IIS 7.0 and Apache 2.2.

Installing instructions are different for each Web server. Please refer to below sections for instructions:

[IIS 5.1](#)

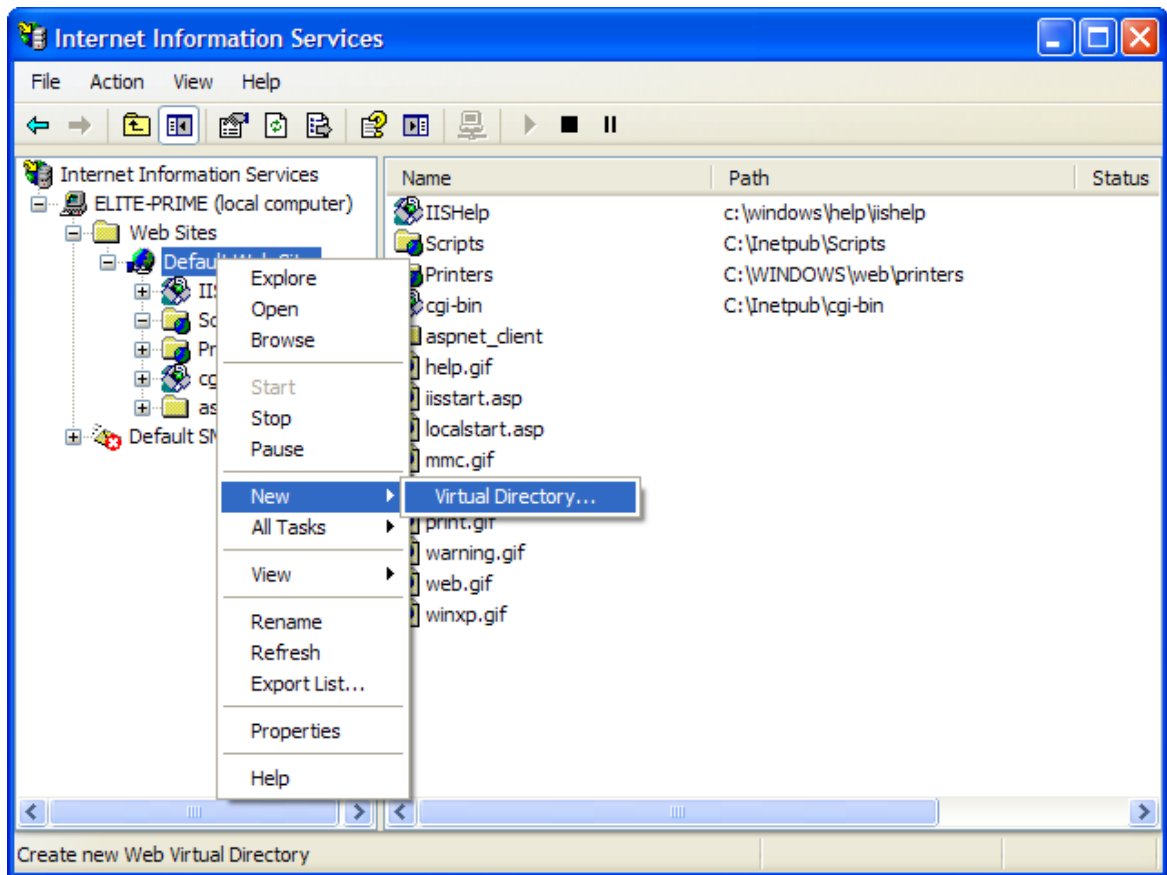
[IIS 6.0](#)

[IIS 7.0](#)

[Apache 2.2](#)

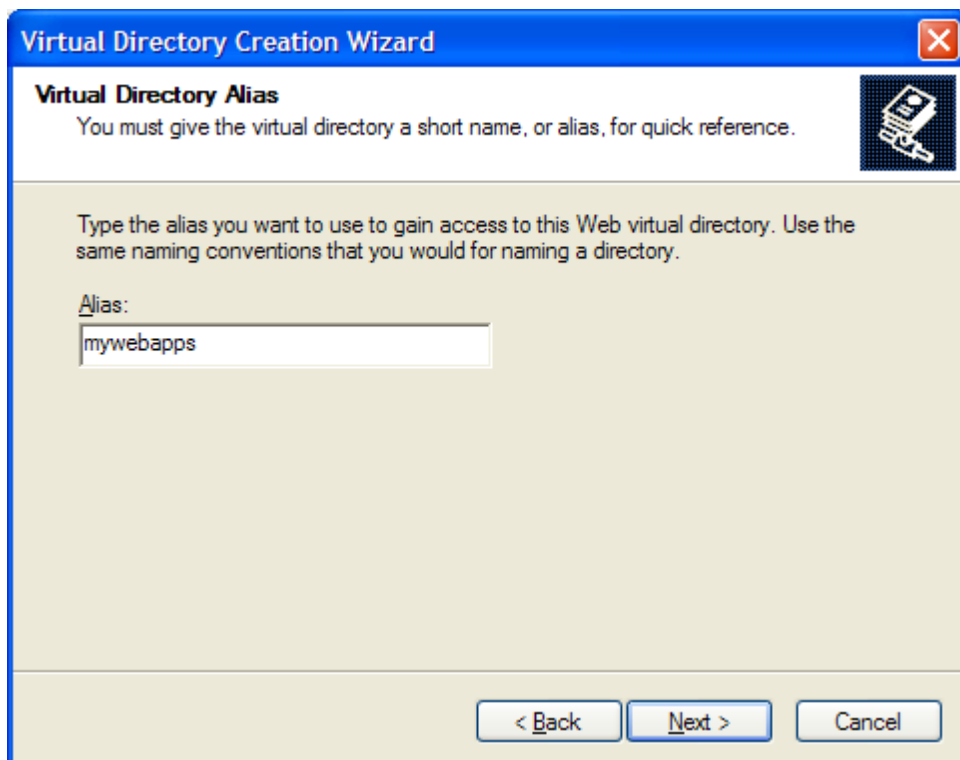
2.1.5.1 IIS 5

First step is to create a new Virtual Directory.

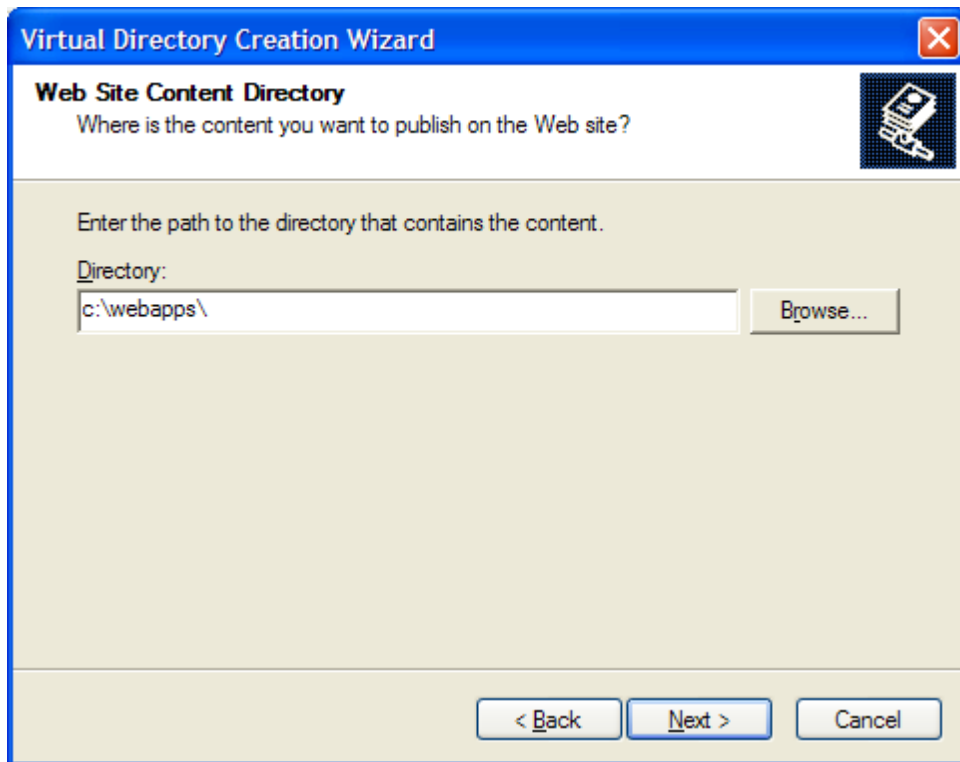




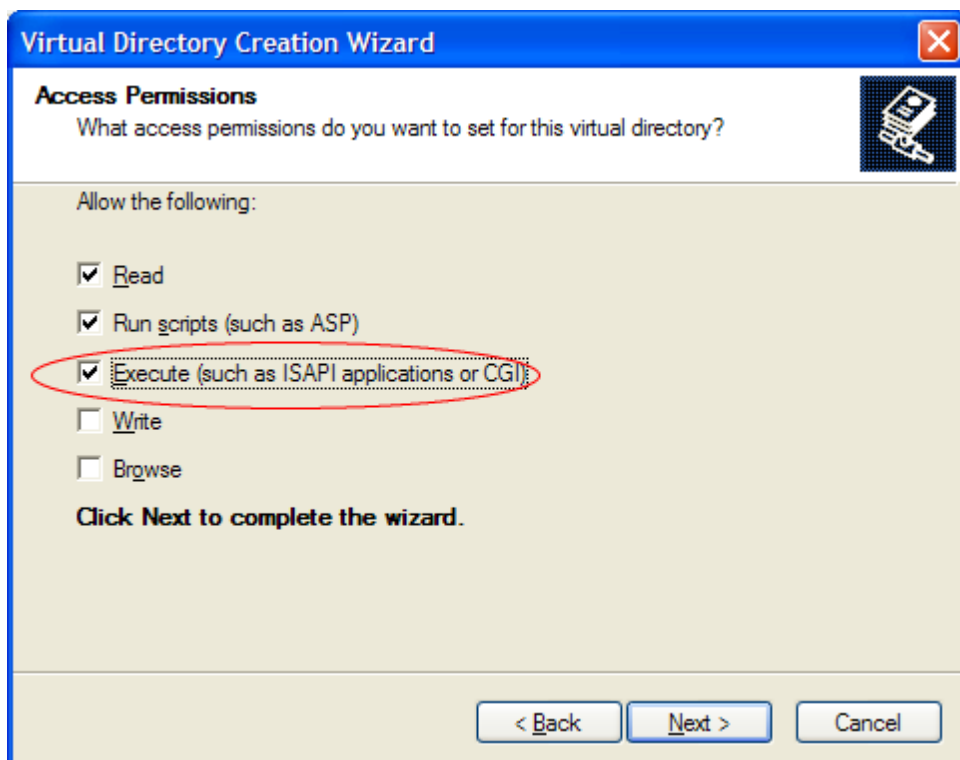
Select an alias for new virtual Directory.

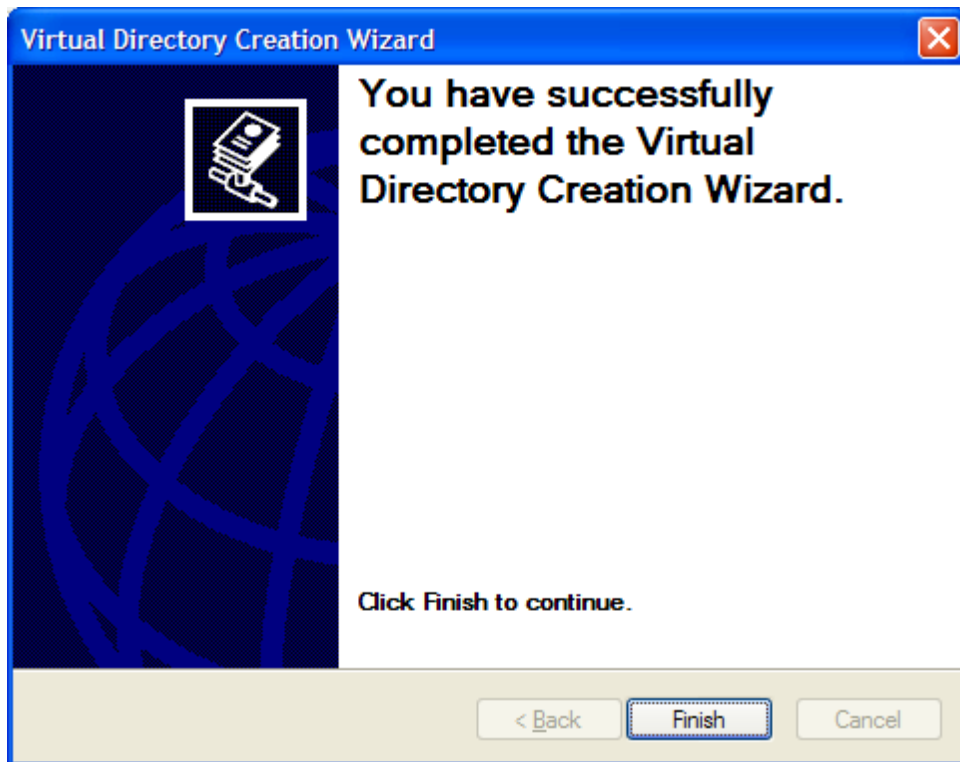


Now assign a folder to newly created alias.



In next step give necessary permissions. **Execute** permission is required.





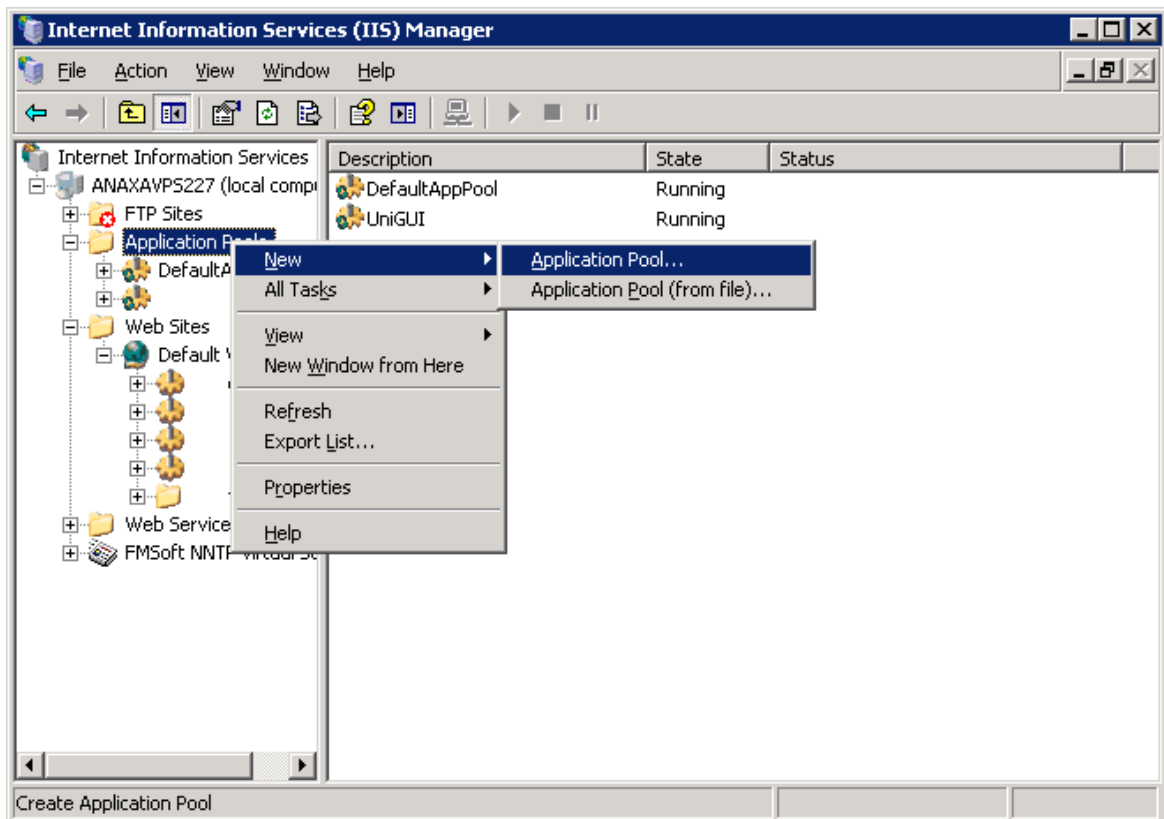
After creating a new virtual directory you are ready to deploy your uniGUI server. Copy your ISAPI module and other required files to virtual directory. You must be certain that **ISS** built-in user **IUSR_<ComputerName>** has enough credentials to access your virtual directory and other folders that may be accessed during web application execution.

Your web application can be loaded by browsing to this URL: *http://localhost/<virtualdirectory>/<module name>.dll*

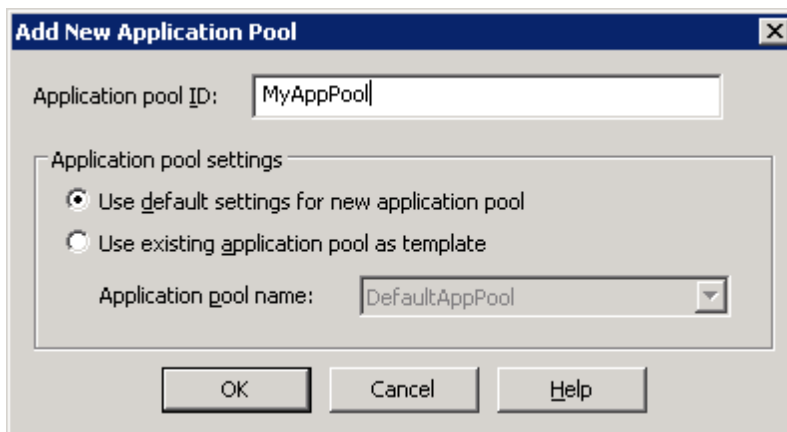
Also refer to [Adjusting Paths](#) for more information.

2.1.5.2 IIS 6

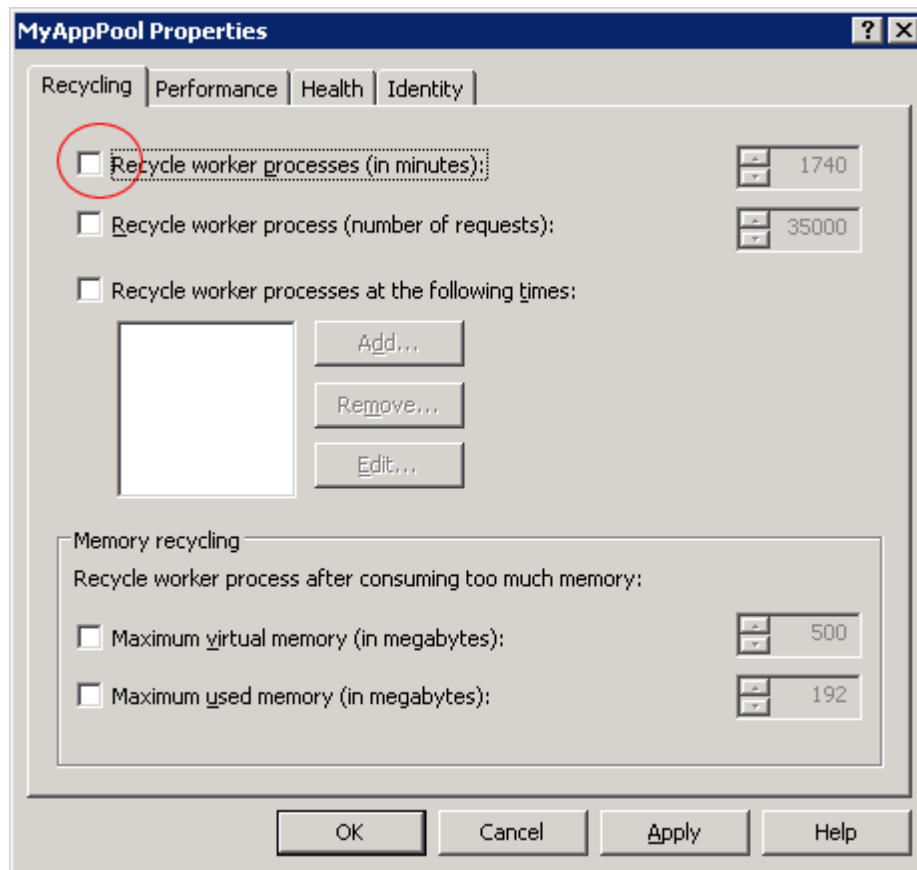
In IIS 6 first step is to create a compatible application pool.



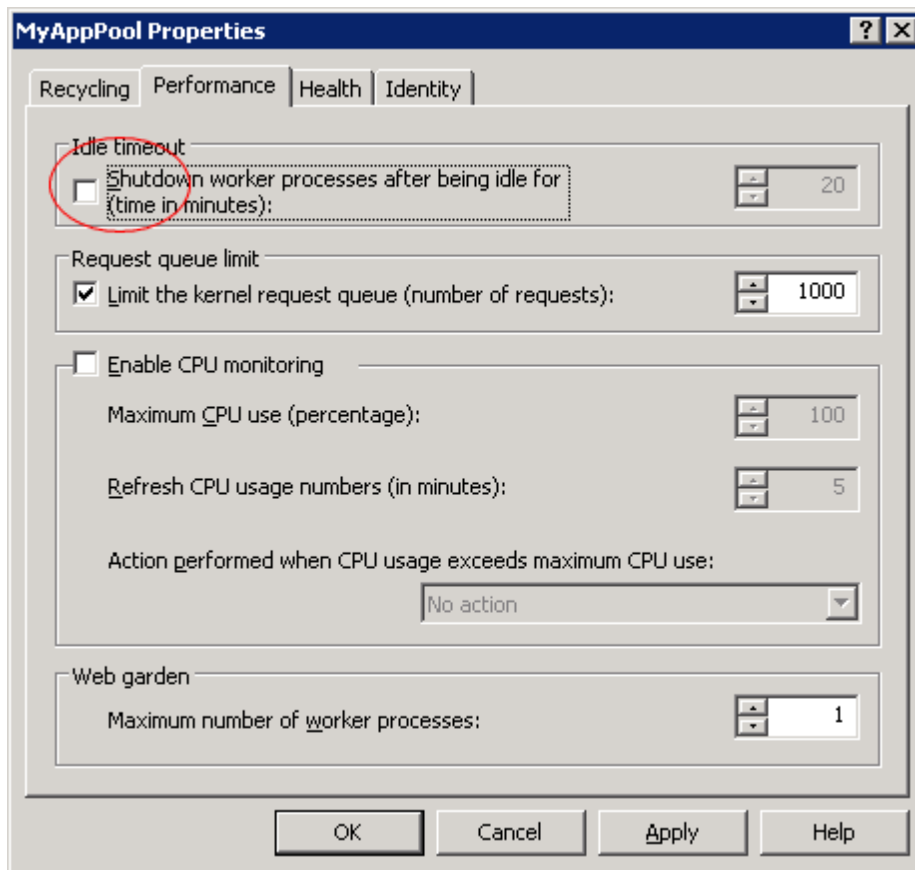
Give a proper name to the new pool.



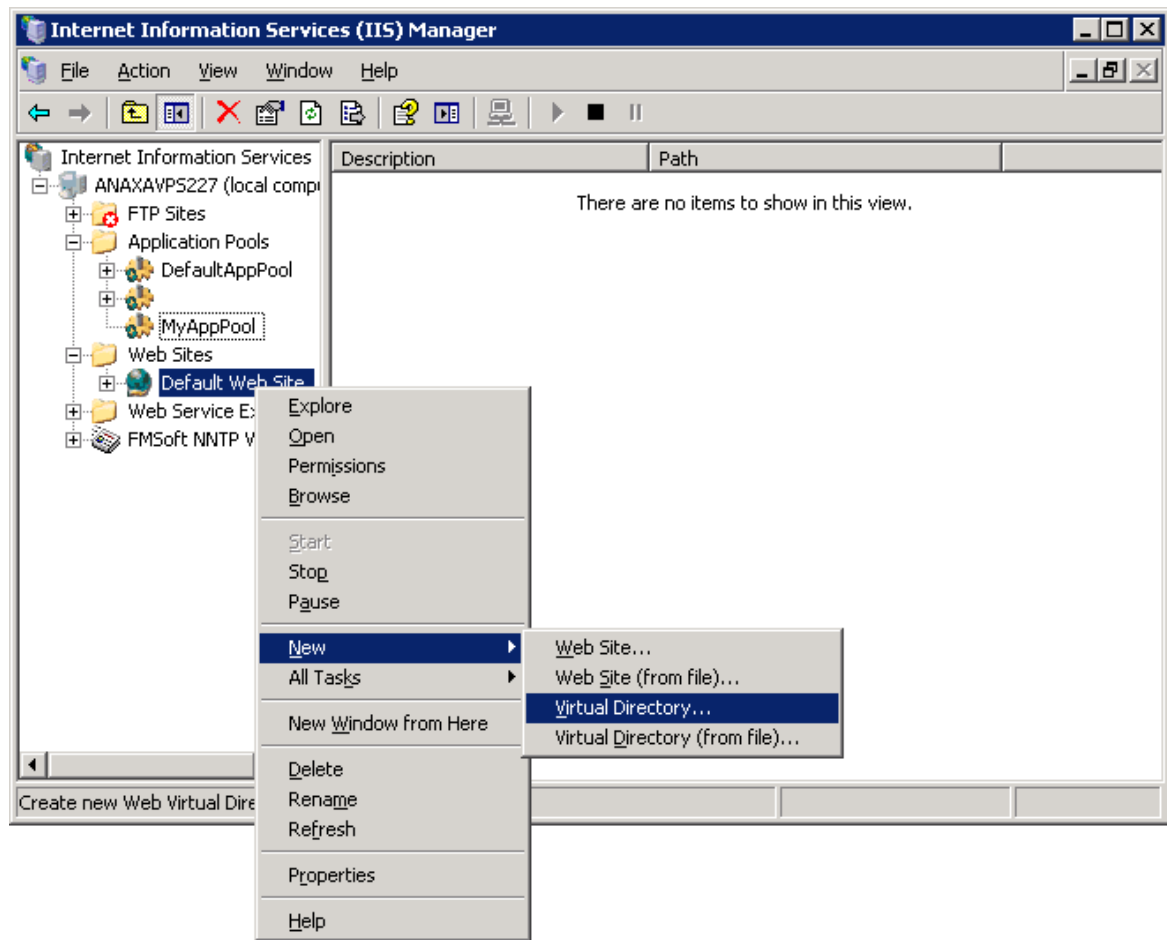
In Application Pool properties, Recycling Tab uncheck the **Recycle worker processes** option.



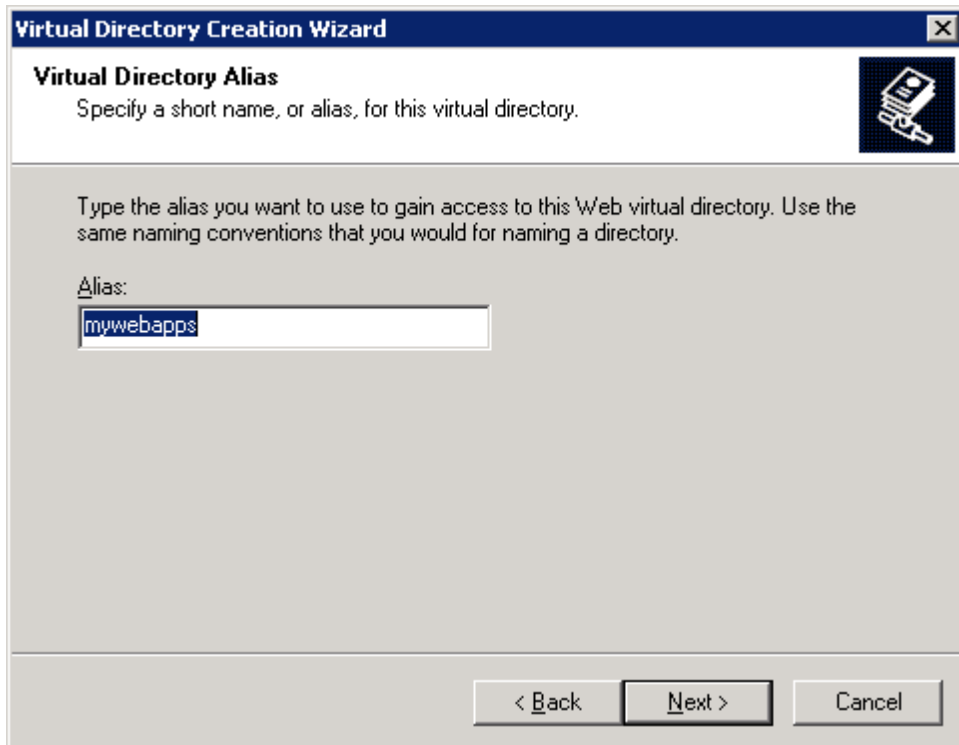
In Performance Tab uncheck the **Shutdown worker processes after being idle...** option.



Next step is to create a Virtual Directory.



Assign an alias to new virtual directory.



Virtual Directory Creation Wizard

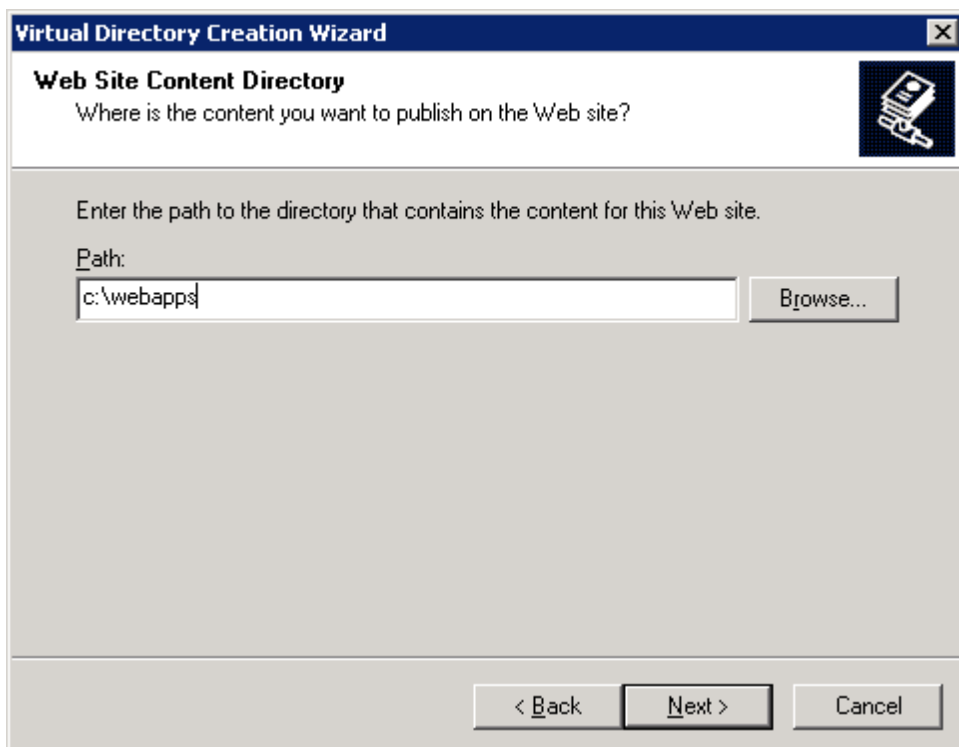
Virtual Directory Alias
Specify a short name, or alias, for this virtual directory.

Type the alias you want to use to gain access to this Web virtual directory. Use the same naming conventions that you would for naming a directory.

Alias:

< Back Next > Cancel

Select the folder where your ISAPI modules are located.



Virtual Directory Creation Wizard

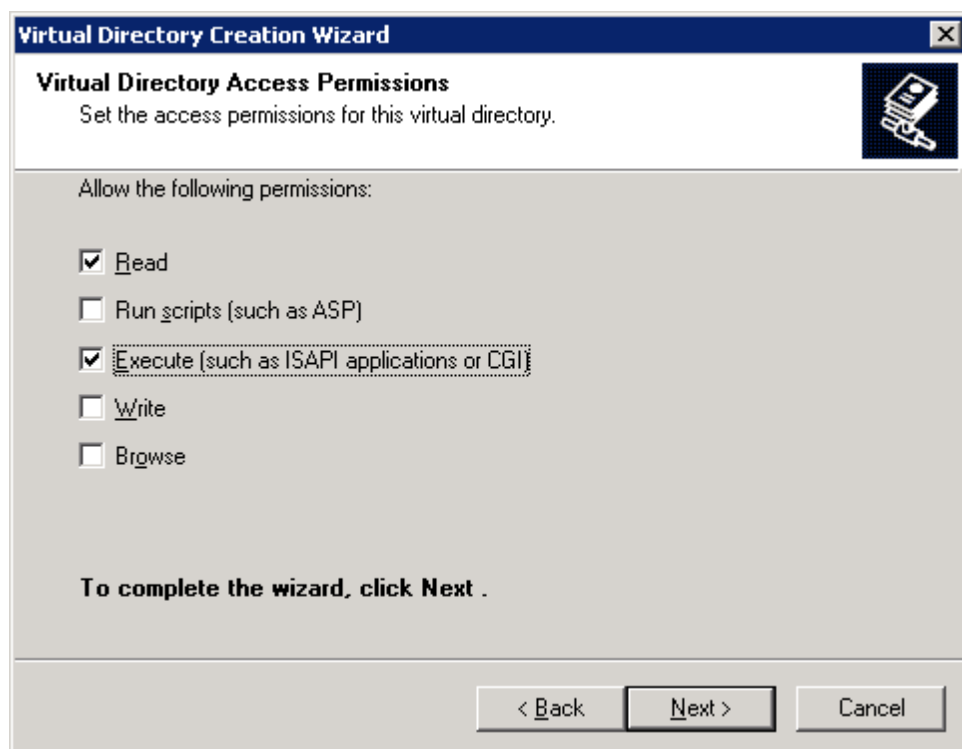
Web Site Content Directory
Where is the content you want to publish on the Web site?

Enter the path to the directory that contains the content for this Web site.

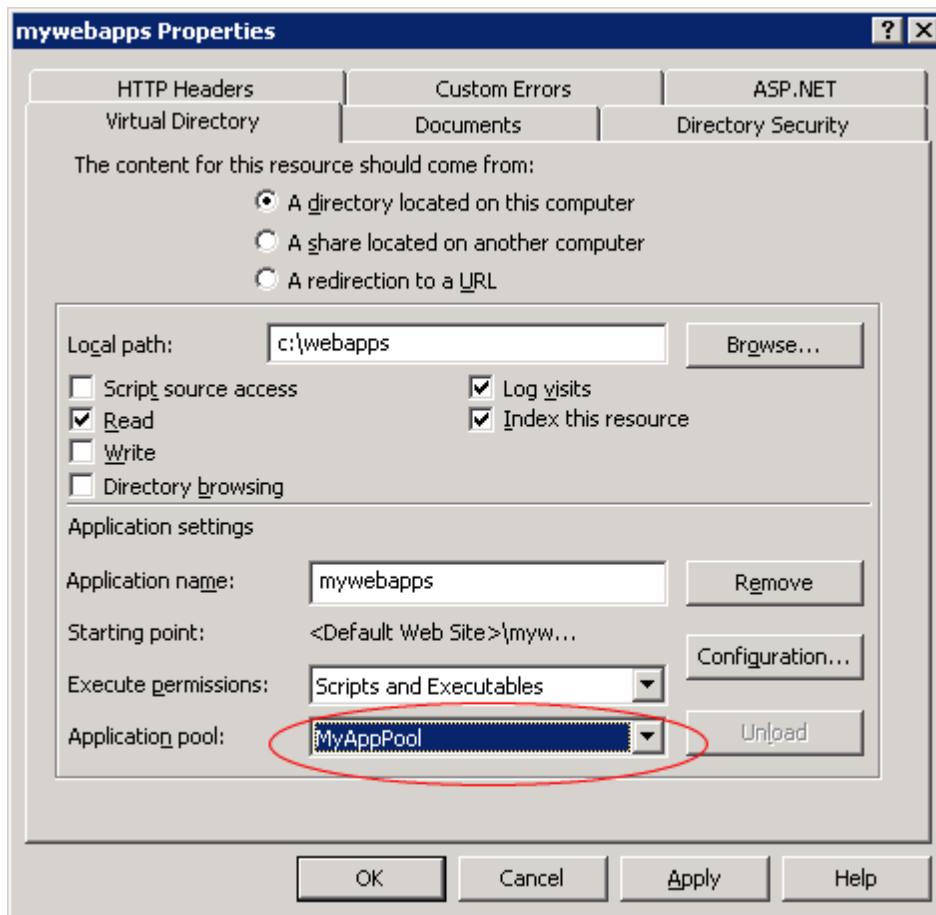
Path:
 Browse...

< Back Next > Cancel

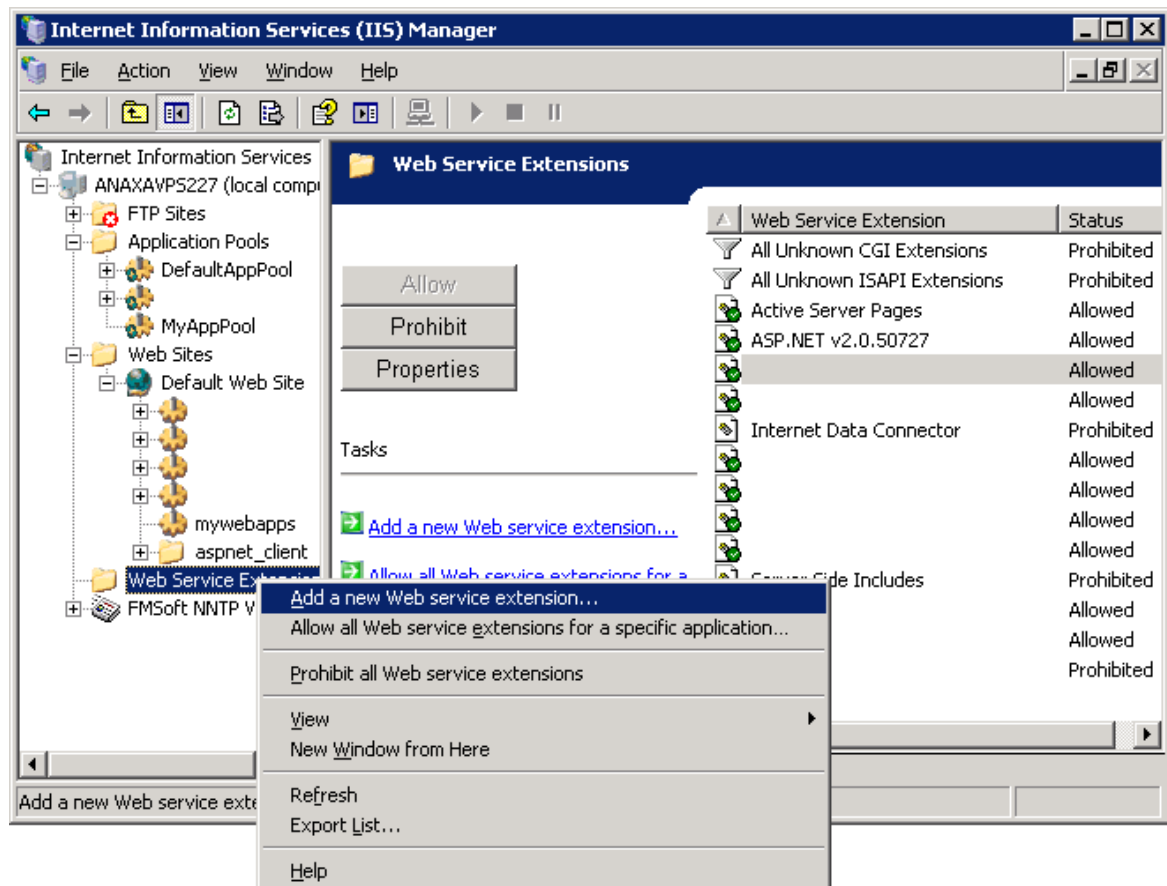
Be sure to grant **Execute** permission to Virtual Directory.



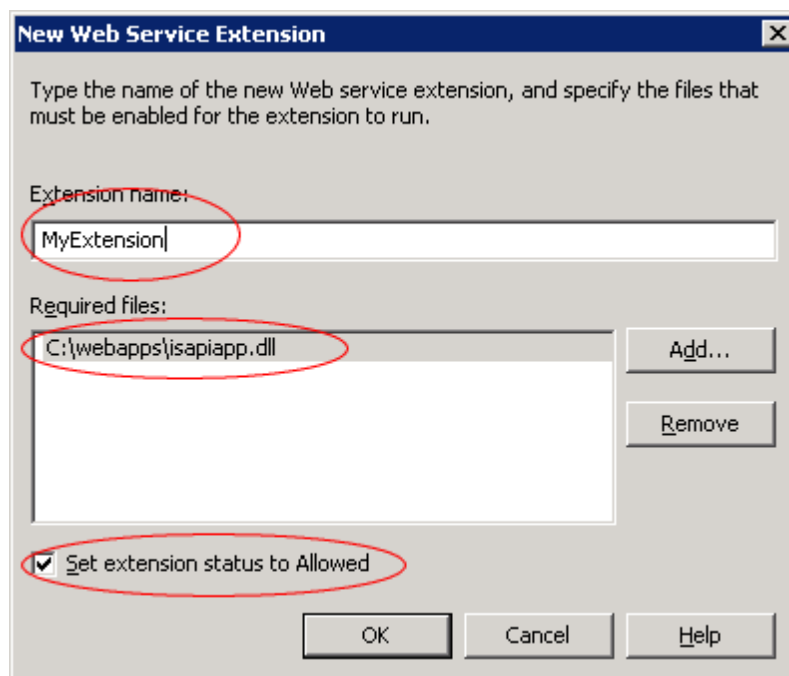
Now open the properties of newly created Virtual Directory and change the default pool to pool you created in first step.



There is one further step in IIS 6. Your ISAPI extension must be added to the list of allowed extensions.



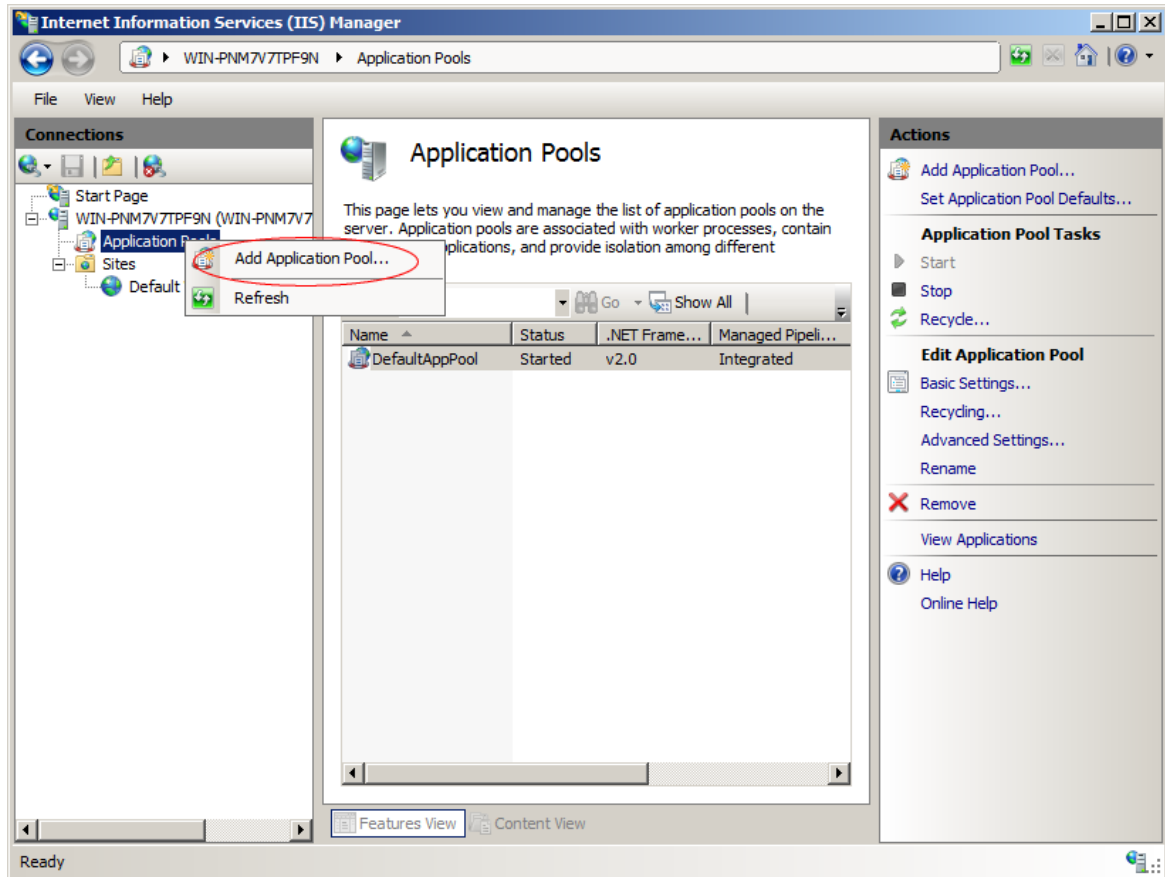
Assign a name to your extension, add the extension's module DLL file and check the **Set extension status to Allowed** option.



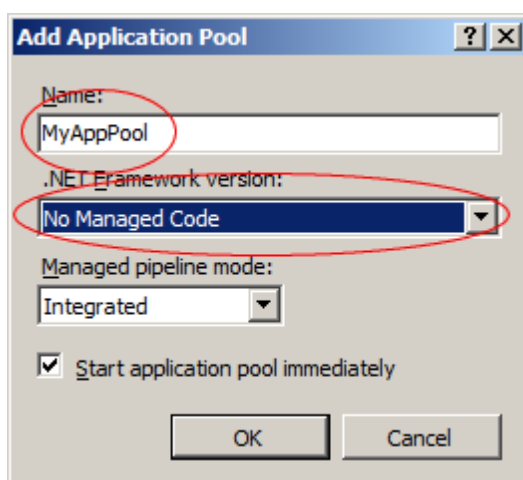
Also refer to [Adjusting Paths](#) for more information.

2.1.5.3 IIS 7

Like ISS 6, in IIS 7 first step is to create a new Application Pool.

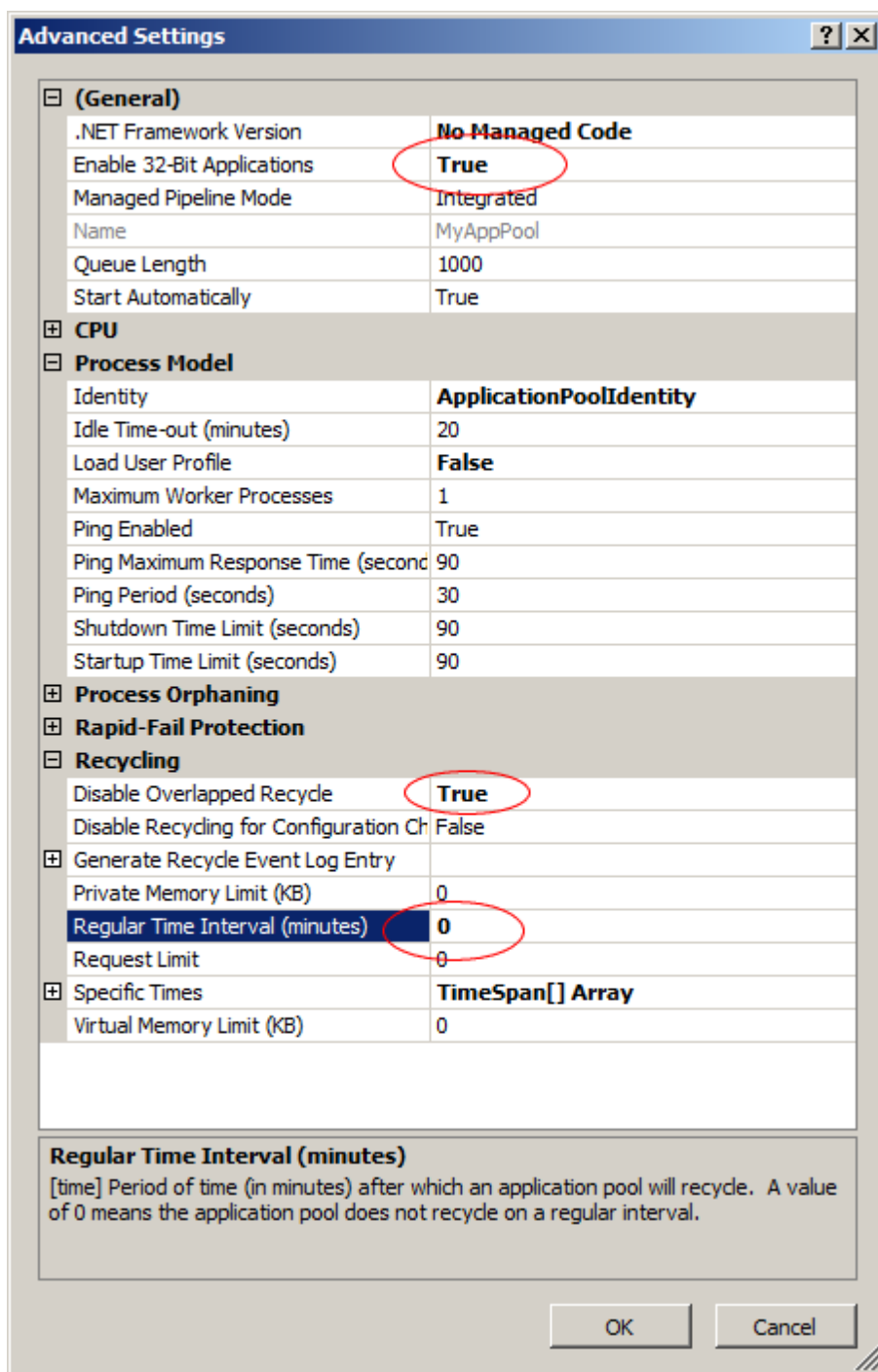


Assign a name to your pool and select **No Managed Code** option.

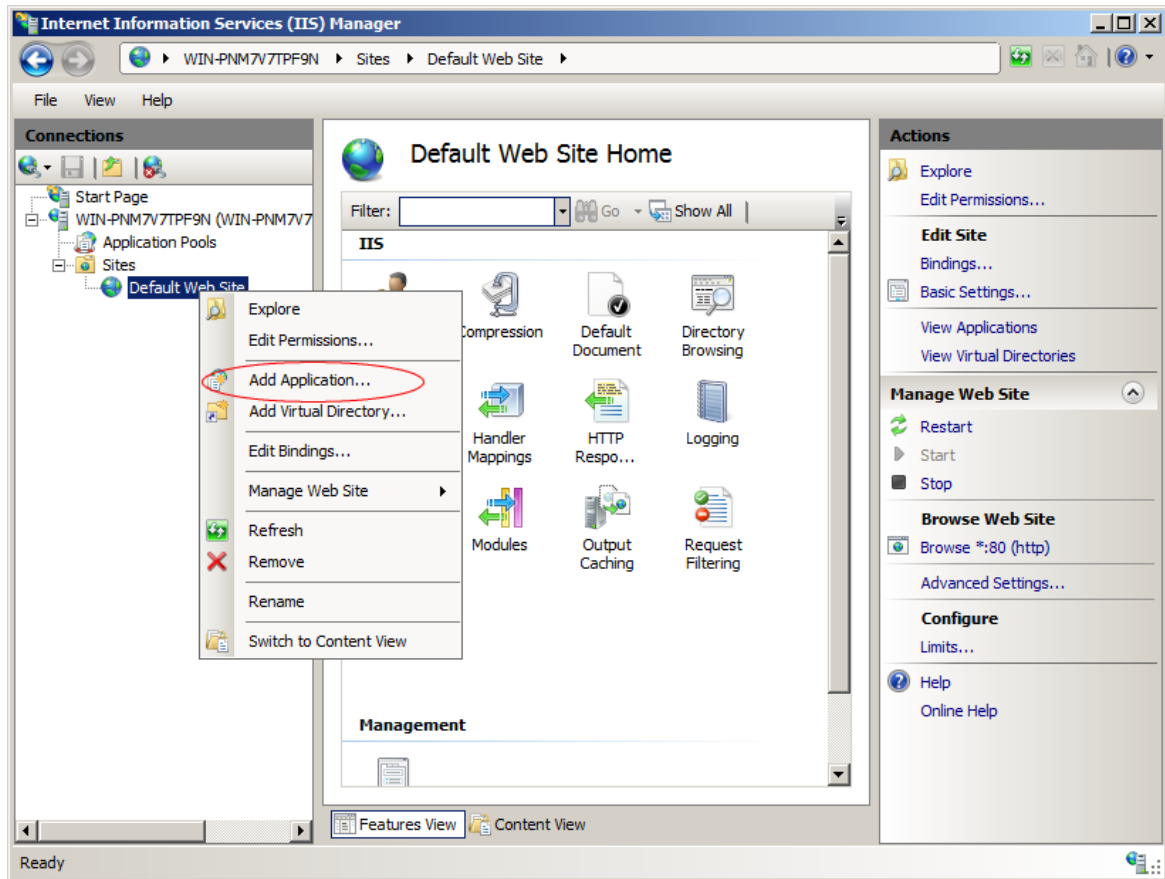


After creating the Pool open Pool's Advanced settings dialog and make the following modifications:

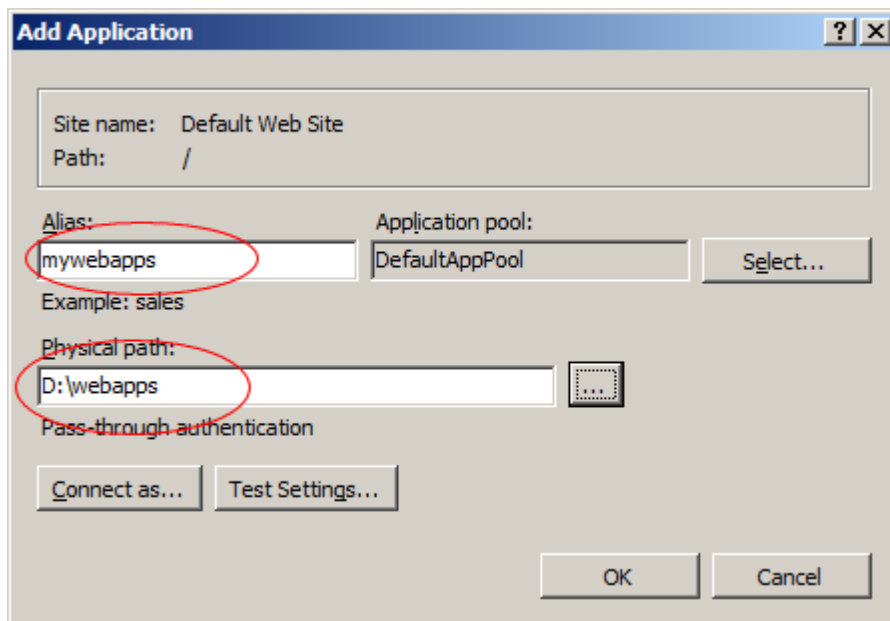
- Set **Enabled 32-Bit Applications** to **True**. (This option is available in 64-bit versions of Windows)
- Set **Disable Overlapped Recycle** to **True**.
- Set **Regular Time Interval** to **0**



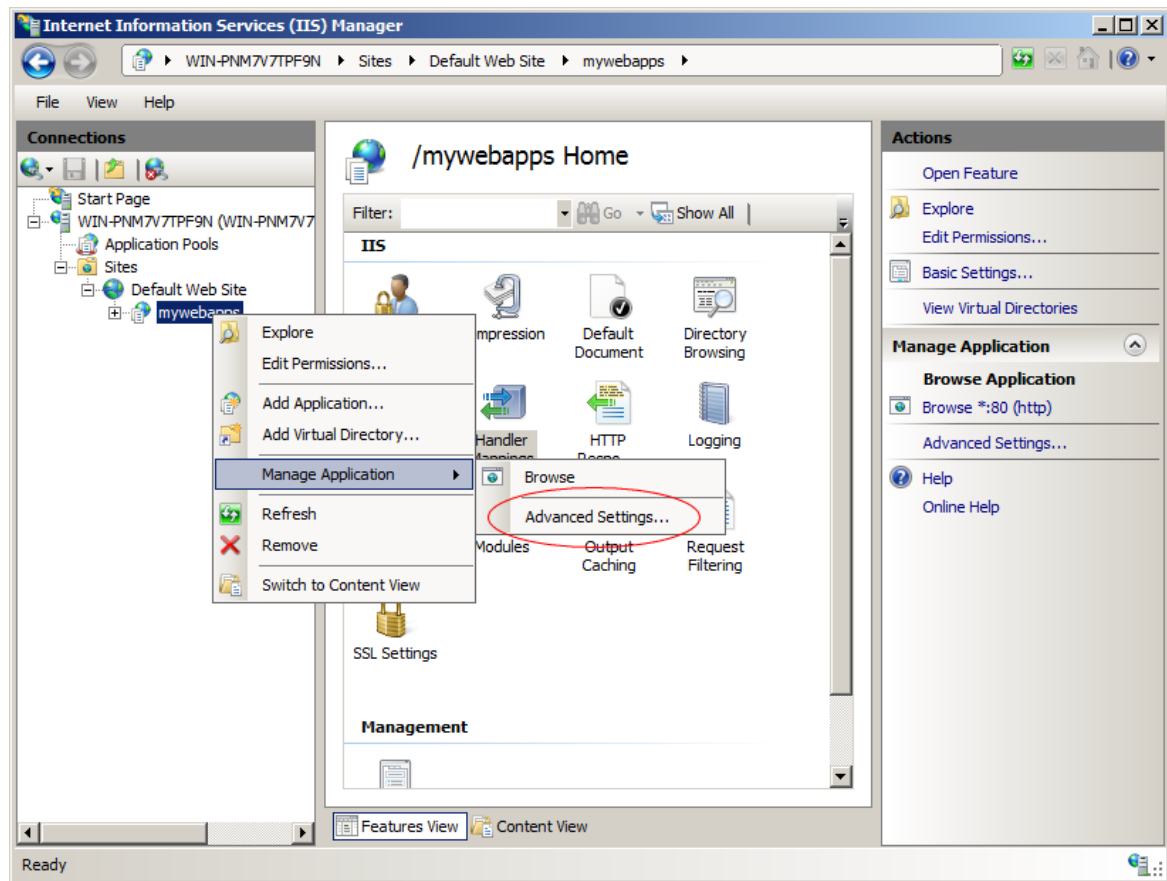
Now add a new Application.



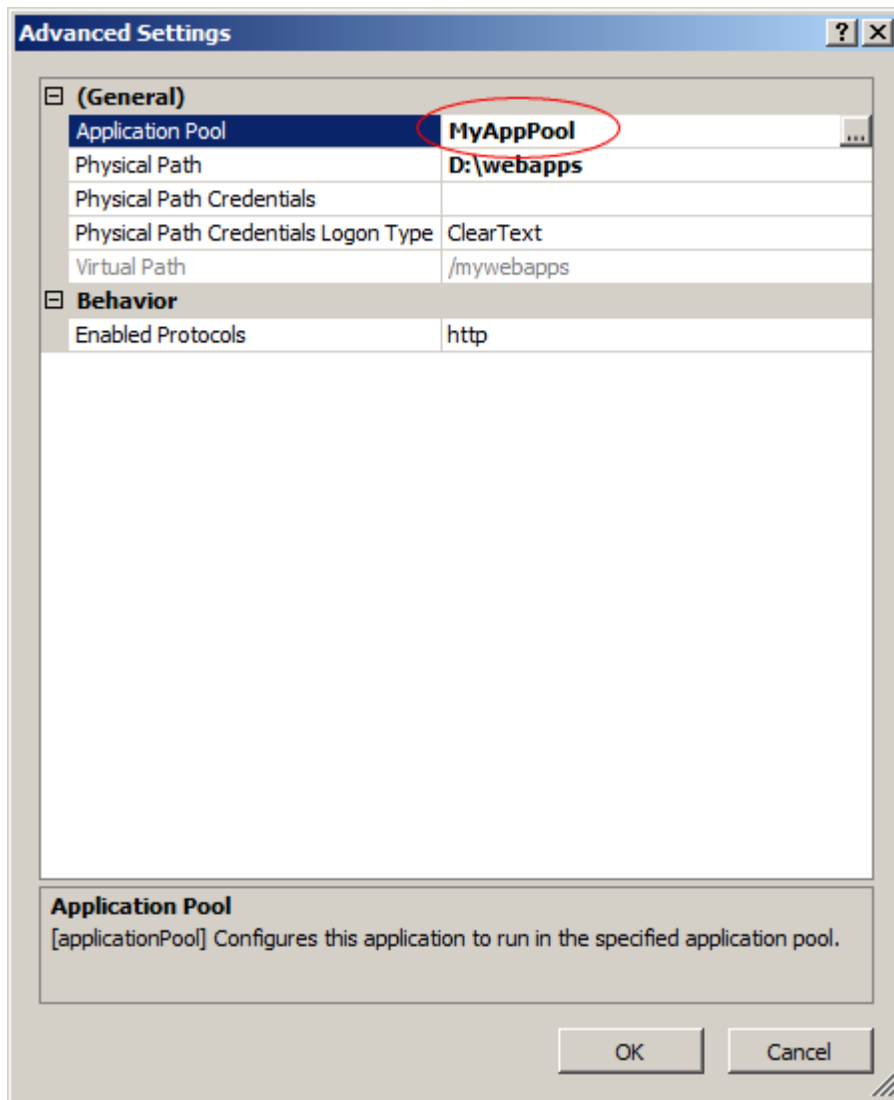
Give it a proper name and adjust the **Physical path**. It is the path where your module DLL files exist.



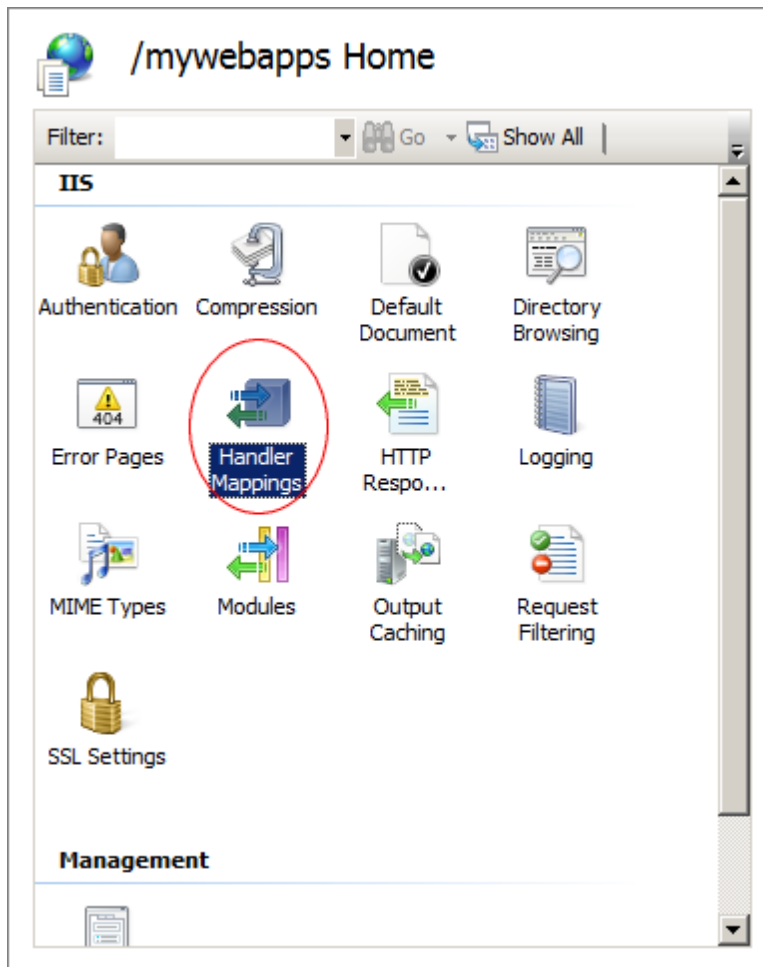
Select Advanced Settings menu and open Advanced Settings dialog.



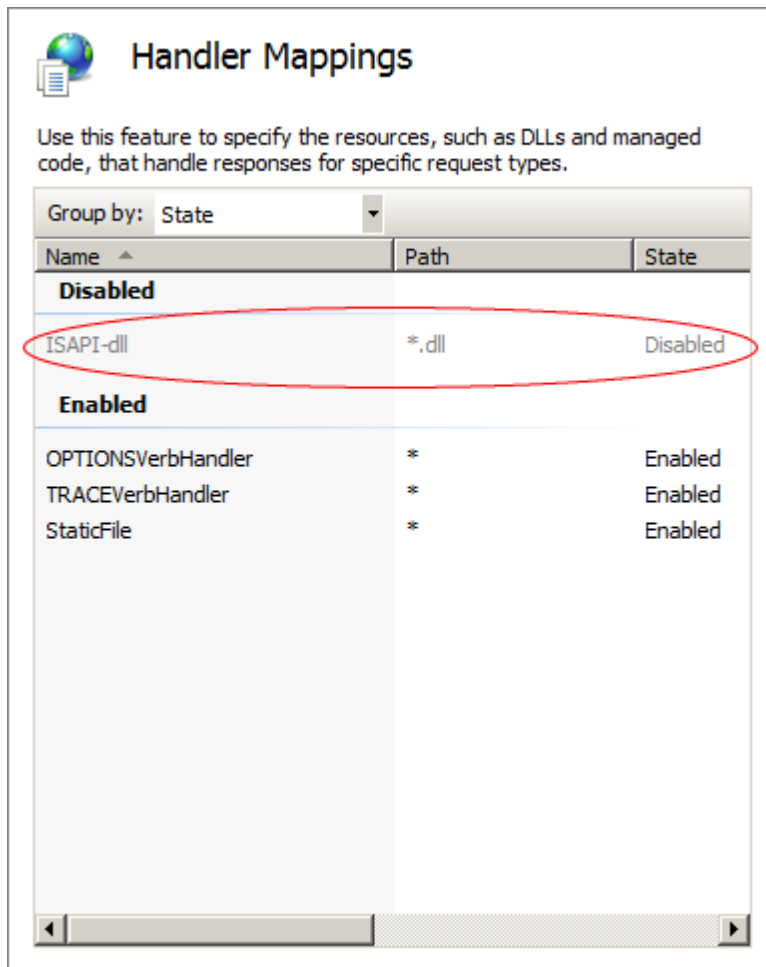
In Advance Settings screen set the **Application Pool** to one you created in first step.



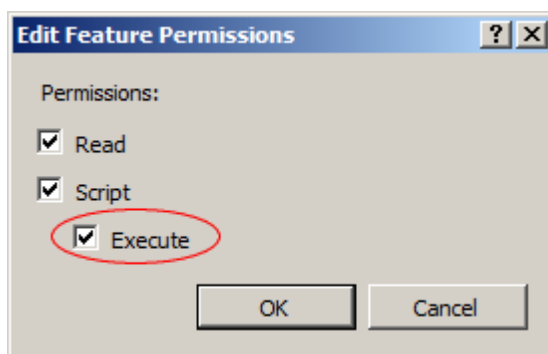
Next step is to adjust the handler mapping for the application you just created.



In Handler Mappings screen right-click on ISAPI-dll and select **Edit Feature Permissions**



Check the **Execute** option and press OK.



Also refer to [Adjusting Paths](#) for more information.

2.1.5.4 Apache 2.2

Apache 2.2 web server for Windows allows running ISAPI modules. For this, a plugin called **mod_isapi** must be enabled.

Apache doesn't have a visual interface for configuration. You must do some modifications to **httpd.conf** file.

First of all, uncomment the following line:

```
LoadModule isapi_module modules/mod_isapi.so
```

Next you must associate **.dll** files with ISAPI module.

Add the following line

```
AddHandler isapi-handler .dll
```

Next step is add your module directory to Apache directory entries.

```
<Directory "C:/webapps">
    Options Indexes FollowSymLinks ExecCGI
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Finally create a new **Alias** for your directory.

```
Alias /mywebapps "C:/webapps"
```

Be sure to restart your Apache server after making the necessary modifications to **httpd.conf** file.

2.1.6 Windows Service

Windows service is created and deployed like other regular Service applications created using Delphi.

To install service simply type following command at command line:

```
MyServiceApp -install
```

You can start service from Windows service manager or type the following command:

```
net start ServiceName
```

When you create a new project default value for service name is: **UniServiceModule**
You can change it from **ServiceModule.pas->UniServiceModule->Name**

To uninstall service:

```
MyServiceApp -uninstall
```

2.1.7 SSL Configuration

uniGUI supports SSL protocol for Standalone Server and Windows Service applications. Let's emphasize that built-in SSL support is only for Standalone and Windows Service applications. In ISAPI mode you must configure SSL by configuring your particular ISAPI server; **IIS**, **Apache** and etc.

First step to configure SLL is to obtain the required certificate files. In Standalone and Service modes uniGUI uses [Indy](#) as the underlying TCP transport layer.

There are three files which you must provide here:

root.pem
cert.pem
key.pem

As you can see files are in [pem](#) format. **Pem** files are a human readable **base64** ascii files which can be opened and edited in an editor. A pem file may contain a single certificate or more than one certificate. In order to work with **Indy** each **pem** file must contain only one certificate.

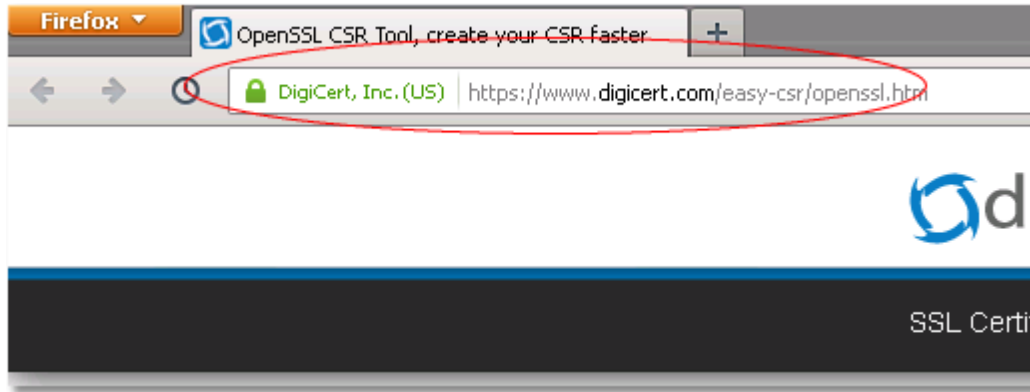
Below sample shows contents of a **pem** file:

```
-----BEGIN CERTIFICATE-----
MIIB8jCCAV+gAwIBAgIQfjGd2Py0qZJGqdkPiRlDdjAJBgUrDgMCHQUAMBAxDjAM
BgNVBAMTBWVsaXRlMCAXDTEzMDYwMjE3NTA0OfoYDzIxMTMwNTA5MTC1MDQ4WjAQ
MQ4wDAYDVQQDEwVlbG10ZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAt3pi
pMYHNzUueLZBb1eMrPop6Emta/KLyLaK94v1M1lV/6ITiuFtuSs9gq0s516s2th7
FUkBpgfQvrb+3b9h10WMca8MTbYrLGL+dHqRk4jGt/8GAUeYHkKddk/NeXkZWqaD
3aMdURpTgE2iK/d86C1YdsxqXTxP+Uax/eN4RUECAwEAAaNTMFewFQYDVR01BA4w
DAYKKwYBBAGCNwoDBDAkBqNVHREEJjAkoCIGCisGAQQBgjcUAgoFawSZWxpGVA
RUxJVEUtUFJTTUAMakGA1UdEwQMAAwCQYFKw4DAh0FAAOBgQCDSHm54tMh1sPY
aBrpZeZtbt9e1gPZ2B/Gd7U2KGK46yM8OQQ3LlnPaTc96q2ocD9sL3GP1B2itwX/
THOGUX7MpUipfUg6+8te6A7//gjiGyCf/OauJJrHal8p2QPwecGo3YnxUvTCu9gH
+iGE3Yqxv/6YqgDjGnpNdAvvX9gEfQ==
-----END CERTIFICATE-----
```

There are two types of certificate you can use along with your uniGUI server. You can either use a certificate issued by a **certificate authority** or you can use a **self-signed certificate** for development, test and / or private use.

2.1.7.1 Obtain a SSL Certificate from an Authority

SSL certificates are published by companies which are specialized in issuing and hosting SSL certificates. In order to obtain such a certificate you need to make a contract and purchase a certificate dedicated to your company. This certificate will be known to whole web and will contain information regarding your company, the certificate issuer and your web site.



Example for a secured web site

First step is to create a **CSR** (Certificate Signing Request) file. There are several tools to create a CSR file. [OpenSSL](#) is best tool which is widely used for such tasks. After you created the CSR file you must send it to your certificate authority and they will verify information you provided in the CSR file. Upon a successful verification they will sign your certificate and issue required certificate files.

Your certificate authority will guide you in process of creating a CSR file and rest of application process.

After a successful application you will receive your certificate files. As described in previous section you need three files to distribute with your server: **root.pem**, **cert.pem** and **key.pem**.

Note: sometimes **cert.pem** and **key.pem** can be issued as a single file. In this case you need to open this file with a text editor and save the individual certificates in separate **pem** files.

2.1.7.2 Generate a Self-Signed Certificate

This kind of certificate is good when you don't need a globally signed certificate issued by a certificate authority such as [Verisign](#). You can use a self-signed certificate for development purpose or for private use in your intranet network or over the internet. You can use [OpenSSL](#) to generate related certificate files.

First of all download and install **OpenSSL** Windows binaries from [here](#). After installing open a command prompt and follow below instructions.

We recommend downloading the lite version of the binaries:

[Win32 OpenSSL v1.0.1e Light](#)

[Win64 OpenSSL v1.0.1e Light](#)

a) Generate a self-signed Root certificate.

If you already have a root certificate installed in Windows you can try exporting it instead of generating a new one. Simply go to **Control Panel** and click the **Internet Options -> Content -> Certificates**. Select the root certificate you want to export. Choose the **base64** format and select folder and file name to export. This will export your root certificate in **.cer** format which can safely rename it to **.pem** and use it in your uniGUI project.

You can also create a root certificate from scratch.

At command prompt issue following command:

```
openssl genrsa -out root.key 1024
```

This will create a new **root.key** file with strength of 1024 bit. Other options are 2048 and 4096. Normally 1024 bit is enough.

If you want to create a root key with a password use below command instead:

```
openssl genrsa -des3 -out root.key 1024
```

Next step is to self-sign the certificate.

```
openssl req -x509 -days 365 -new -nodes -key root.key -out root.pem
```

If your root key is created with a password use below command instead:

```
openssl req -x509 -days 365 -new -key root.key -out root.pem
```

Now you will be prompted to provide several details needed to sign your certificate. You will also be prompted for a password if your root.key file is created with a password in first step.

Enter pass phrase **for** root.key:

You are about to be asked to enter information that will be incorporated into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

 Country Name (2 letter code) [AU]:**TR**
 State or Province Name (full name) [Some-State]:**Ankara**
 Locality Name (eg, city) []:**Cankaya**
 Organization Name (eg, company) [Internet Widgits Pty Ltd]:**FMSOft**
 Organizational Unit Name (eg, section) []:**R&D**
 Common Name (eg, YOUR name) []:**Farshad Mohajeri**
 Email Address []:**info@fmsoft.net**

Note: **365** is the number days which certificate will remain valid.

This will place a new **root.pem** file in same folder. This file will be used in your uniGUI server.

b) Generate a self-signed key.

Next step is to generate a self-signed key. This step will produce the **key.pem** and **cert.pem** files.

At command prompt issue the following command:

```
openssl req -x509 -days 365 -nodes -newkey rsa:1024 -keyout key.pem -out cert.pem
```

Again, you'll be prompted to answer several questions.

```
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:TR
State or Province Name (full name) [Some-State]:Ankara
Locality Name (eg, city) []:Cankaya
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FMSOft
Organizational Unit Name (eg, section) []:R&D
Common Name (e.g. server FQDN or YOUR name) []:Farshad Mohajeri
Email Address []:info@fmsoft.net
```

To create same key with a password use below command:

```
openssl req -x509 -days 365 -newkey rsa:1024 -keyout key.pem -out cert.pem
```

This time you'll be prompted to enter a password:

```
Enter PEM pass phrase:  
Verifying - Enter PEM pass phrase:
```

This password will be assigned to SSL->SSLPassword parameter of UniServerModule. (See [SSL Configuration](#))

When all above procedures are completed you will end up with three files named **root.pem**, **key.pem** and **cert.pem** which are required to setup and run your project in **SSL** mode.

2.1.7.3 Configure SSL Parameters

Copy all three **pem** files to same folder your **uniGUI** server executable resides.

You also need **OpenSSL** standard **DLL** library files.

```
libeay32.dll
ssleay32.dll
```

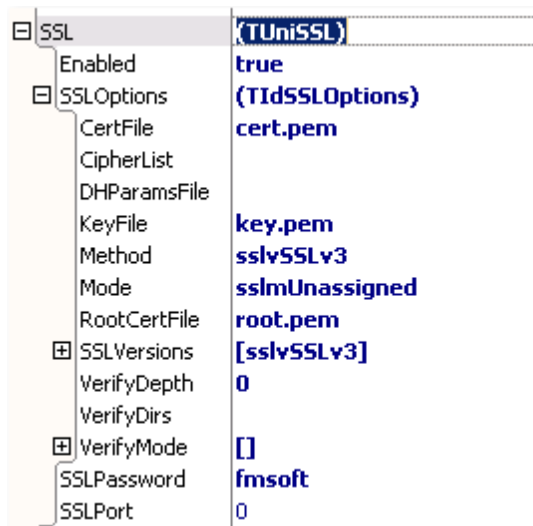
These **DLL** files are distributed as a part of **OpenSSL** installation. You can also find the most recent version of **OpenSSL DLL** files under below folders:

```
[UniGUI Installation Folder]\..\Framework\uniGUI\SSL\dll\x86
[UniGUI Installation Folder]\..\Framework\uniGUI\SSL\dll\x64
```

Depending on how you deploy your app you can copy them under Windows system folder (such as C:\Windows\System32) or put them in same folder as your **uniGUI** server.

Note: There is a chance that above dll files are already installed by other programs and already present in your system path. In this case you must make sure that most recent versions of dlls are installed.

Next step is to open your project's **ServerModule** and make the following changes:



```
SSL->Enabled = True
SSL->SSLOptions->CertFile = cert.pem
SSL->SSLOptions->KeyFile = key.pem
SSL->SSLOptions->RootCertFile = root.pem
```

If you have protected your key files with a password then you need to assign it here:

```
SSL->SSLPassword = [Password]
```

Next parameter to configure is the SSL port. You can leave it to default value which is **0**. In this case you can access your server through port which is configured in **ServerModule -> Port** which default value is 8077.


```
https://localhost:8077
```

Or you can use the default port for SSL which is **443**. Since **443** is the default value you can omit the port number in address:

```
https://localhost
```

Normally you only want to use **https** protocol for a secured web application, but if for any reason you need both **http** and **https** protocols for same site you can enable them by assigning different ports to each protocol.

Consider the following configuration:

```
ServerModule->SSL->SSLPort = 443  
ServerModule->Port = 8077
```

In above configuration you are able to access standard **http** protocol from port **8077** and access **https** protocol from port **443**.

If you choose the default port of **443** for SSL make sure neither **IIS** nor any other web server software is not listening on this port.

Index

- E -

ExtRoot 14

Endnotes 2... (after index)