

uniGUI Developer's Manual

© 2015 FMSoft Co. Ltd.

Table of Contents

Foreword	0
Part I Installation	3
1 System Requirements	3
2 Installation Instructions	4
3 Sencha Touch Installation	10
Part II Developer's Guide	11
1 Web Deployment	11
Sencha License Considerations	11
Adjusting Paths	11
VCL Application	13
Standalone Server	13
ISAPI Module	15
IIS 5	16
IIS 6	20
IIS 7	28
Apache 2.2.....	35
Windows Service	35
SSL Configuration	36
Obtain a SSL Certificate from an Authority.....	37
Generate a Self-Signed Certificate.....	38
Configure SSL Parameters.....	41
2 Stress Test Tool	43
Introduction	43
Usage	44
Recording a Session.....	44
Running the Stress Test.....	46
Server Flood Protection Considerations.....	49
Additional Settings.....	49
Index	51

1 Installation

1.1 System Requirements

Supported **Delphi*** versions are **Turbo Delphi**, **Delphi 2006**, **Delphi 2007**, **Delphi 2009**, **Delphi 2010** and **Delphi XE-XE7**.

*All **Delphi** versions must be installed with their latest updates and service packs.

C++ Builder is supported but not tested with all Versions.

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

Note 1: Requirement for runtime environment can be very different and will be discussed under deployment topic.

Note 2: In order to install **uniGUI** for **C++ Builder** you need to have **RAD Studio IDE** installed. Installing **C++ Builder** alone is not enough. Please see [Installation Instructions](#) for details.

1.2 Installation Instructions

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

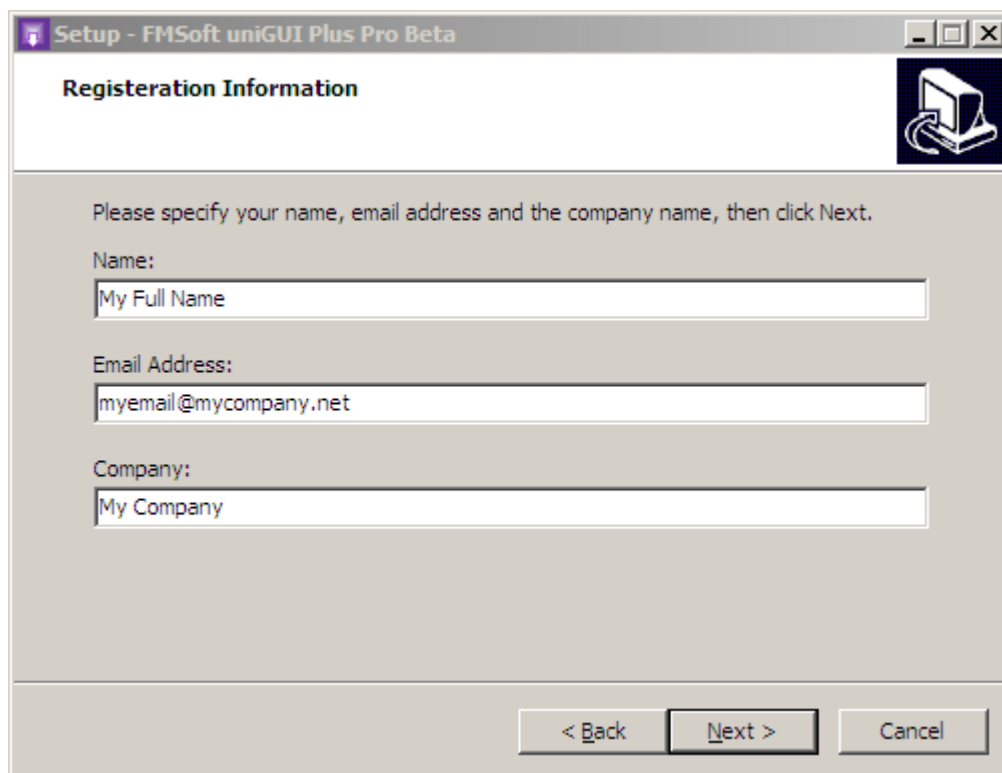
- Before installing a new version remove all design packages from Delphi and uninstall uniGUI from Windows Program Add/Remove.
- After re-compiling an application with this new version, "ext" folder must be re-deployed to PCs running new version of your application or you can simply re-install the newly 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.
You will notice that there are two versions for of setup:

- **FMSoft_uniGUI_{Edition_0.XX.0.YYYY}_Beta.exe**
 - This is the one which will be installed on developer PC for development purpose only.
- **FMSoft_uniGUI_{Edition_runtime_0.XX.0.YYYY}.exe**
 - This one is only for deployment purpose and will be installed on your server PC which your uniGUI apps will run.

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.

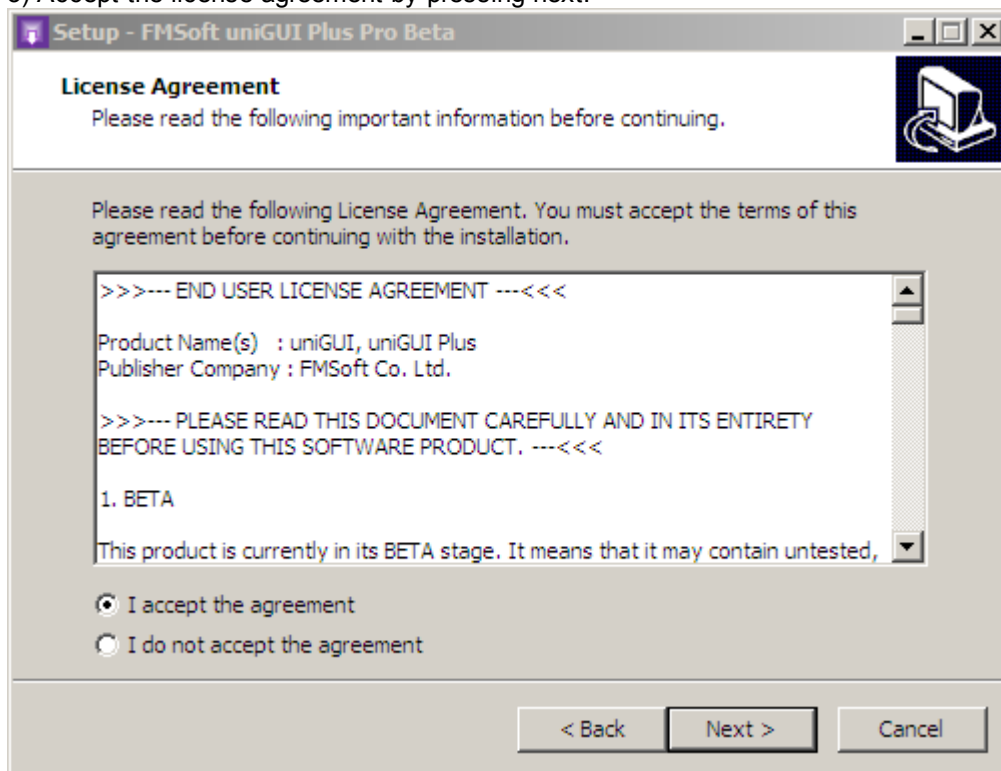
Name:

Email Address:

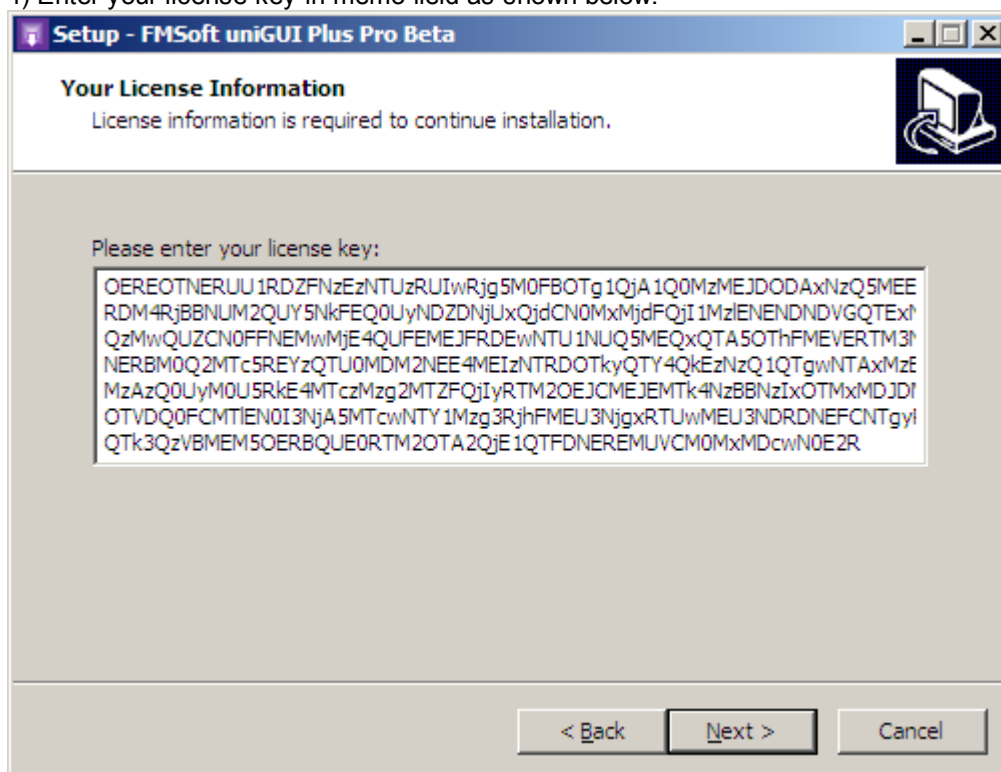
Company:

< Back Next > Cancel

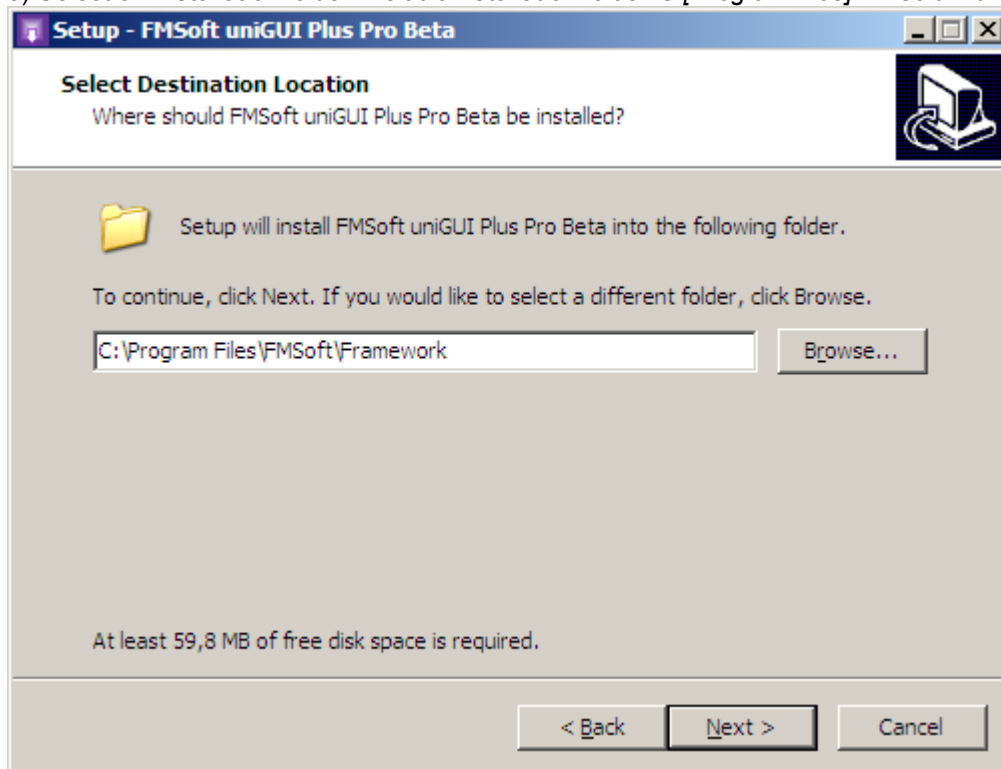
3) Accept the license agreement by pressing next.



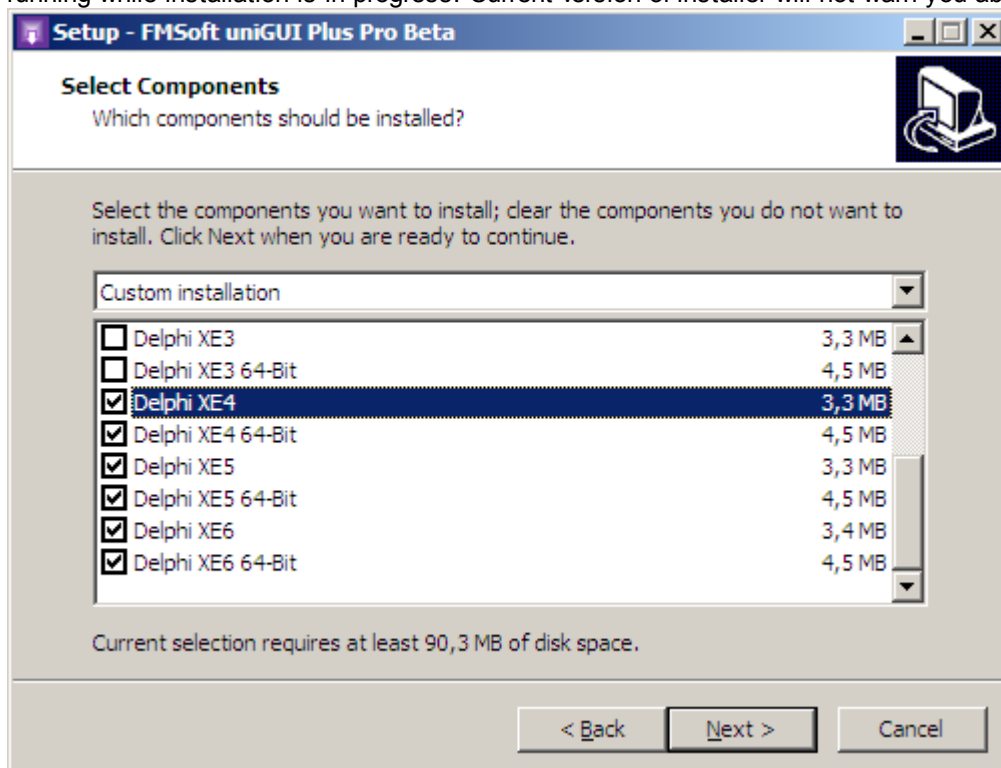
4) Enter your license key in memo field as shown below.



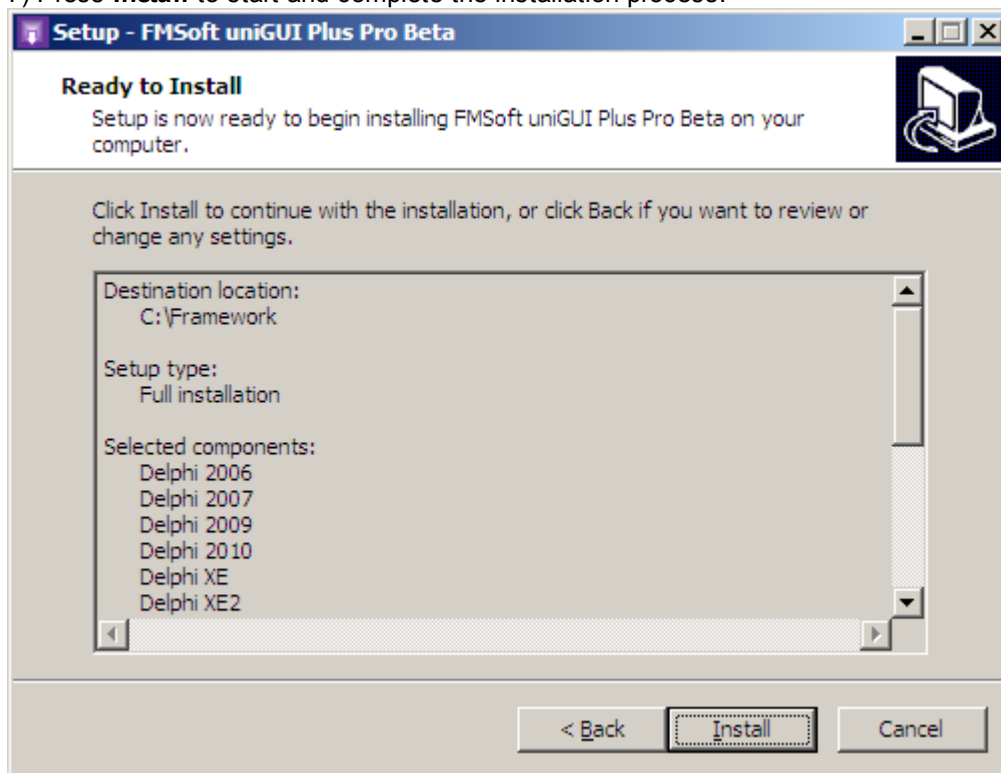
5) Select an installation folder. Default installation folder is *[ProgramFiles]\Fmsoft\Framework*.



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.

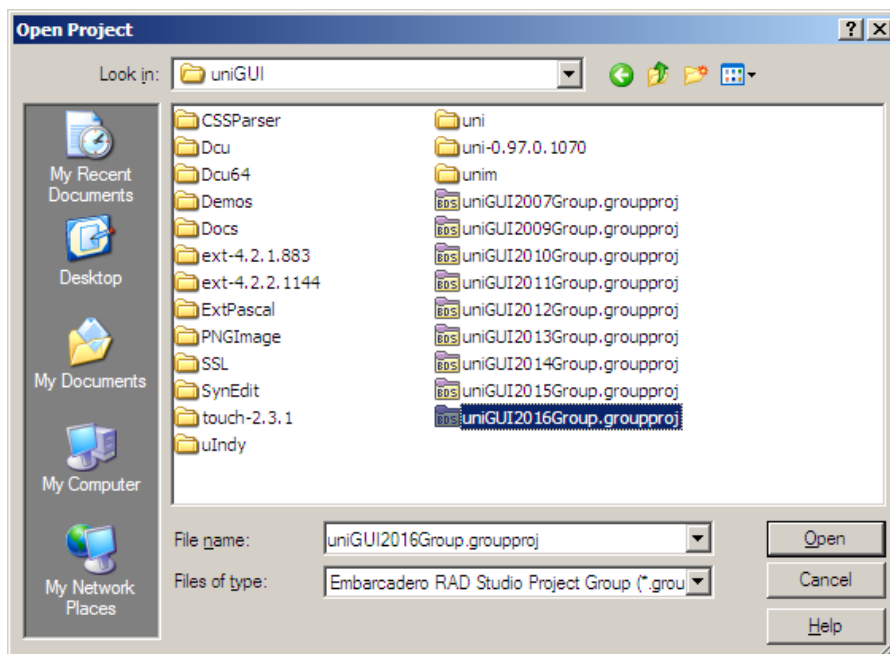


7) Press **Install** to start and complete the installation process.



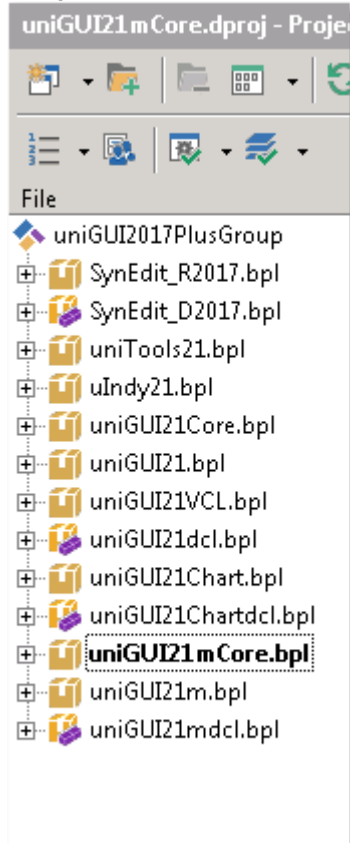
8) Start Delphi and open the project group for your Delphi version. e.g. **uniGUI2016Group (Delphi XE6)**.

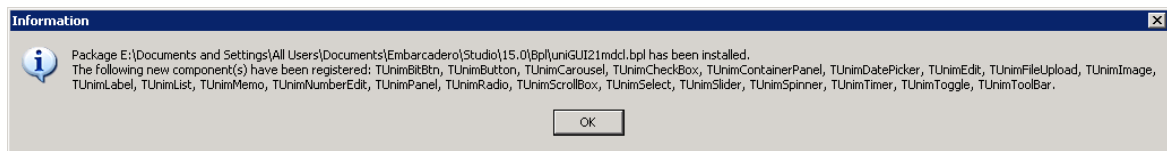
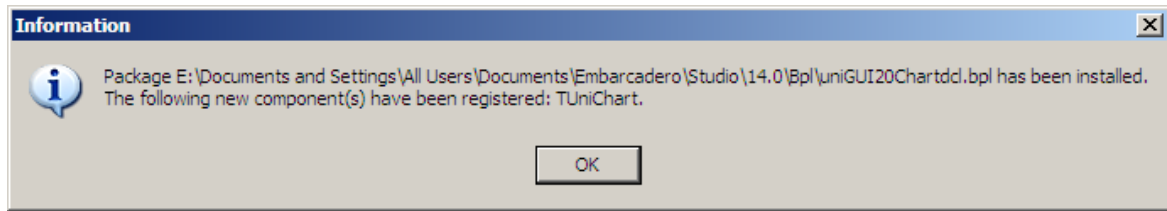
****Additional step for C++ Builder:** Instead of Delphi IDE open project in **RAD Studio IDE**.



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**.





Plus Edition only

****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.

1.3 Sencha Touch Installation

Note: This step is only required for uniGUI Plus Edition.

Current version of commercial uniGUI Plus with [Sencha Touch](#) support does not distribute **Sencha Touch** files.

You must download it from [Sencha Touch web site](#).

Here are the instructions:

Please download **Sencha Touch** from [here](#). What you need here is the folder named **touch-2.4.2** which will be extracted under **touch-2.4.2-commercial** folder.

You must copy **touch-2.4.2** folder to same folder your **Sencha Ext JS** folder resides.

For example if your folder structure looks like:

```
..\uniGUI\Dcu
..\uniGUI\Demos
..\uniGUI\ext-4.2.2.1144
```

You must copy **touch-2.4.2** folder to **..\uniGUI**.

So after copying it should look like:

```
..\uniGUI\Dcu
```

..\uniGUI\Demos
..\uniGUI\ext-4.2.2.1144
..\uniGUI\touch-2.4.2

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 in combination with **uniGUI**.

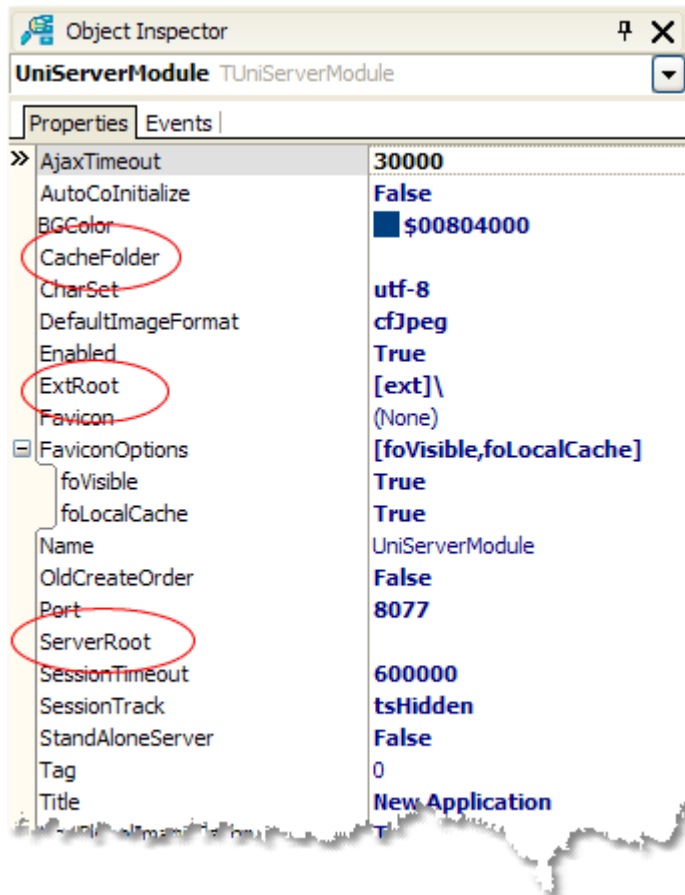
You may not use or distribute included **OEM Sencha Ext JS** for any other purpose other than using it to develop and deploy **uniGUI** projects. This library can not be treated as a standalone library and can't be used to create independent software products based on it.

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.

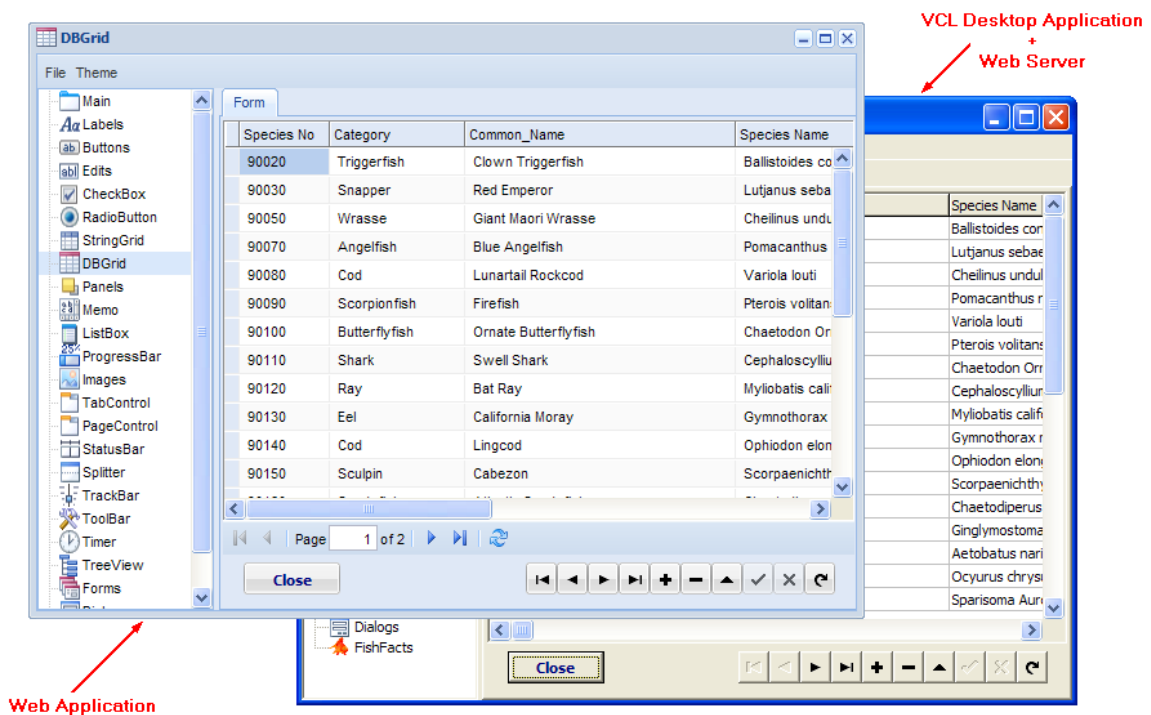
CacheFolder

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 you 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.

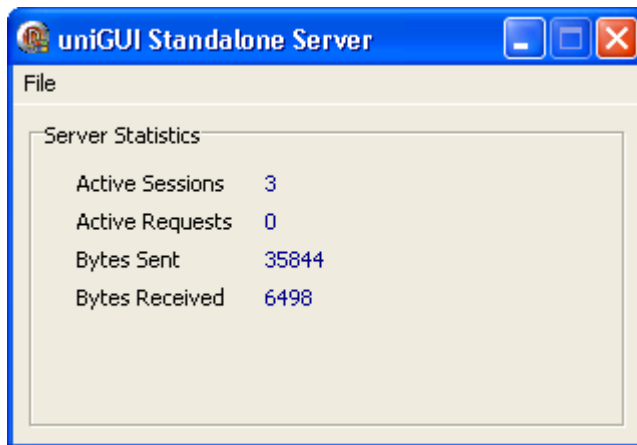


2.1.4 Standalone Server

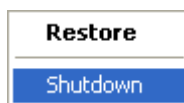
Standalone Server mode is very similar to VCL Application with some differences which makes it a better option for Web deployment. In this mode your application main form is no longer visible on the desktop, instead an icon is placed in Windows taskbar.



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

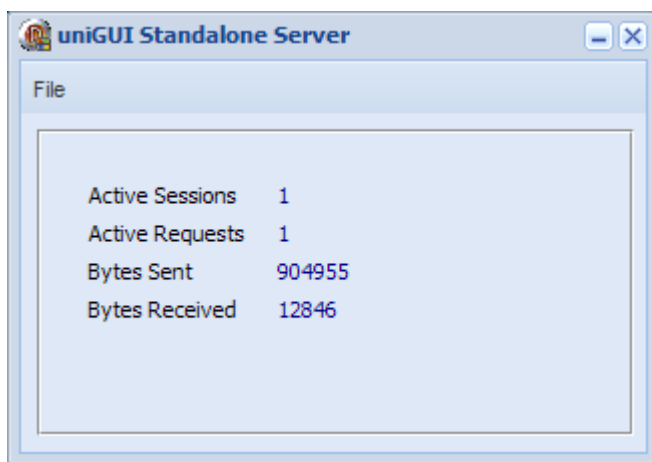


You can shutdown the server by right-clicking the icon and selecting **Shutdown** menu item.



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

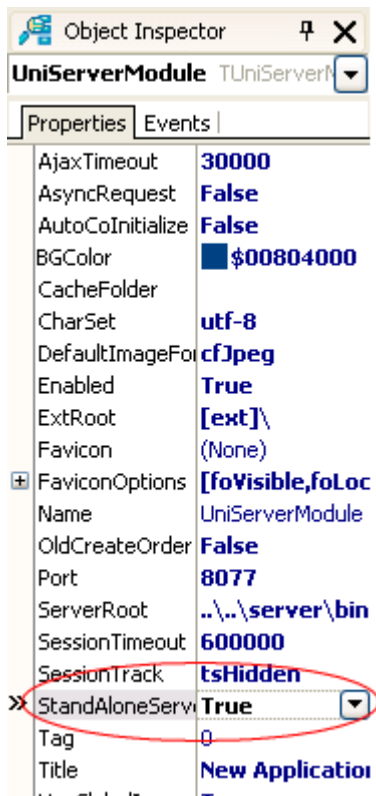
<http://mysite:port/server>



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](#)

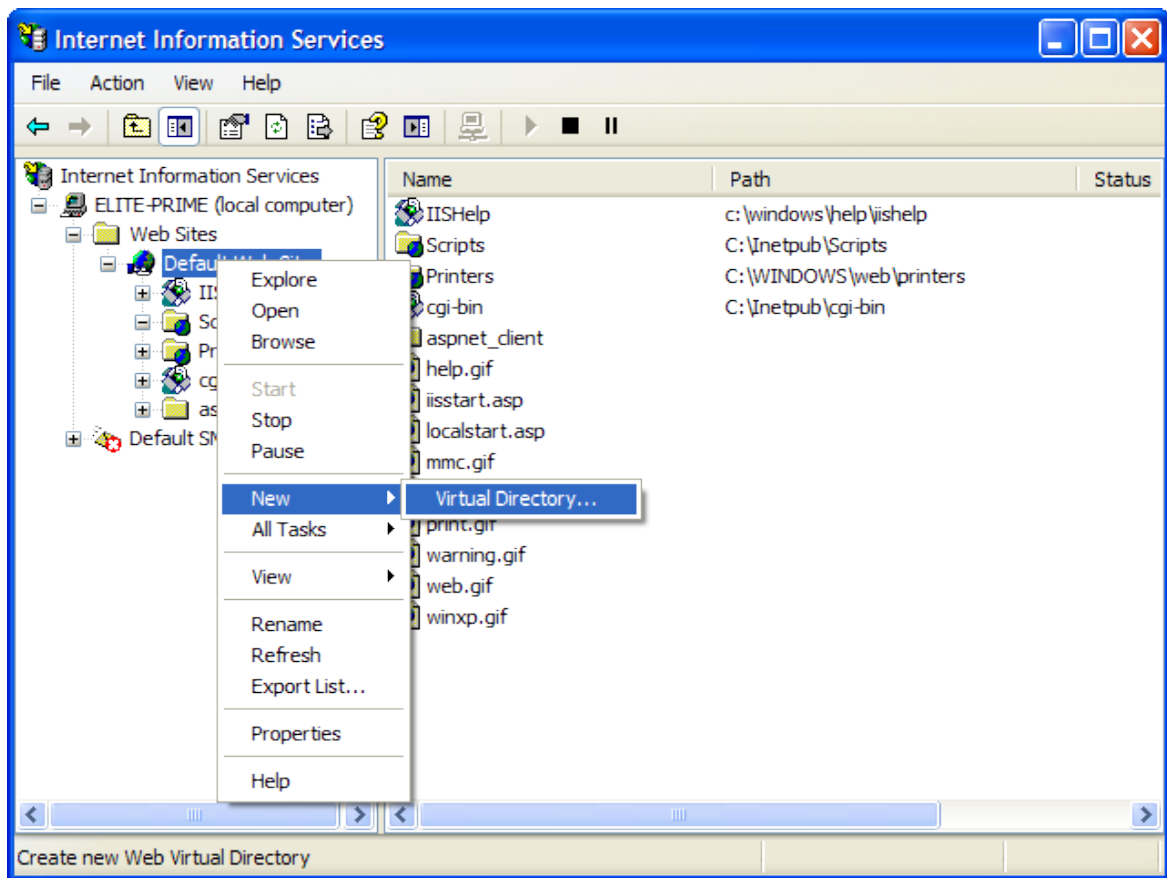
[IIS 6.0](#)

[IIS 7.0](#)

[Apache 2.2](#)

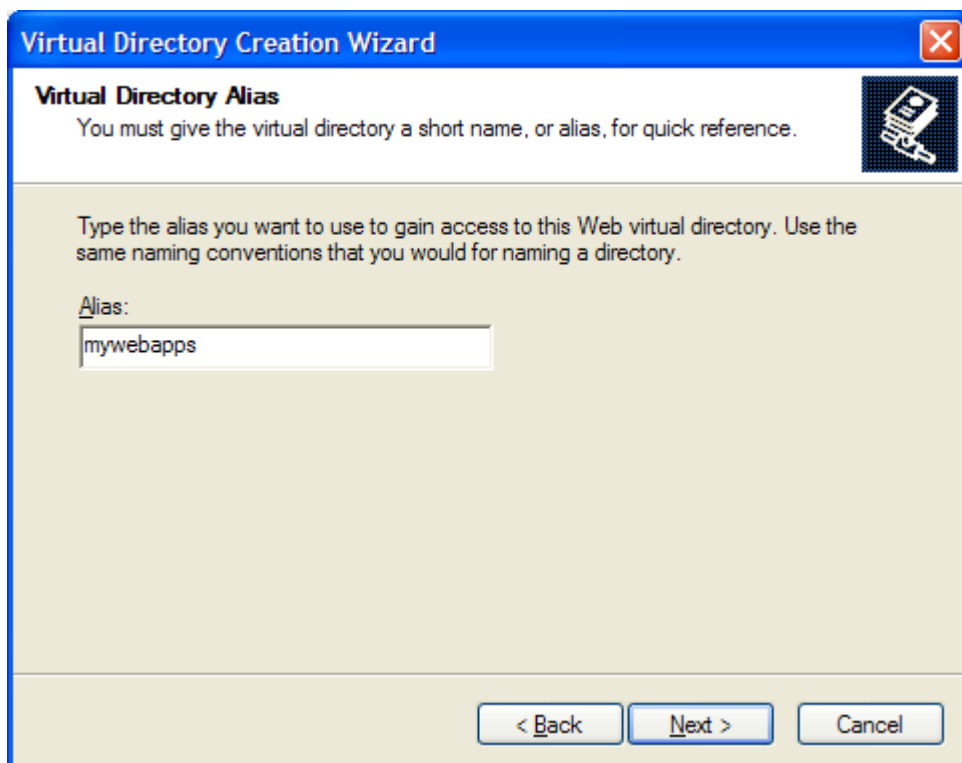
2.1.5.1 IIS 5

First step is to create a new Virtual Directory.

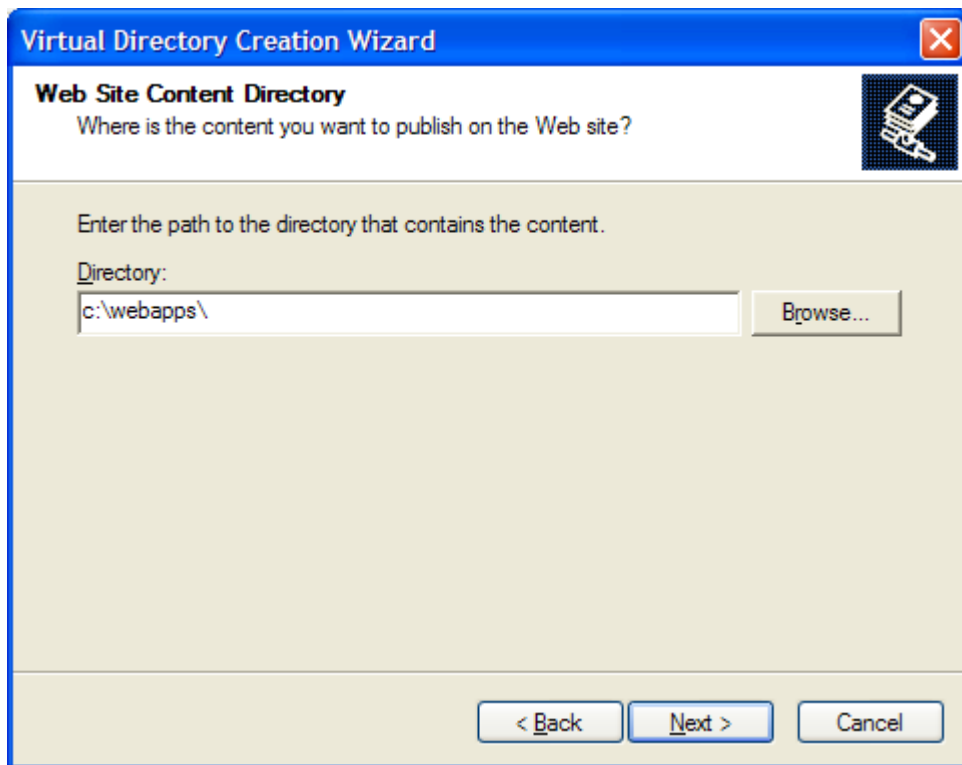




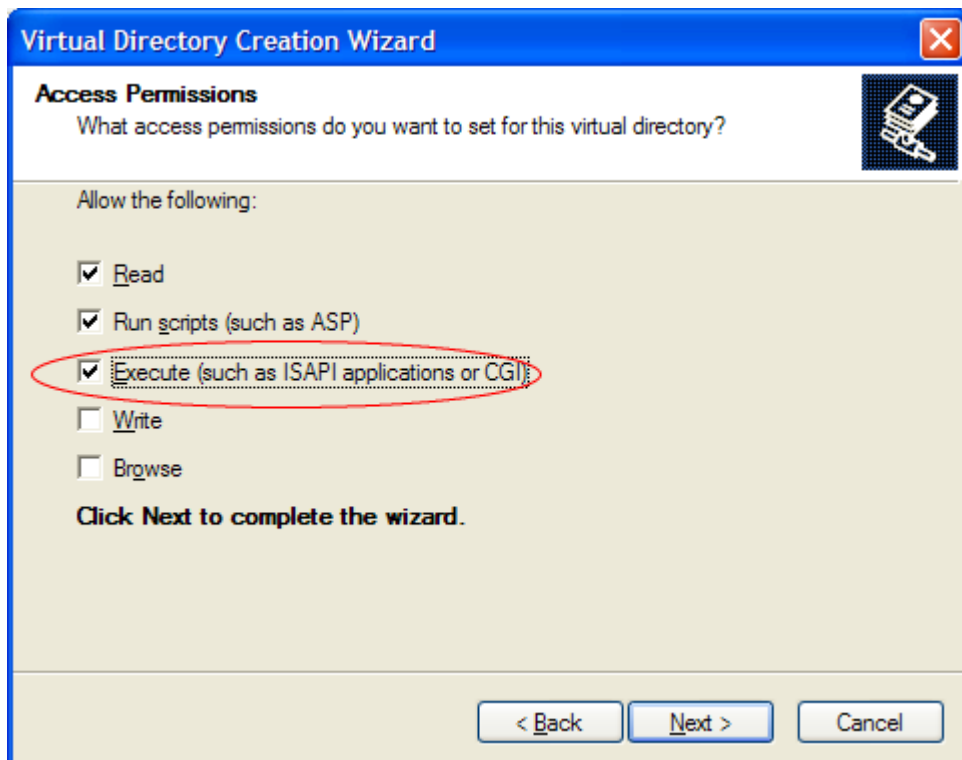
Select an alias for new virtual Directory.

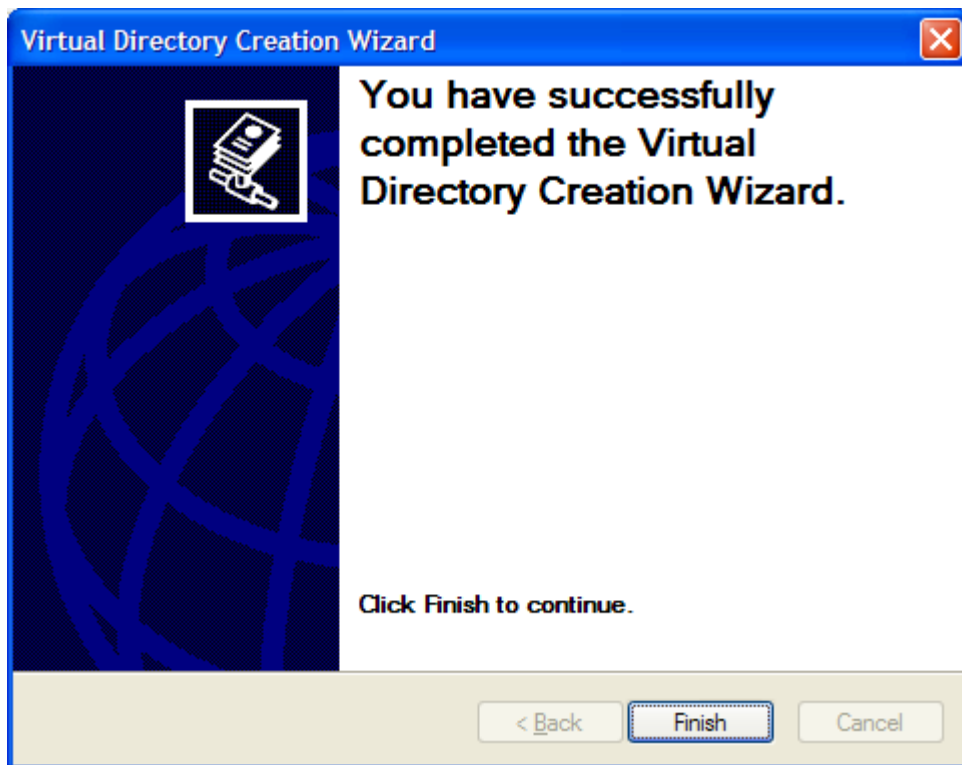


Now assign a folder to newly created alias.



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





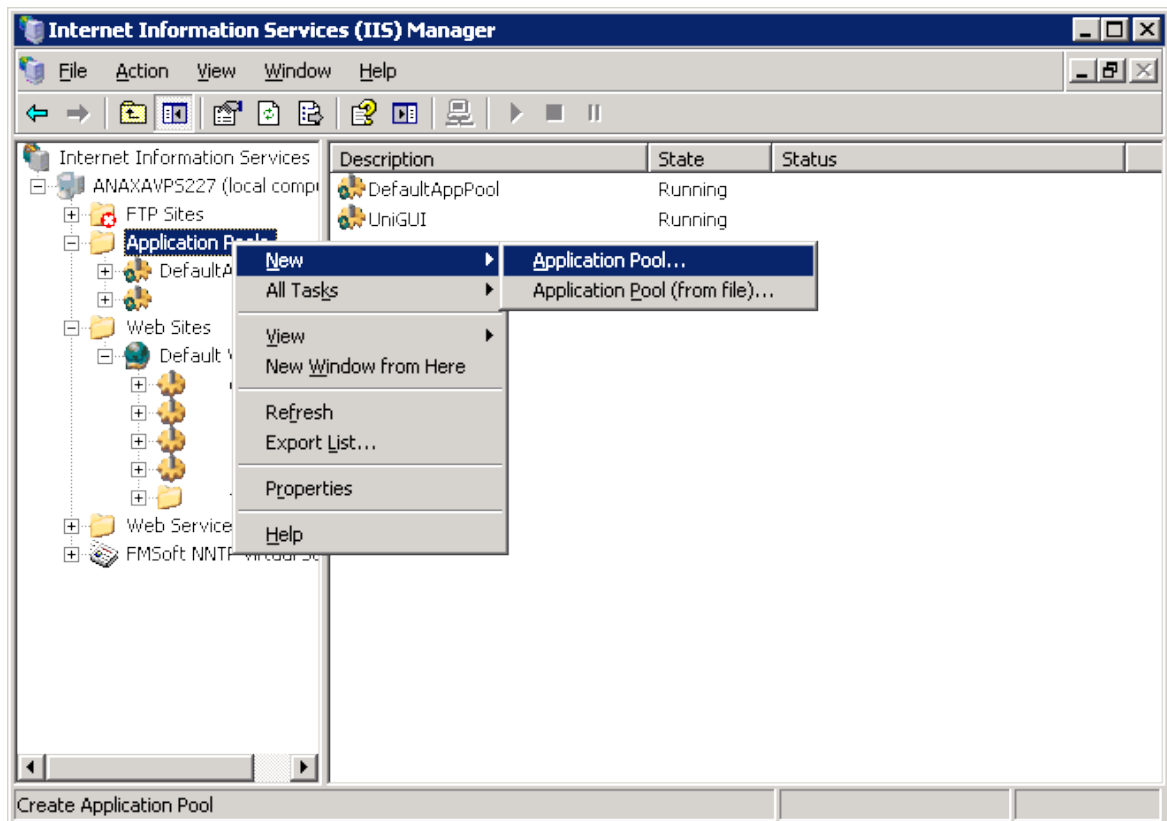
After creating a new virtual directory you are ready to deploy your uniGUI server. Copy your ISAPI module and other required files to virtual directory. You must be certain that **ISS** built-in user **IUSR_<ComputerName>** has enough credentials to access your virtual directory and other folders that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

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

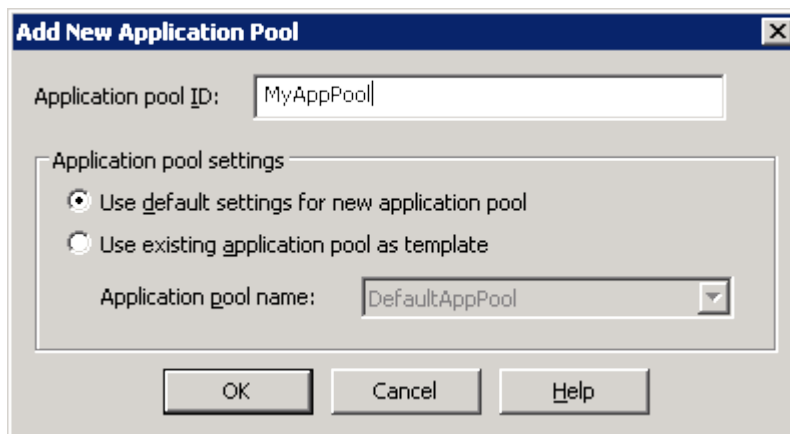
Also refer to [Adjusting Paths](#) for more information.

2.1.5.2 IIS 6

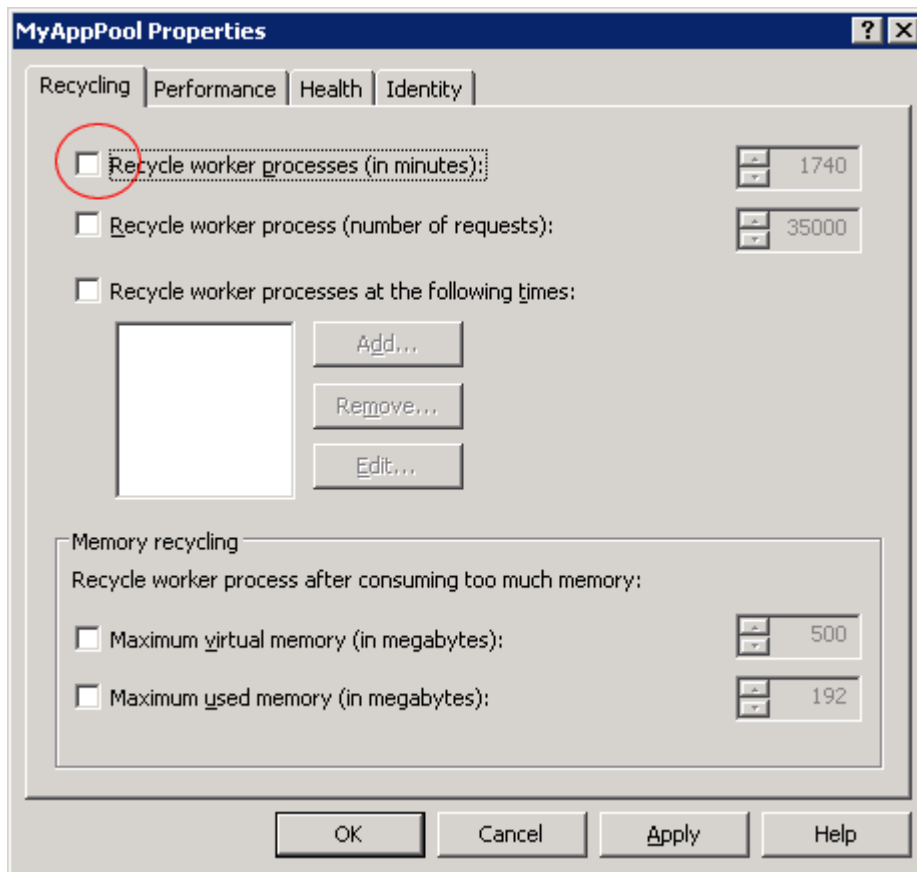
In IIS 6 first step is to create a compatible application pool.



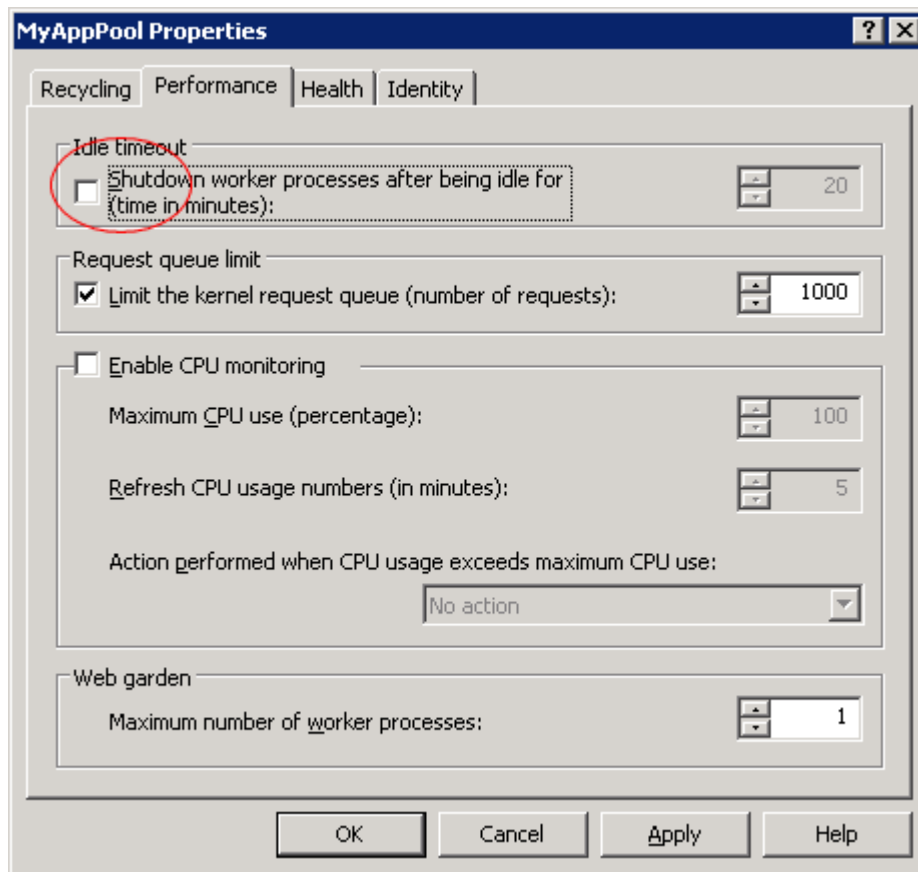
Give a proper name to the new pool.



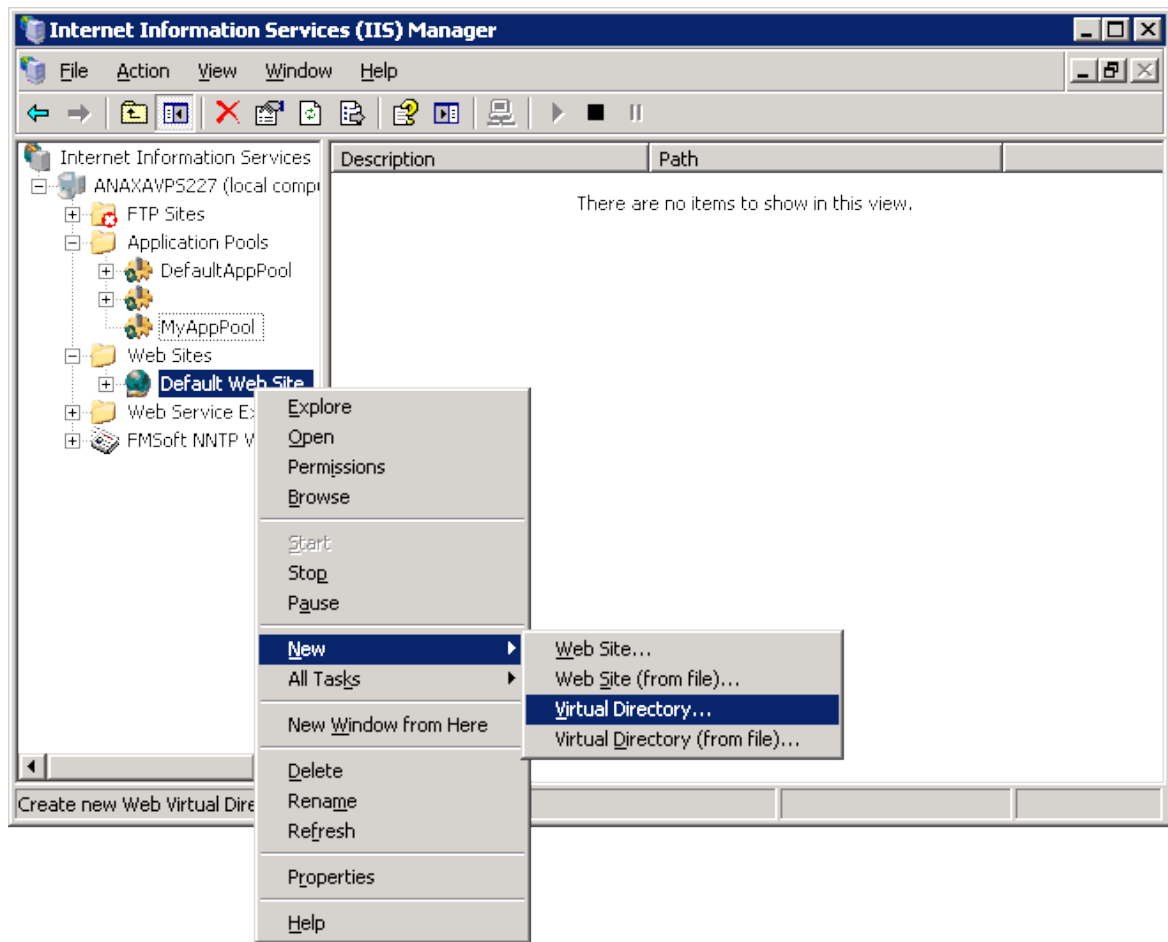
In Application Pool properties, Recycling Tab uncheck the **Recycle worker processes** option.



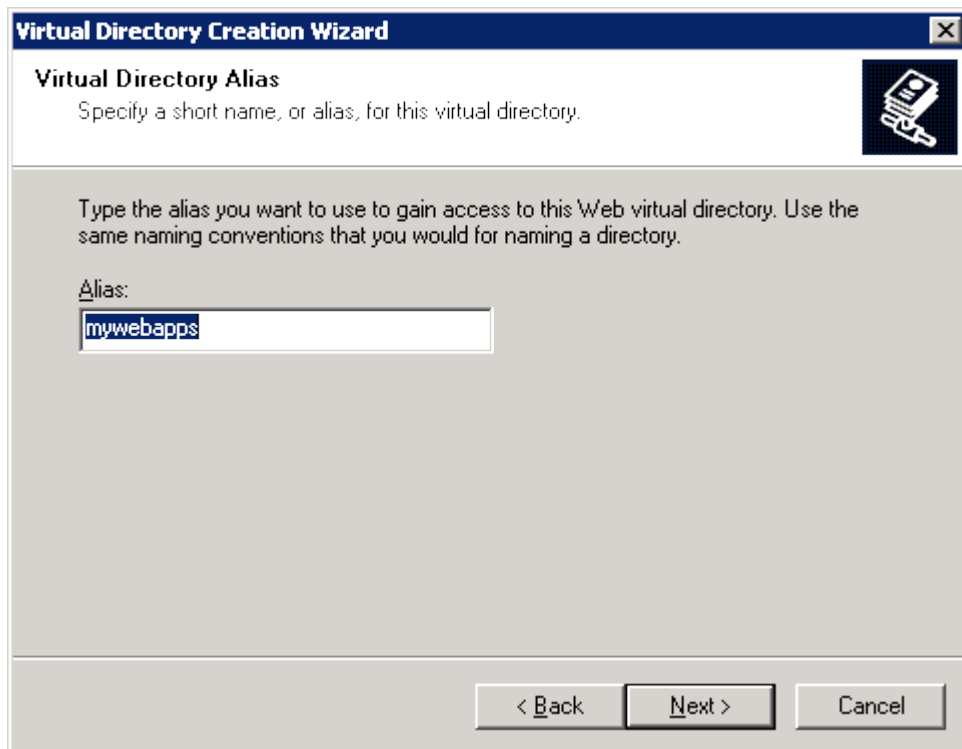
In Performance Tab uncheck the **Shutdown worker processes after being idle...** option.



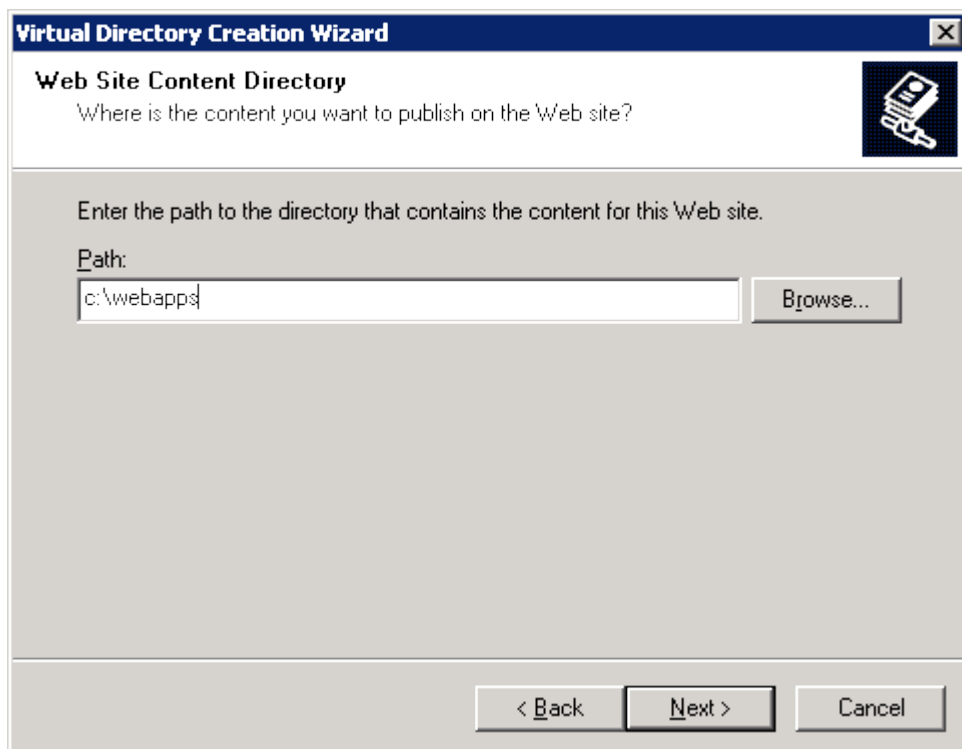
Next step is to create a Virtual Directory.



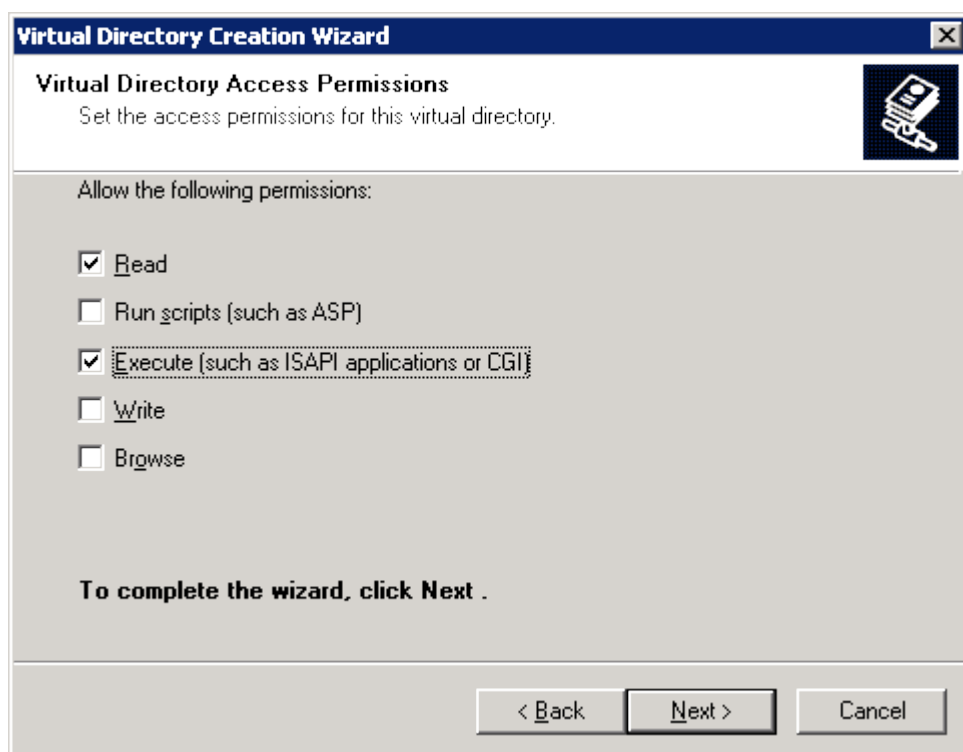
Assign an alias to new virtual directory.



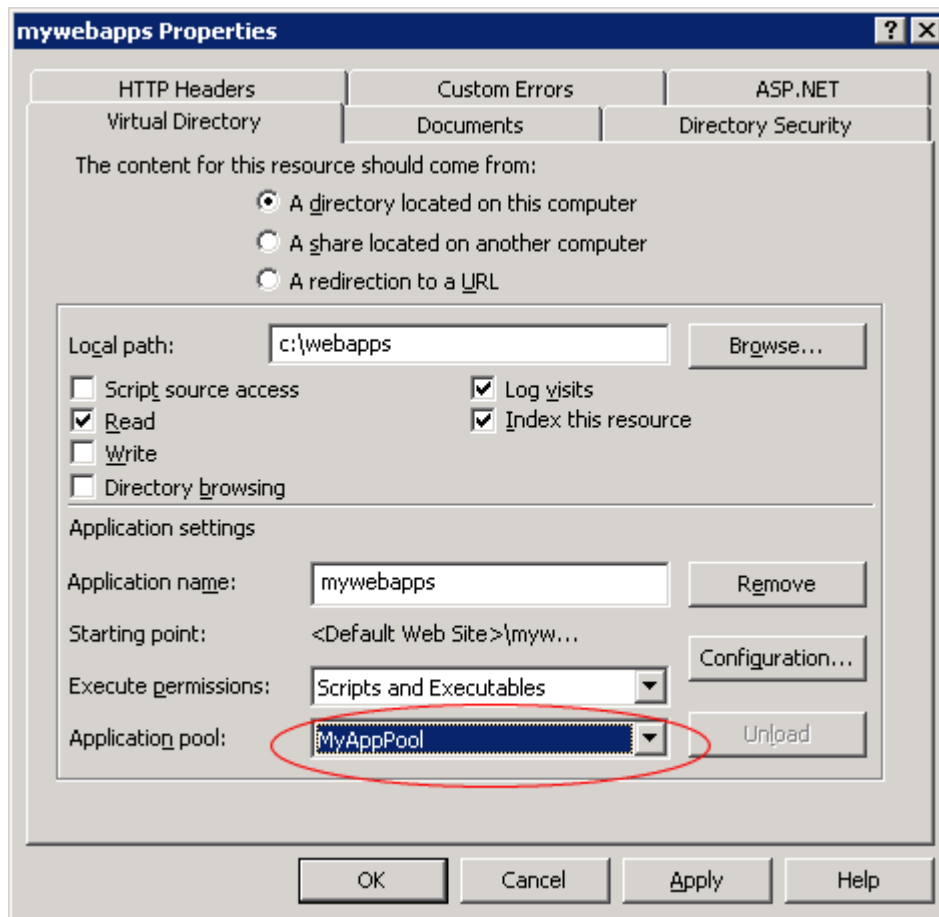
Select the folder where your ISAPI modules are located.



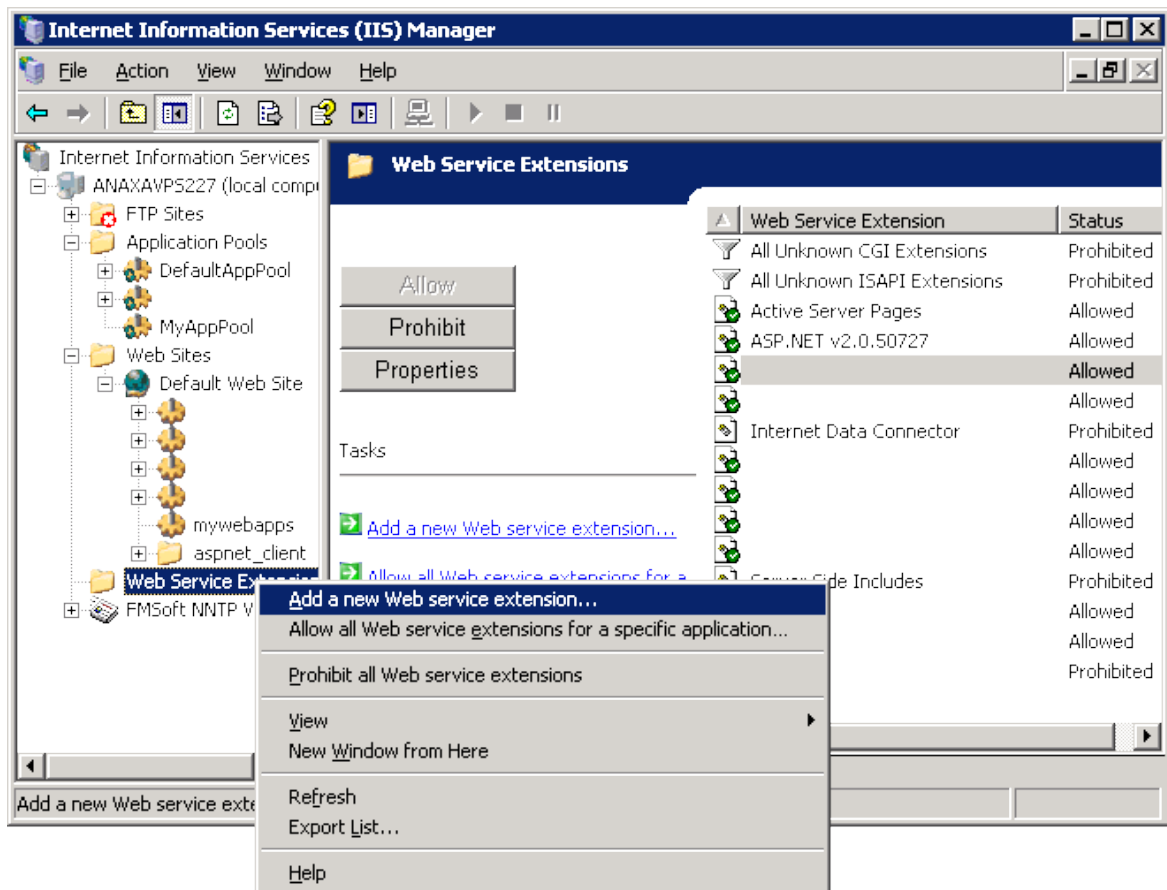
Be sure to grant **Execute** permission to Virtual Directory.



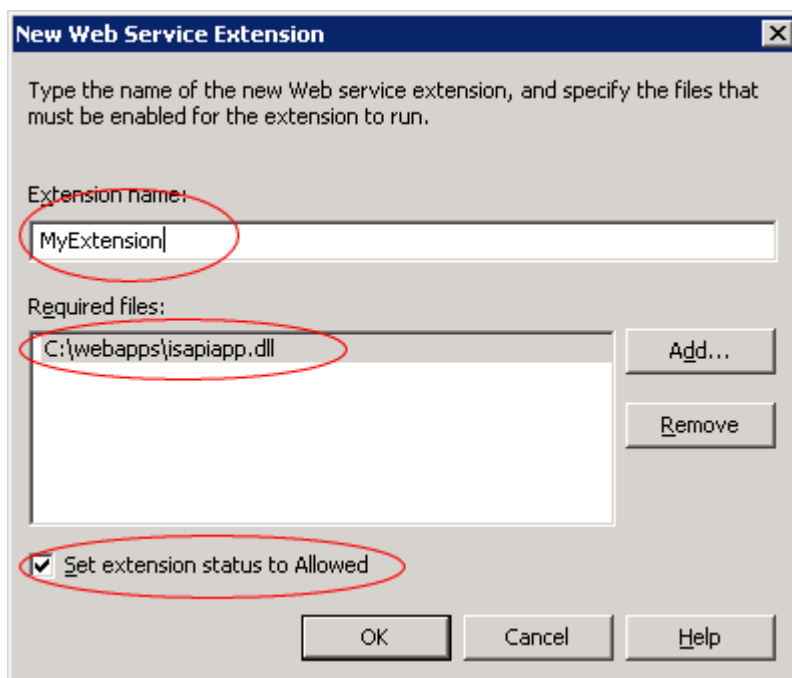
Now open the properties of newly created Virtual Directory and change the default pool to pool you created in first step.



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



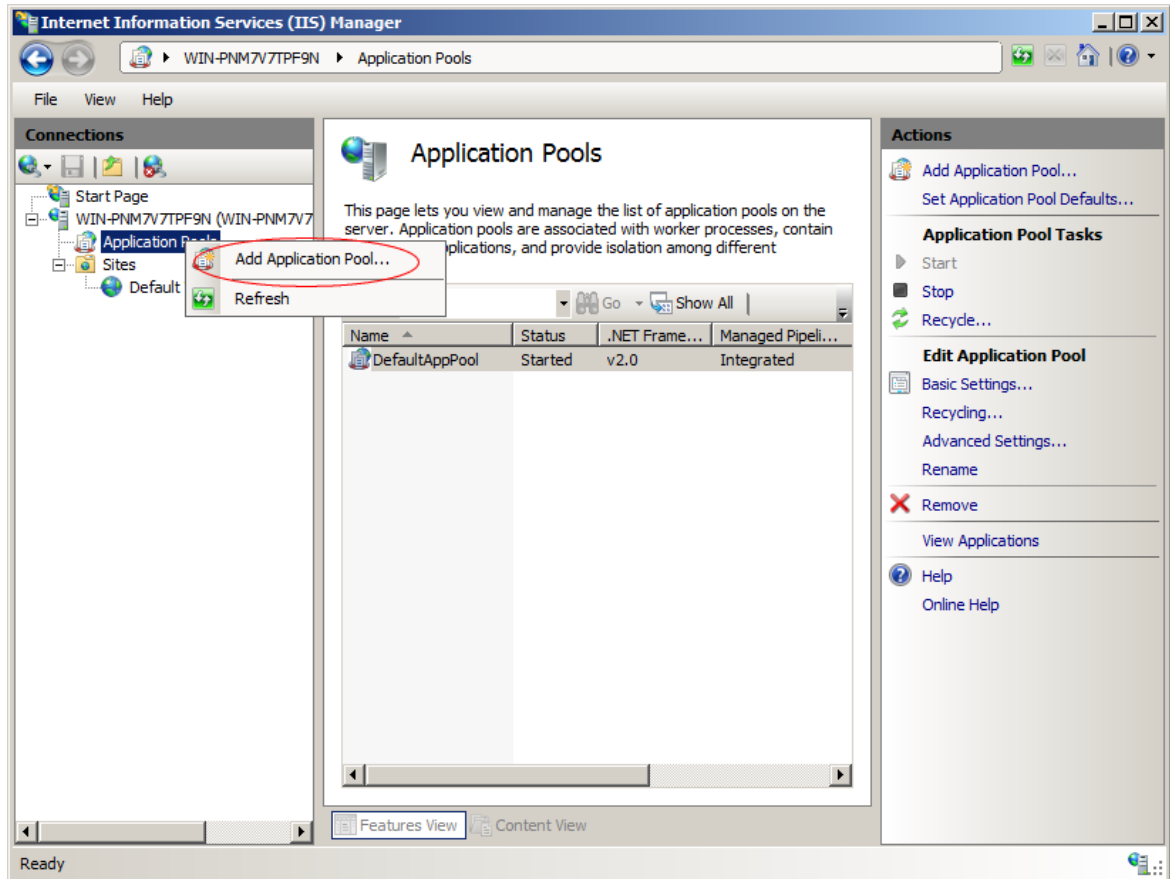
Assign a name to your extension, add the extension's module DLL file and check the **Set extension status to Allowed** option.



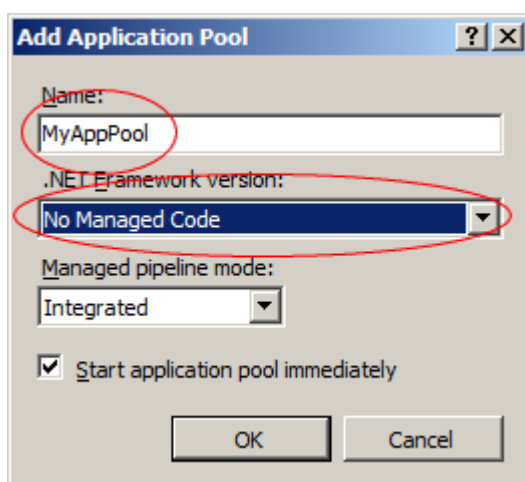
Also refer to [Adjusting Paths](#) for more information.

2.1.5.3 IIS 7

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



Assigne 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**

Advanced Settings

(General)

.NET Framework Version	No Managed Code
Enable 32-Bit Applications	True
Managed Pipeline Mode	Integrated
Name	MyAppPool
Queue Length	1000
Start Automatically	True

CPU

Process Model

Identity	ApplicationPoolIdentity
Idle Time-out (minutes)	20
Load User Profile	False
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90

Process Orphaning

Rapid-Fail Protection

Recycling

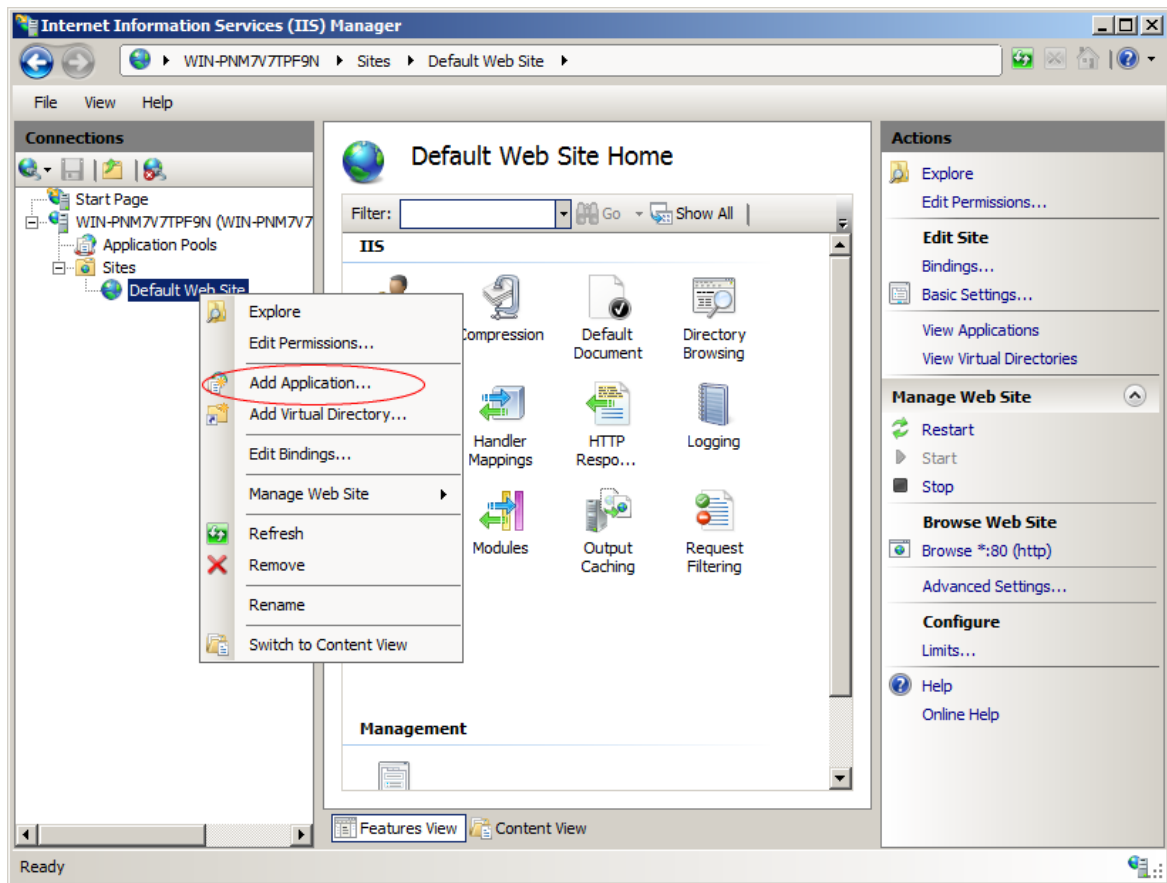
Disable Overlapped Recycle	True
Disable Recycling for Configuration Changes	False
Generate Recycle Event Log Entry	
Private Memory Limit (KB)	0
Regular Time Interval (minutes)	0
Request Limit	0
Specific Times	TimeSpan[] Array
Virtual Memory Limit (KB)	0

Regular Time Interval (minutes)

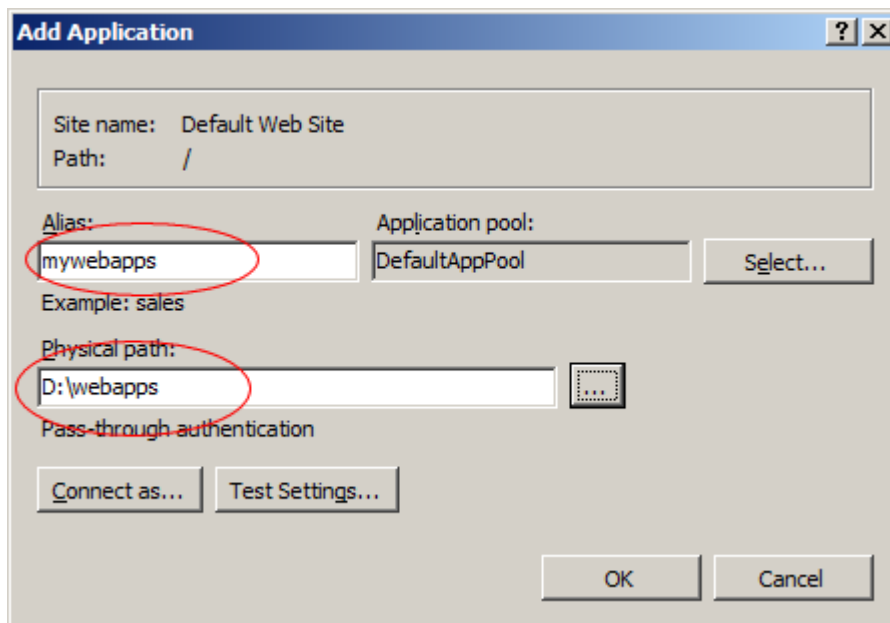
[time] Period of time (in minutes) after which an application pool will recycle. A value of 0 means the application pool does not recycle on a regular interval.

OK Cancel

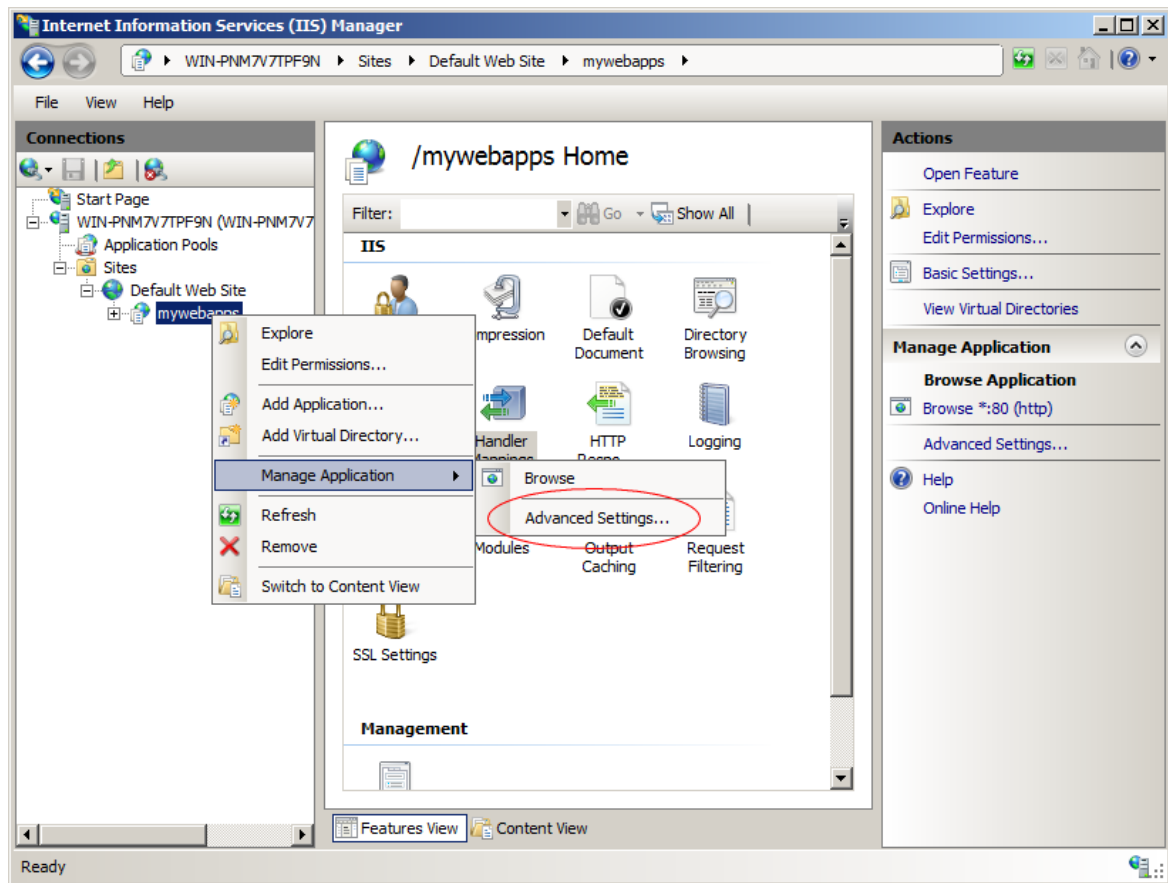
Now add a new Application.



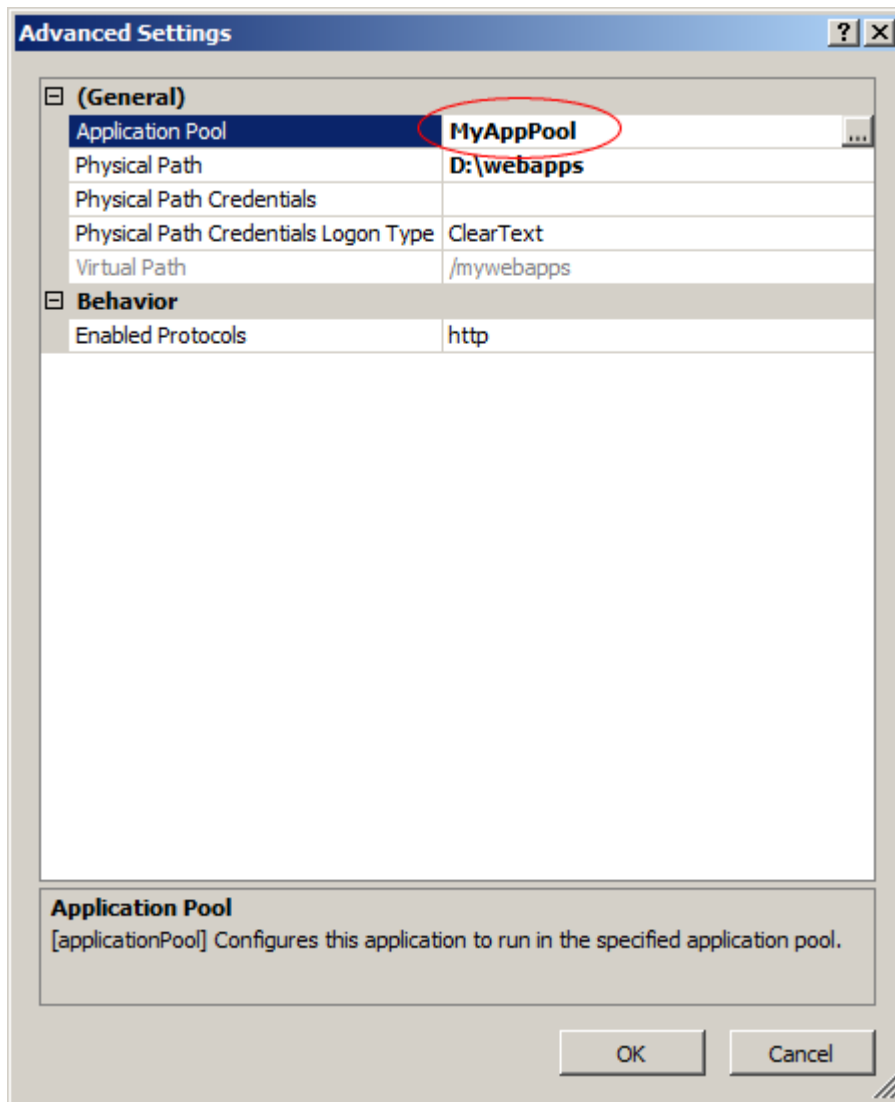
Give it a proper name and adjust the **Physical path**. It is the path where your module DLL files exist.



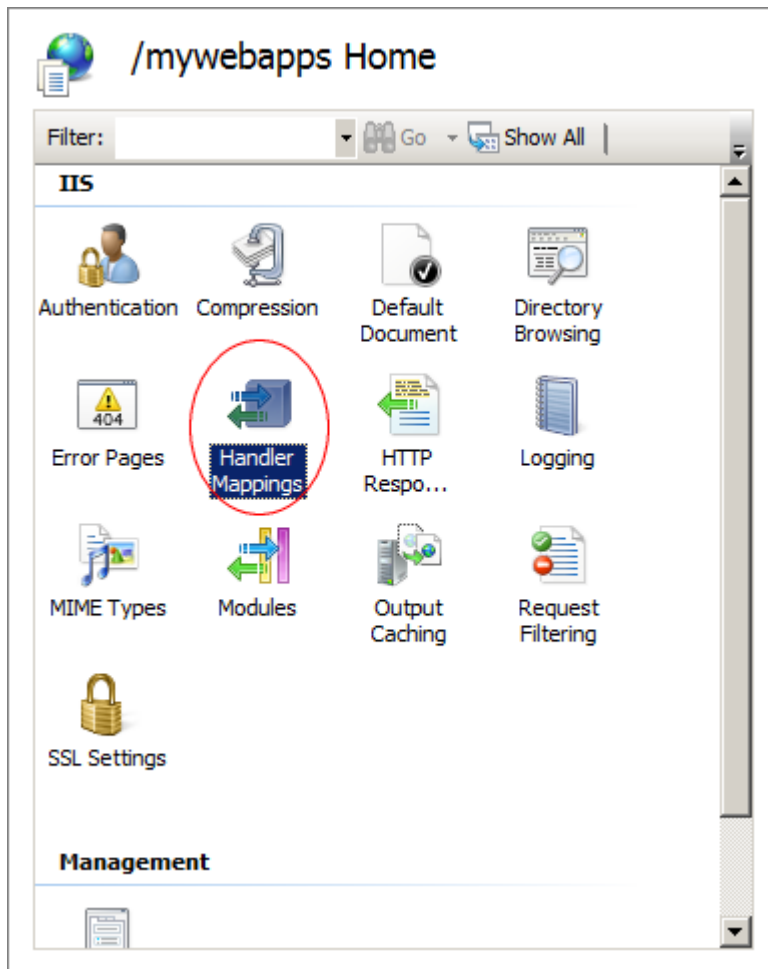
Select Advanced Settings menu and open Advanced Settings dialog.



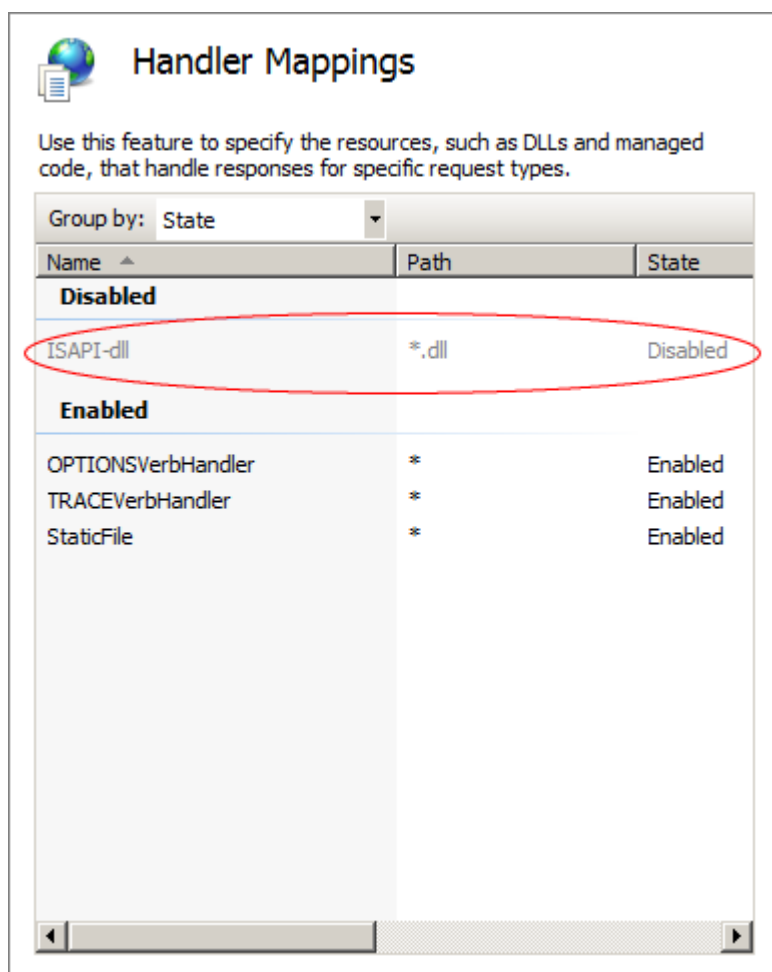
In Advance Settings screen set the **Application Pool** to one you created in first step.



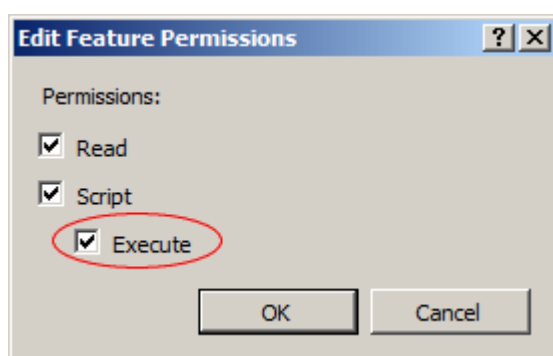
Next step is to adjust the handler mapping for the application you just created.



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.

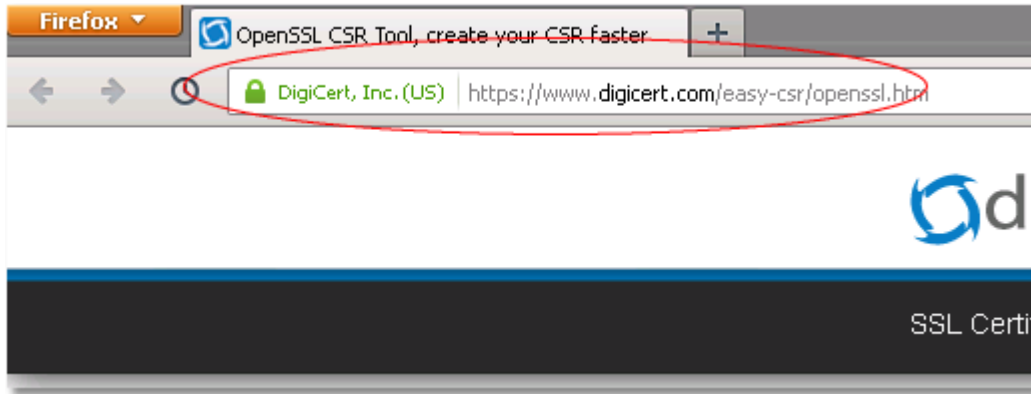
Below sample shows contents of a **pem** file:

```
-----BEGIN CERTIFICATE-----
MIIB8jCCAV+gAwIBAgIQfjGd2Py0qZJGqdkPiRlDdjAJBgUrDgMCHQUAMBAxDjAM
BgNVBAMTBWVsaXRlMCAXDTEzMDYwMjE3NTA0OfoYDzIxMTMwNTA5MTc1MDQ4WjAQ
MQ4wDAYDVQQDEwVlbG10ZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAt3pi
pMYHNzUueLZBb1eMrPop6Emta/KLyLaK94v1M1lV/6ITiuFtuSs9gq0s516s2th7
FUKBpgfQvrb+3b9h10WMca8MTbYrLGL+dHqRk4jGt/8GAUeYHkKddk/NeXkZWqaD
3aMdURpTgE2iK/d86C1YdsxqXTxP+Uax/eN4RUECAwEAAaNTMFewFQYDVR01BA4w
DAYKKwYBBAGCNwoDBDAkBqNVHREEJjAkoCIGCisGAQQBgjcUAgoGFAwSZWxpdGVA
RUxJVEUtUFJTTUAMakGA1UdEwQCAAwCQYFKw4DAh0FAAOBgQCDSHm54tMh1sPY
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
```

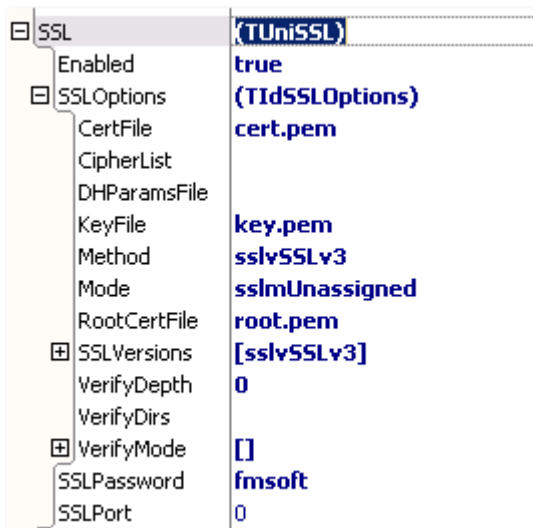
These **DLL** files are distributed as a part of **OpenSSL** installation. You can also find the most recent version of **OpenSSL DLL** files under below folders:

```
[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.

2.2 Stress Test Tool

2.2.1 Introduction

Scalability is one of the major concerns when developing web applications. Each web application must be able to serve multiple users. Number of users can vary depending on type of application. An application designed for a small intranet may be targeted by 10-50 concurrent users at most while an internet web application may be used by more than 500-1000 simultaneous users.

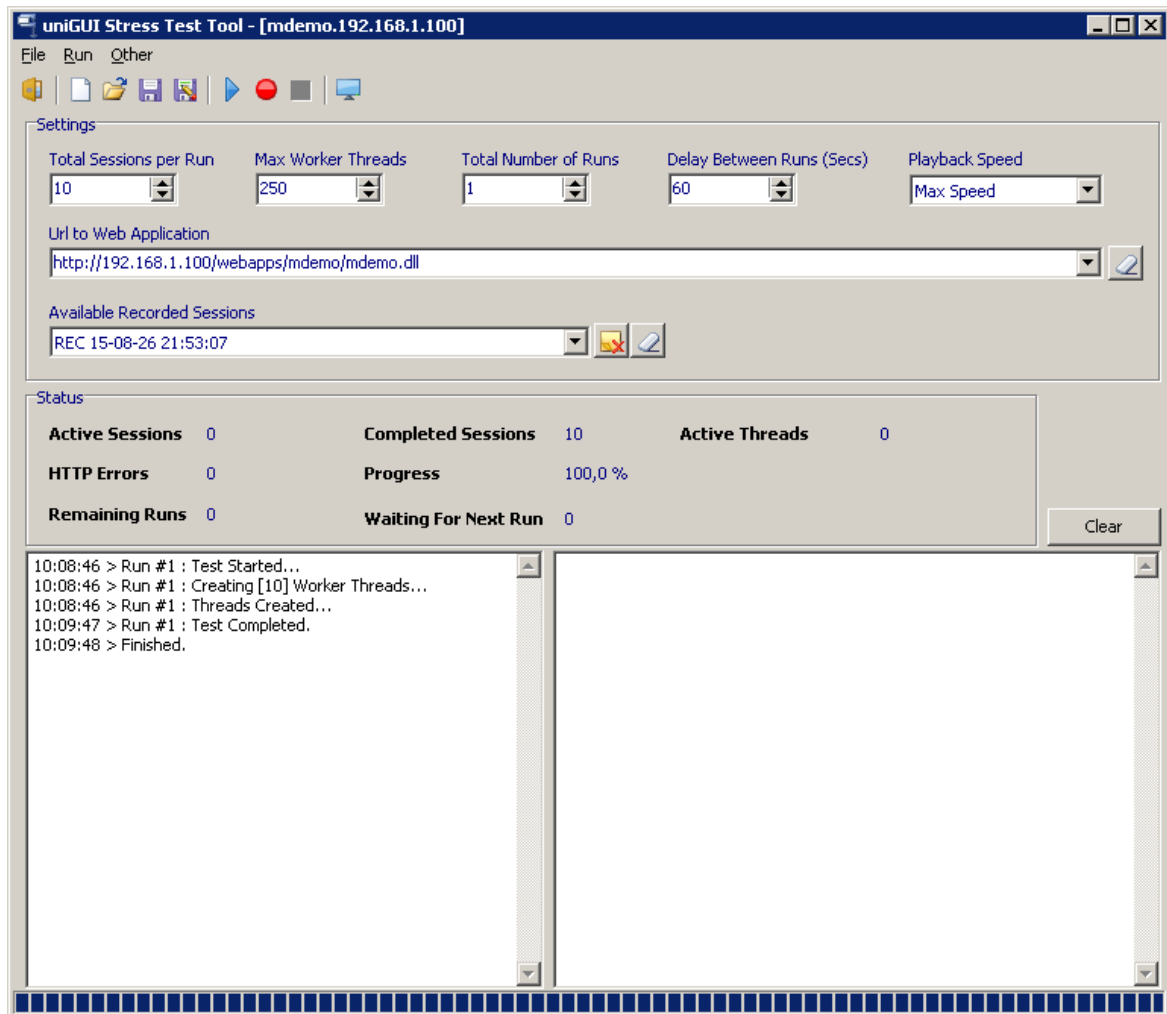
In order to make sure your web application is ready for deployment in multi-user production environment several tests must be performed. Firstly, application must be analyzed for its resource usage. For a desktop app where only a single instance of your app is running resource usage will not be your main concern because your application can use all of resources available in system. On the contrary, a web application server is designed to serve multiple users and each user can be dedicated only a fraction of available resources.

Scalability simply means ability of your project to scale up when number of sessions increases in a real world system. A scalable web app should use several techniques to manage and use system resources in most efficient way possible. For example, if each of your sessions consumes ~1000 KB of memory (~1MB) in this case a web app running 400 sessions will require at least 400 MB of physical/virtual memory to run properly. However, if each session consumes more than 10MB then it is easy for your app to run out of memory when there are more than 100 active sessions. Other system resources such as disk space, database connections and etc. should also be taken into account.

For developers it may not be an easy task to simulate production environment on his/her desktop. Your app maybe running smoothly when there are only a few sessions, but when number sessions start to pass above a certain threshold several resource related issues and scalability problems may arise. To deal with such scalability issues **uniGUI** is equipped with a very useful named **Stress Test Tool** which aims to simulate a multi-user environment right on your desktop. **Stress Test Tool** is located under `..\uniGui\Utils\StressTestTool` folder and it is deployed with full source code so it can be customized by developers if needed. This tool introduces many advanced features which enables developers to simulate a production environment under heavy load.

2.2.2 Usage

Here is the main form of stress test tool:




Using this tool you can record a web session and then playback to simulate multi-user behavior in production environment. Stress test tool stores each test environment in a project file which can be used later to repeat the test. A project saves all of your recorded sessions and test parameters. It is good to create a separate Stress Test project for each uniGUI project. uniGUI Stress projects are stored under **Projects** folder with an extension of (*.uprj)

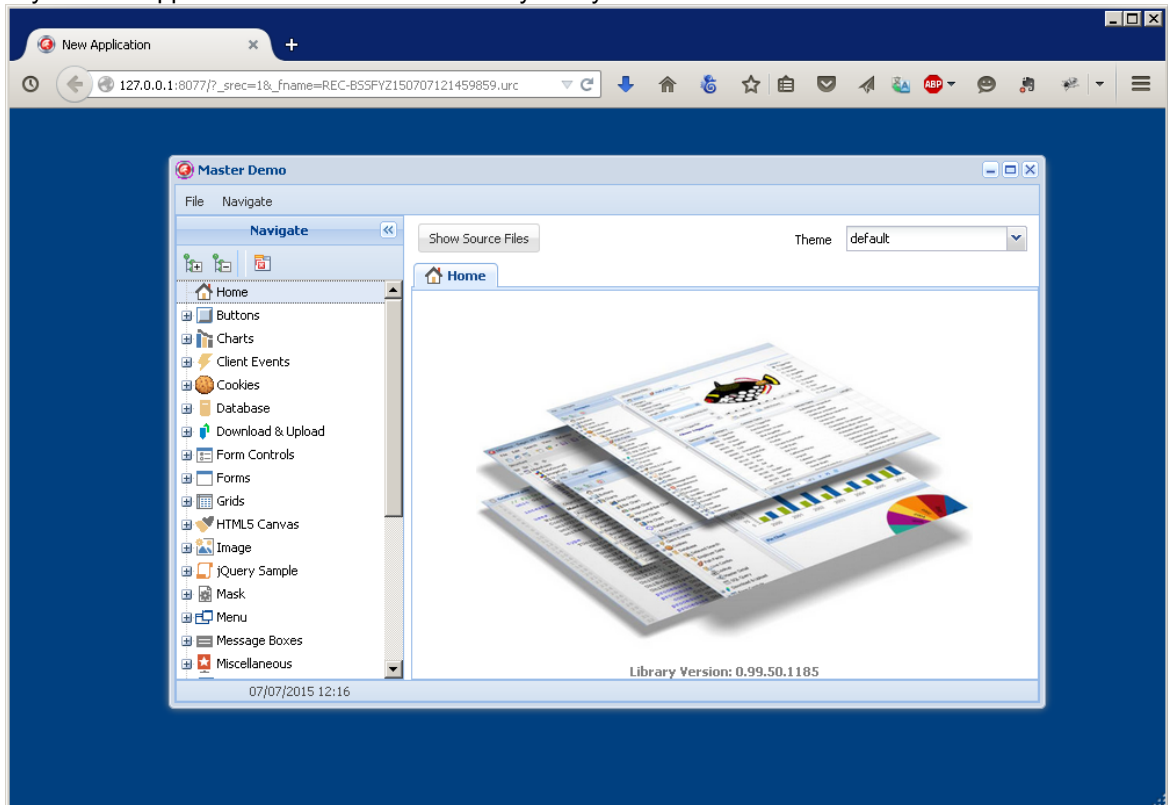
2.2.2.1 Recording a Session

Recording a session is an easy task.

- First of all, you must compile your server with **ServerModule->Options->soAllowSessionRecording = True**. This parameter must be set to False once your app is ready for production.
- Build and run your uniGUI server.
- In **Stress Test Tool** set **Url to Web Application** to Url address of your server. If your server is

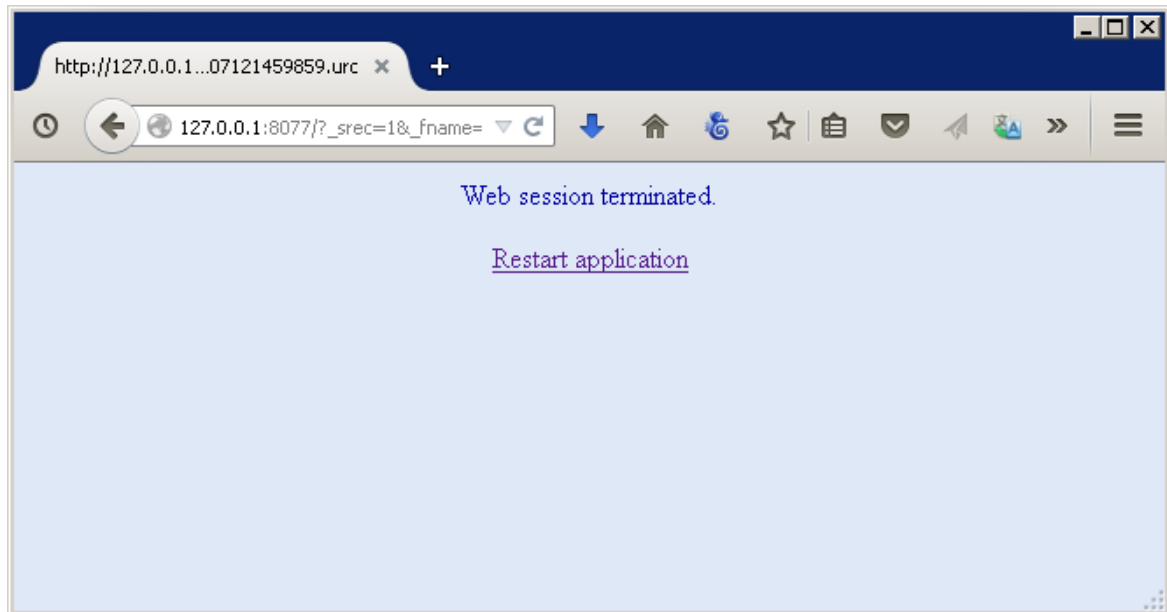
running in same machine a typical address would be <http://127.0.0.1:8077>



- Press the Record  button. This will launch a new browser window which will show main form of your web application. This action will use your system default web browser.



Session Recoding

- At this stage you can start using web application by navigating in menus, forms, grids and etc. All of your actions will be recorded.
- You can continue recording this session until you think all required menus, forms, grids and functions of your app are fully visited.
- When you are done you can terminate your web application.



- Now switch to Stress Test Tool and press the Stop button .
- This finish your record session and adds it to list of available records for this project.
- You can save your project using Save button .


Important: A session must be recorded in a way that session output is always predictable i.e. it will always produce same HTML content. For example, pressing **Button1** should always open **Form1** and selecting menu item **User1** should always show a frame named **TUserFrame**. Normally, it is the case for 99% of web applications where each user action results same output. There are certain cases where user action may not produce a predictable result. Inserting rows in a DBGrid is an example for this case. When user inserts a new row it creates a new grid row and also a new grid page after certain number of rows are inserted. This means that each inserts produces a different JavaScript content which is not predictable and it totally depends on current number of rows in the attached database table. For this, it is recommended to avoid inserting data into a DBGrid while recording a session. Likewise deleting or editing a row may not run properly during session playback. Consider that your table contains 100 rows and you delete a single row while recording. When you playback 100 instances of same session will try to delete same row while only one session can actually delete it.

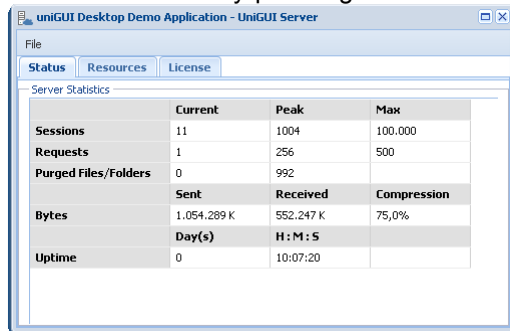
2.2.2.2 Running the Stress Test


Running a stress test by playing a previously recorded session is an advanced feature which will help you to simulate a real time multi-user environment. After you have one or more recorded sessions you can proceed with actual stress test procedure.

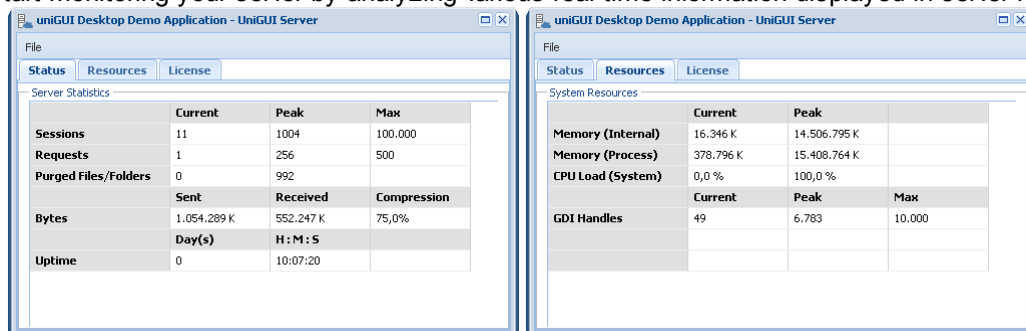
Stress test tool will play back a session by sending sequences of recorded Ajax requests to server. This will simulate an environment where many users are using the web application. It is important to know that play back process does not run your web application in a real browser window. The simulation is performed by sending previously recorded Ajax packets to server in the exact order generated while recording a session.

To run a stress test you must follow below steps:

- **Make sure that you don't run the Stress Test Tool in IDE and in debug mode. Running in debug mode will highly reduce speed and performance of the Stress Test Tool. It is better to compile it and run the executable directly.**
- **To record and playback a mobile session you must explicitly add '/m' to end of the application Url.**
- Load a saved project or record a new a session if there are no recorded session available. (see previous section to see how sessions can be recorded).
- Adjust various test parameters:
 - **Total sessions per run:** This is the number of sessions which will be created in each run. It is proper to start with a low value and increase it gradually. This value should be a value lower or equal to server's *ServerModule->ServerLimits->MaxSessions* property.
 - **Max Worker Threads:** Number of worker threads used to communicate with the server. This should be a value lower or equal to server's *ServerModule->ServerLimits->MaxRequests* property.
 - **Total number of runs:** Total number of times that recorded session will be played.
 - **Delay between runs:** Number of seconds to wait before a new run starts
 - **Playback Speed:** Using this parameter you can adjust the speed at which session is simulated. If you choose Max Speed no delay will be inserted between Ajax calls so your session will be played at max speed. You can also choose Real Time which will try to simulate playback close to the original timing of the recorded session.
- Start server monitor by pressing  button.



- Now you can press Start button  which to start the actual stress test.
- Start monitoring your server by analyzing various real time information displayed in server monitor:



- **Sessions:** It shows number of sessions that are running in your uniGUI server. Peak value shows maximum number of sessions that were running during server lifetime.
- **Requests:** Number of pending Ajax requests which are currently being processed by the server.
- **Purged Files/Folders:** These are files or folders which no longer belong to a session, but not

erased yet. Framework will gradually delete purged files and folders in a background thread. Under normal conditions this value should eventually become zero.

- **Bytes:** Number of bytes Sent/Received during server lifetime.
- **Uptime:** Time passed since server has been started.
- **Memory:** It is one of the vital information which you need to constantly keep an eye on while performing the stress test. It will show total amount of memory your server application is consuming. It is important especially when your server is a 32-bit app. While 32-bit application can consume up to 2 GB of RAM in theory, in practice any value above 1.0-1.1 GB means that you are in danger zone. That happens because memory used by uniGUI app is very fragmented and for other reasons Delphi's memory manager can not occupy 2 GB of RAM for an 32-bit uniGUI server. When your app runs out of memory it will start logging "Out of Memory" message and user will also see this message when a new session is started. This is a serious situation and may produce unpredictable results which will eventually lead to a server crash. If you are concerned with memory usage and your app will use more than 1GB of RAM it is better to target your server for 64-bit platform. In a 64-bit server it is very unlikely to get an "Out of Memory" error because even when physical RAM is consumed Windows will switch to virtual memory which is limited by swap space available on your hard drive. So in a 64-bit server you can consume memory much larger than the available physical memory. Of course, it must be noted that virtual memory is much slower than physical memory, so in a 64-bit uniGUI server when your system runs out of physical RAM you will notice a considerable drop in web application's performance and responsiveness.
 - **Memory (Internal/Process):** Internal memory is amount of memory internally used by the uniGUI application. Process is memory is the total memory occupied by process which is running the uniGUI app. For example, if your app is an ISAPI dll main process which is an ISAPI worker process will use more memory than internal memory used by the app itself. If your app relies on external dll libraries such as Midas, again memory used by process will be higher than memory used internally.
- **CPU Load:** This will show amount of CPU power currently being used. If your CPU has multiple cores (which is the case for all modern CPUs) this value will show total CPU usage for all CPU cores. An idle uniGUI server will use very low CPU ticks so when both OS and uniGUI app are idle CPU usage should not be more than 1%.
- **GDI Handles:** GDI handle is a limited system resource which is limited by OS. uniGUI is optimized in a way to keep GDI handle usage at a very low level. If your app uses components which rely on GDI handles you must check this value to see if it remain within a safe range. A typical uniGUI app will not consume more than 100 GDI handles when it is in idle state (there are no pending requests) even when there are hundreds of active sessions. uniGUI is designed in a way so a GDI handle is returned to OS as soon as it is not needed. If you see a constant increase in GDI handle then you must look for components which may leak GDI handles or overuse them. When your app runs out of GDI Handles it will start logging **EOutOfResources** exceptions in your log file. This is a serious problem which will eventually lead to server crash.
- Another important step is to monitor your web application log files. Stress test runs in background without any GUI output, so log file is the only way to see if your app runs properly. It is also the only way to see any error logs generated while running the stress test. Log files are placed under .\logs folder where your application's binaries are located. For each calendar day a separate log file is created. You must open related log file in a text viewer/editor and see if there are any unexpected logs such as Access Violations or other important Exceptions. By carefully analyzing the log file you can have a better idea about your application stability and scalability. If you need a more detailed log you can use a more advanced debugging tool such as **EurekaLog**.
- **Important:** You must take this into consideration that each stress test runs same recorded session in simultaneous threads for multiple times from same URL, so it is important to set the **ServerModule->SessionRestrict** parameter to **srNone** while you are testing your server.

2.2.2.3 Server Flood Protection Considerations

Apparently, **Stress Test Tool** will send lots of requests in a very short period of time. All these requests are sent from same IP to your uniGUI application which is being tested. Your server OS may interpret these high volume of requests as a flood DOS (Denial Of Service) attack and start blocking such requests. It may happen especially if you run **Stress Test Tool** from a remote PC with **PlayBack** speed set at **Max Speed** for a long period of time. Since flood detection algorithms varies from OS to OS and highly depends on particular configuration and used server software it is hard to predict when such blockages may occur. When server starts blocking requests you will notice lots of HTTP error messages in right side memo in Stress Test Tool. This shows that requests are being blocked and are not reaching their target.

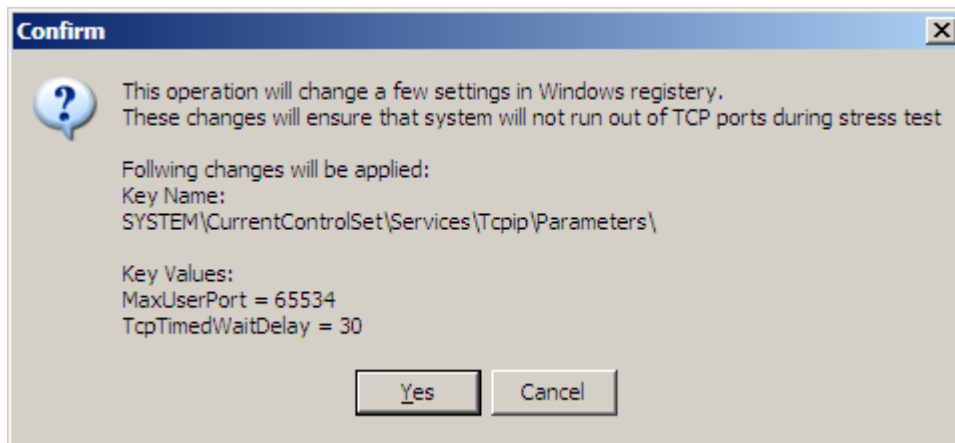
For example, behavior will be different among running uniGUI app as ISAPI module in IIS, standalone EXE and Windows Service. While running the **Stress Test Tool** from a remote PC may trigger flood protection, running both uniGUI app and **Stress Test Tool** on same desktop should work without triggering any flood protection mechanisms.

Additional to server OS some third party protection software such as **Kaspersky**, **Symantec** and etc. may add additional layers of protection to your server IP traffic which may disallow fast and continuous requests coming from Stress Test Tool. In such cases you need to slow down playback speed by setting it to **Real Time** and/or putting a longer delay between **Runs** to allow the requests pass through the anti-flood detection layers.

2.2.2.4 Additional Settings

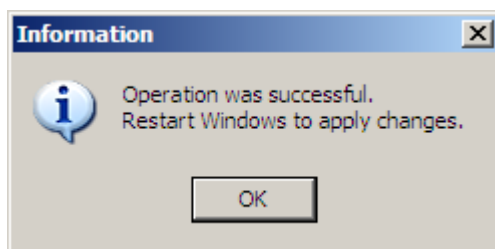
By nature **Stress Test Tool** creates hundreds of HTTP requests per second to perform a heavy stress on subject server. This may go beyond the limits of OS which is running the **Stress Test Tool**. One of these limitations is TCP port limit which is limited to a certain value by OS by default. In order to make a smooth run of **Stress Test Tool** we need to adjust the TCP port limit and increase it to a higher value. This requires a few registry setting in Windows Registry.

Good news this that **Stress Test Tool** can automatically make this change in registry for you. All you need to do is selecting **Other->Adjust Registry Settings** in **Stress Test Tool**.



Above dialog will be opened and it will give a brief information about the settings and will ask you to confirm the operation. These changes are totally harmless and will increase TCP port creation limit on your OS. You must make sure that you have started **Stress Test Tool** as **administrator** to be able to make this change.

A successful operation will show below message and ask you to restart your PC to finalize the operation.



Index

- E -

ExtRoot 15

Endnotes 2... (after index)