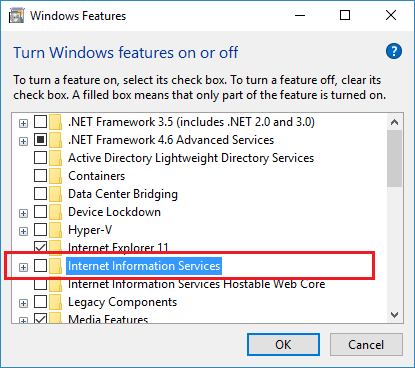
SETUP IIS FOR MARLIN 4.0

This document is a walk-through on how to setup IIS for a Marlin based application. It starts from the installation of IIS on your MS-Windows (Server) machine, and ends with the lowest detailed options.

**Step 1: Activate IIS on your machine**

To use IIS, you must setup your MS-Windows system to have IIS installed / activated. To do this you must go to the control panel, install programs and then choose the “Turn Windows features on or off” option. For Server versions of Windows, you need to go to the feature select menu. Once you get there. You must activate the “Internet Information Services”



Expand this dialog by dragging on the lower right corner and then click the plus before “Internet Information Services”. You will need ‘at least’ the following options:

Web Management Tools

* IIS Management Console
* IIS Management Scripts and Tools
* IIS Management Service

Application Development Features

* WebSocket Protocol

Common HTTP Features

* Default Document
* Directory Browsing
* HTTP Errors
* Static Content
* WebDAV Publishing

Health and Diagnostics

* All features !

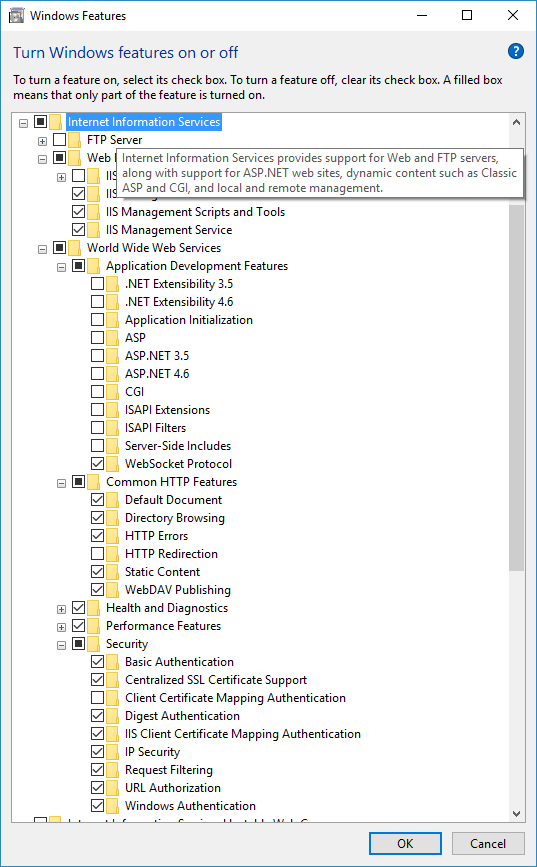
Performance Features

* All features! (all compression options)

Security

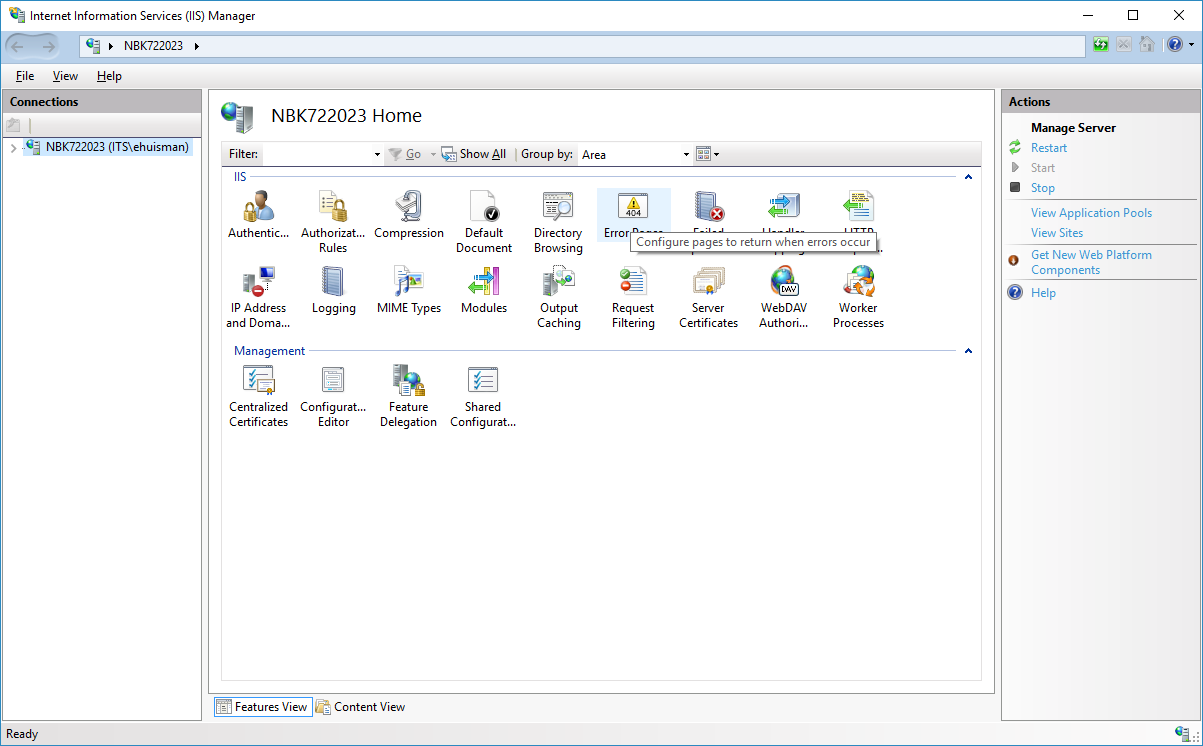
* Everything except “Client Certificate Mapping Authentication”

So the features select dialog should look like this:



Now press OK and if necessary, reboot your system.

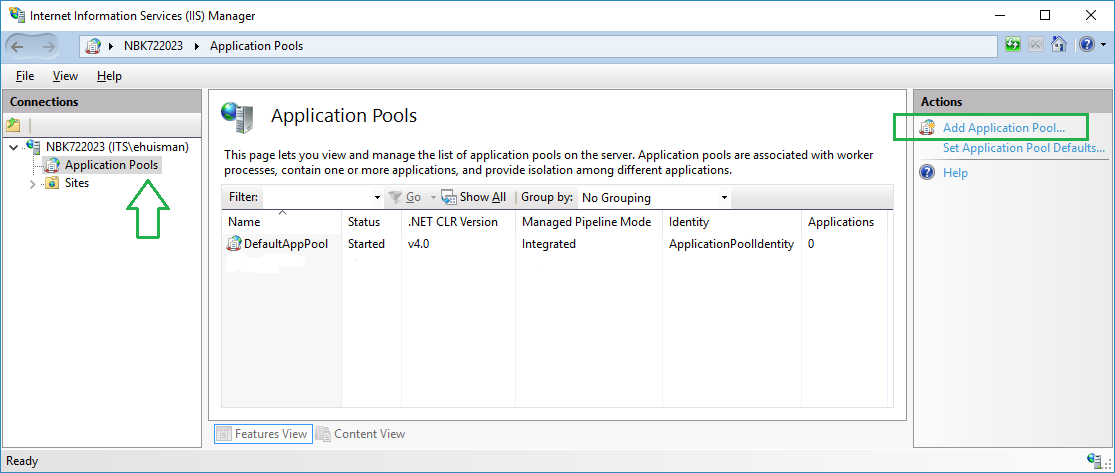
After this, you can start IIS-Admin from your start menu. If necessary, the program will prompt you for a administrators password. The program will look something like this:



Now that you are ‘up and running’, the next step is to introduce an application pool.

**Step 2: Create an application pool**

An application pool is a logic representation of one or more processes on your machine (w3wp.exe = world-wide-web worker process) that will run your application. An application pool can run one or more websites and applications that you define. To add an application pool, click open the computer connection, click on the “Application Pools” line and choose “Add application pool” on the right under “Actions”.



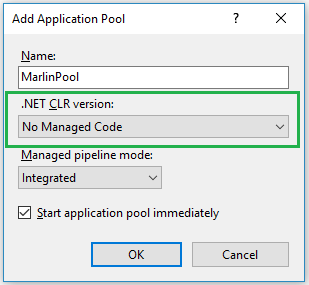
When you add an application pool, you can give it any name you want. Most preferable the name of your own application. For now I will give it the name “MarlinPool”.

**More important: under the option of choosing a “.NET CLR version” you \*\*MUST\*\* choose for the “No Managed Code” option, as a Marlin program will always run in Native mode.**

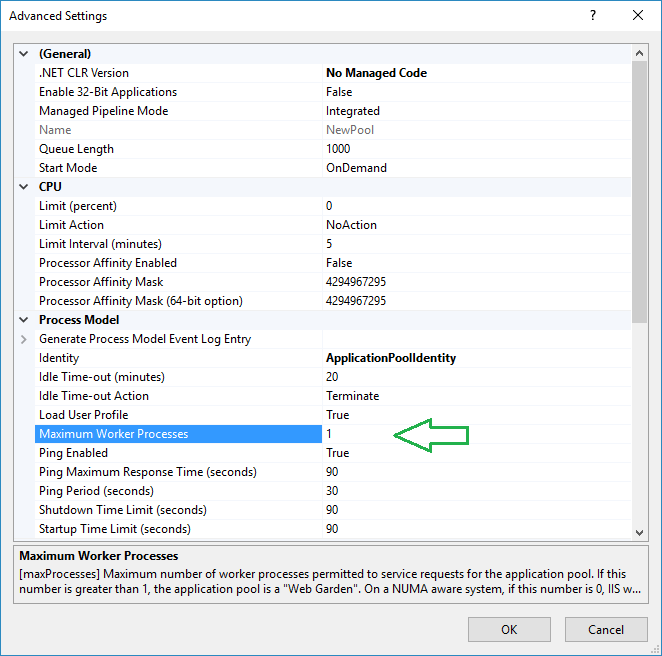
You are also advised to leave the pipeline mode on “Integrated”, as Marlin was not tested in the so-called “Classic” mode of IIS.

As a last point, you can choose to start the application pool immediately, or leave this option off. The latter will result in slower reactions of the website on first call, as everything has to be loaded in memory.

After choosing these options, you must see something like this:



Now choose “OK” and you will have your first (empty) application pool. Now that you have your application pool, you can click on “Advanced settings” in the action pane. This opens a form with more settings for this pool.

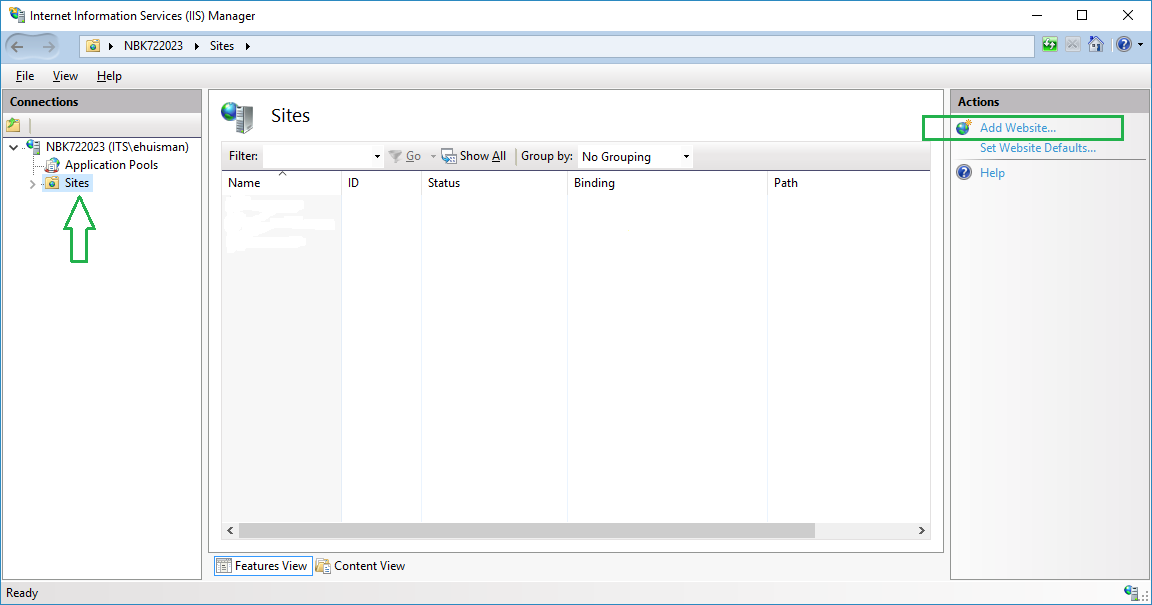


Please be advised that Marlin was designed to run in the “1 worker process” mode. If you want to experiment with more processes, and you have something like sessions within your application, you must be prepared to program some solution to extend the session information over process boundaries!

Now that we have our application pool, we can add our websites.

**Step 3: Adding the websites**

To add websites to IIS, we now go back to the main view of the IIS admin program and click on the Sites. Select “Add Website” from the action pane to the right.

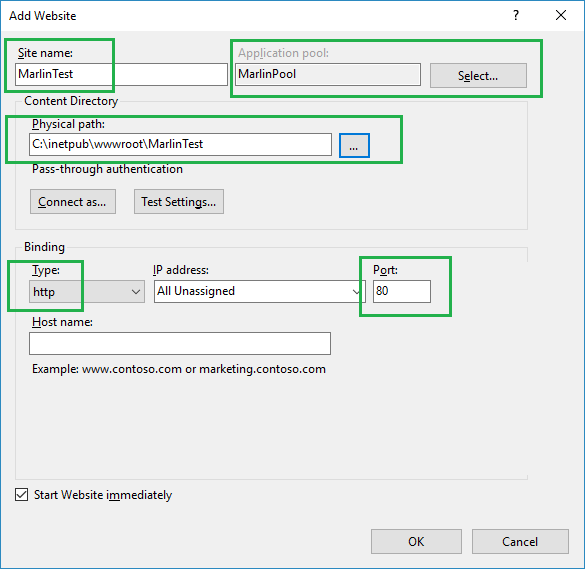


We will now setup three sites for the testing framework. The basic information to add these sites is represented in the following table:

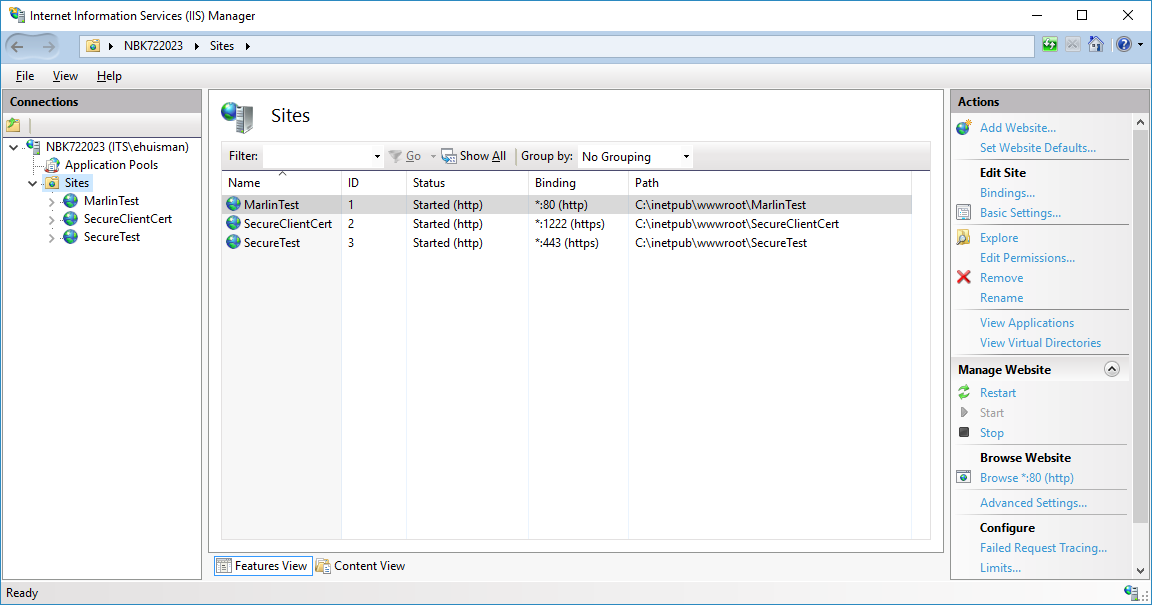
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Website | Url-Path | Directory | Binding | Port |
| MarlinTest | /MarlinTest/ | C:\inetpub\wwwroot\MarlinTest | http | 80 |
| SecureTest | /SecureTest/ | C:\inetpub\wwwroot\SecureTest | https | 443 |
| SecureClientCert | /SecureClientCert/ | C:\inetpub\wwwroot\SecureClientCert | https | 1222 |

You can add this information in the ‘add website’ form. Be aware that the second option is to select an application pool. Off course we will change de default “DefaultAppPool” to our newly created “MarlinPool” from step 2 of this walk-through.

The form for the first website will look like this, after you have added the information:



Repeat these steps for the other two websites, using the information from the table. The result must resemble something like this:



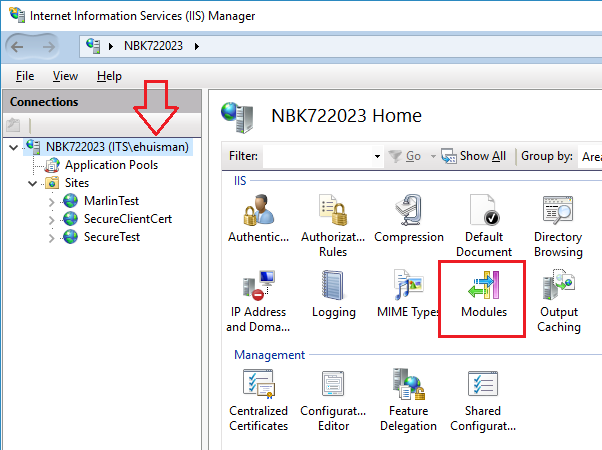
Now that we have defined our websites, we can work on the content and the special options for each site.

The first step is to add our compiled code to these websites.

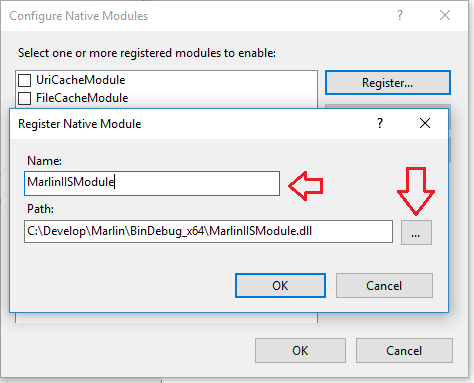
**Step 4: Adding a native module**

To be able to run our native (C++) compiled Marlin module, we must compile the “MarlinIISModule” in the Marlin server project. On my machine the project is in the “C:\Develop\Marlin” directory, so the debug runtimer is in the “C:\Develop\Marlin\BinDebug\_x64” directory as a native DLL: “MarlinIISModule.dll”

After compiling, go back to the main computer node in IIS-Admin and click on the “Modules” button in the ‘Home’ pane of IIS-Admin.



On the modules pane, click “Configure Native Modules” in the action pane, and then use the button “Register…” to register a new native module. Again: you can give it any name you want, as long as you select the output of the compiled program, containing the “HTTPServerIIS” and the “MarlinModule” classes, along with your ‘ServerApp’ derived application.

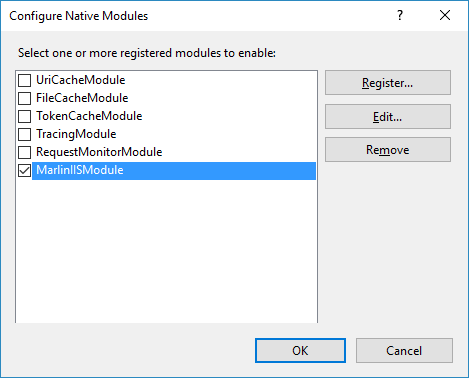


Click “OK”, and your module is ready for activation.

There are now two ways to activate the MarlinIISModule program.

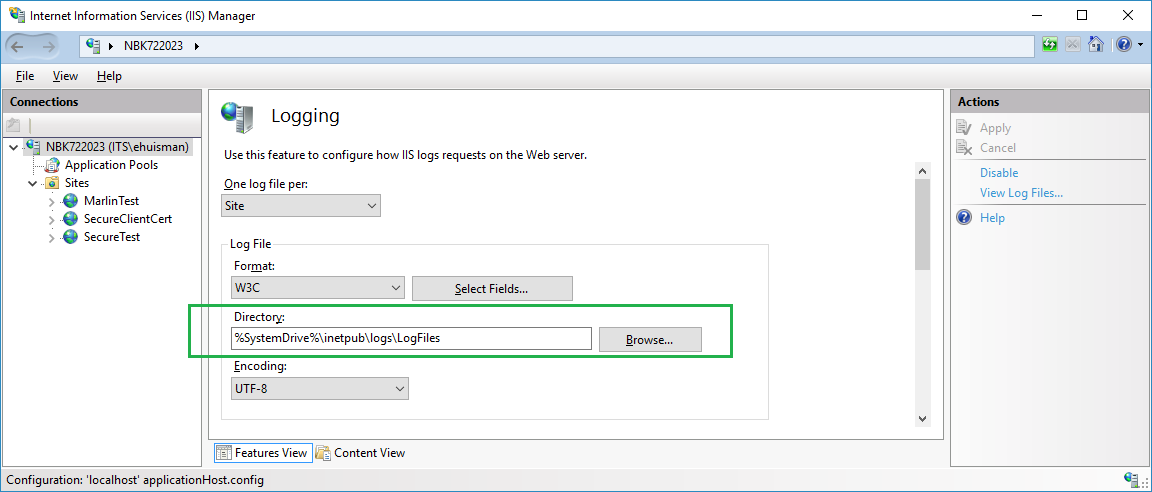
You can activate it right here, on the computer node. This will activate the module for all sites that your machine is servicing. Not only your site, but EVERY site!.

A better way to activate the module, is to leave it here as it is, and go to each site individually. Click on the modules icon on the site, choose “Configure Native Modules…” for that site, and then activate the Marlin module. It is slightly more work, but it ensures that applications installed earlier or later will not also start the Marlin module, and thus waist computer time on non-serviced calls.



**Step 5: Configuring the logging**

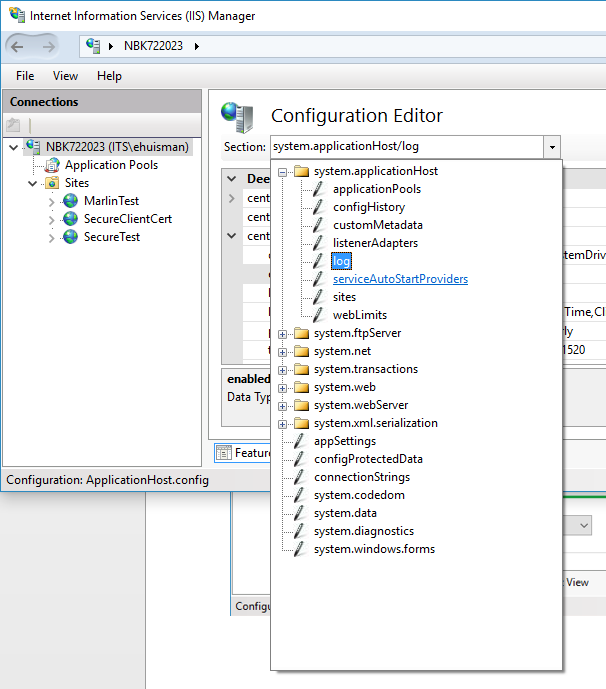
Configuring the logging in Marlin is done in two steps. First you must activate where to put your logfile. The setting for this is taken from the W3C logfile location of IIS. You can find this location on the main computer pain when clicking the “Logging” icon:



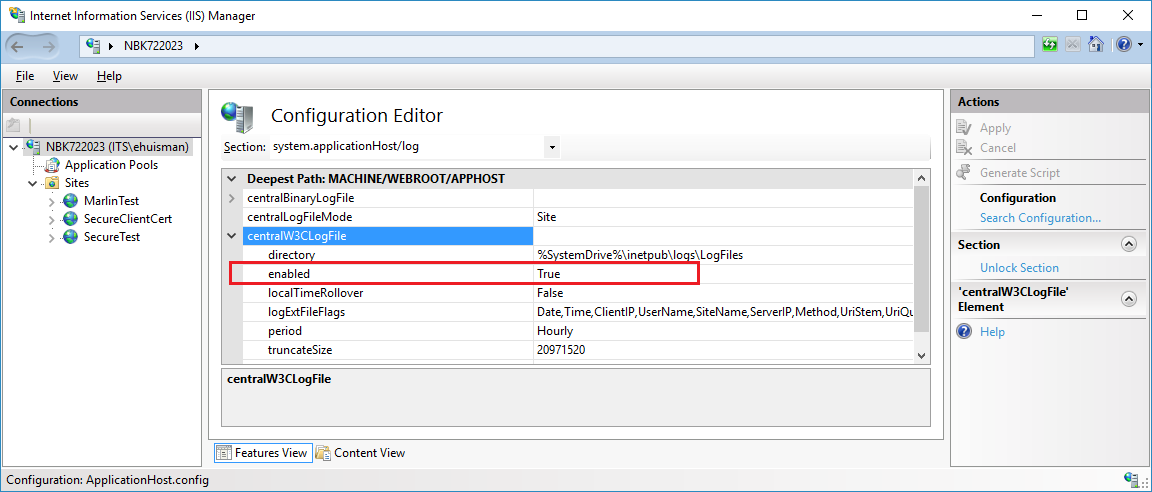
Turning logging on or off is a bit harder to do.

* Go to the main computer pan and click “Configuration Editor”
* Under Section, choose “system.ApplicationHost” and then “Log”
* Click the option “centralW3CLogfile”

Like this:



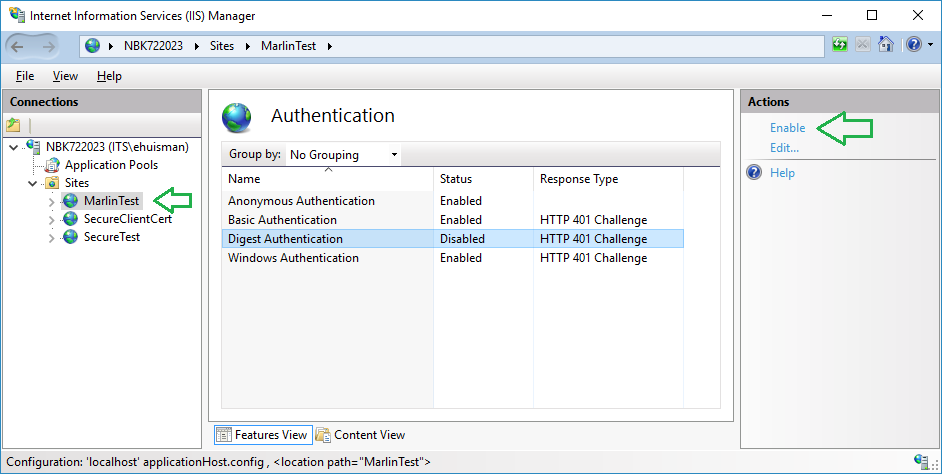
And then you can enable/disable the logging.



**Step 6: Configuring the authentication**

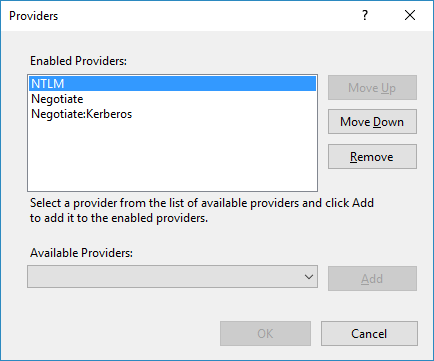
Now we can activate the activation methods as supported by Marlin. Select each site and for each site click on the “Authenticate” icon. Now select “Enable” for each authentication method you wish to activate for your site. Be aware that if you do NOT want to have authentication, you must activate “Anonymous authentication” for your website. You can leave the default response type to “HTTP 401 Challenge”, as this is the standard HTTP way of letting a client know that it should authenticate itself.

If one of these authentication methods do NOT show up on this pane, you must go back to step 1 (Activating IIS on your machine) as you might have forgotten to select the needed authentication methods.



Some of these authentication methods have extra actions on the Action pane. Most notably we must activate authentication providers for the Windows Authen­tication method. You can do this by activating the “Windows Authentication” method and then choose ‘Providers’ for that method.

Marlin applications now in action use the standard NTLM authentication method. This is the standard Windows authentication, where we can use single sign on from other machines in the same domain. So Add this provider and press “Move Up” until it is in the first position. This is what you must have in the end.

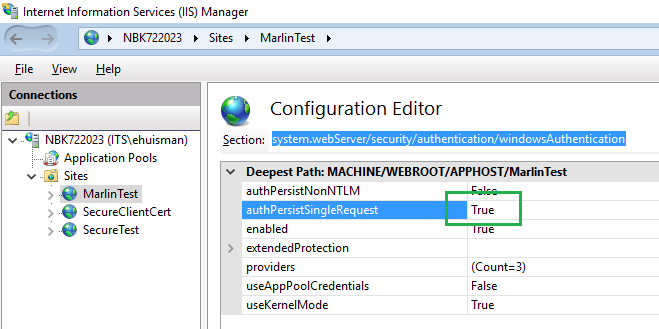


To activate (or de-activate) the NTLM caching of login requests, you must work a little harder.

Go to the website main pane, press the “Configuration Editor” icon and then select the configuration path:

* System.WebServer
* Security
* Authentication
* WindowsAuthentication

The confusing fact is that if you select ‘True’ the caching is OFF. And it is ON if you select ‘False’.



**Step 7: Setting up SSL**

Two of the test sites are secure sites over https. To make these work, we need a SSL certificate. So the first step is to create a test certificate. Or we can import the testing certificates from the “Certificate” directory in the Marlin project. In a real life production scenario, we would of course import a certificate from a third party Certification Authority.

Step 7a: import a certificate

Follow these steps:

* Go to the main computer node in IIS-Admin
* Select the “Server Certificates” icon
* Select “Import…” on the actions pane
* Select your \*.pfx certificate file
* Add your password for the certificate
* Select a store, by preference we normally choose the “Web Hosting” store
* Press “OK”

Step 7b: Create a testing certificate

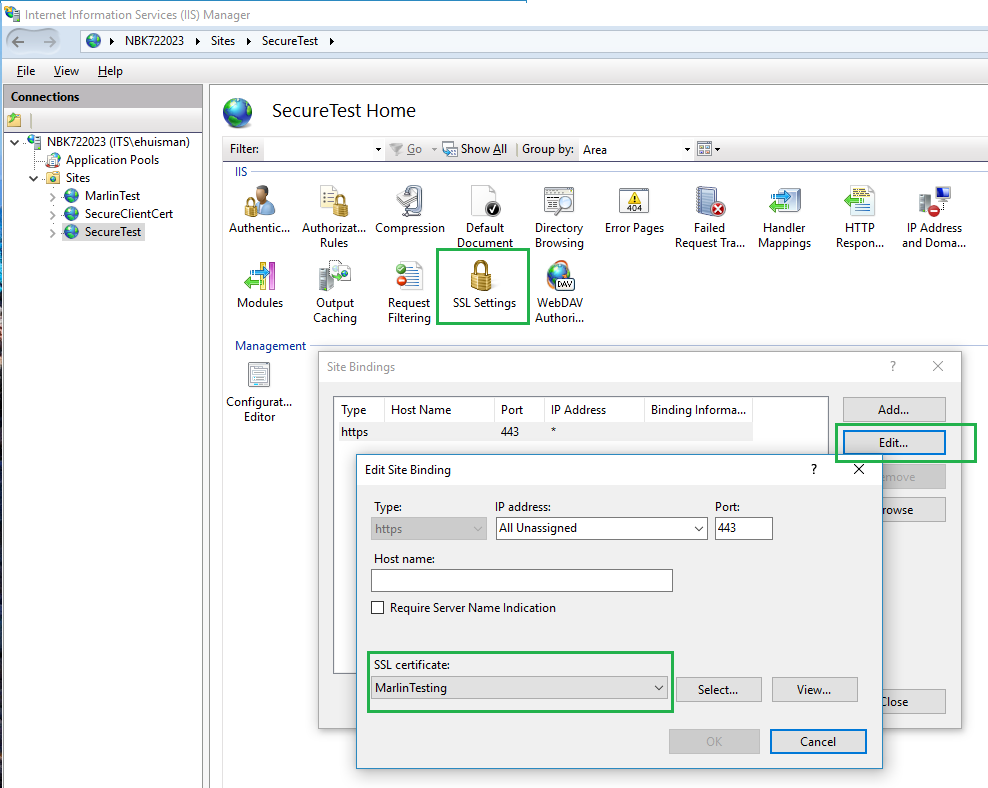
Follow these steps:

* Go to the main computer node in IIS-Admion
* Select the “Server Certificates” icon
* Select “Create self-signed Certificate…” from the actions pane
* Fill in a friendly name
* Select the “Personal” store
* Press “OK”

Step 7c: Activate the certificate on the site

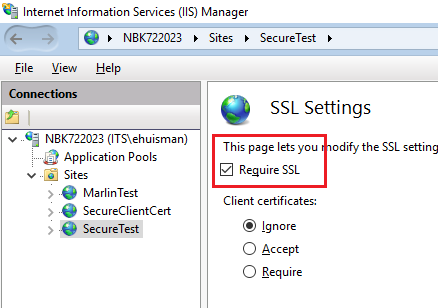
Follow these steps:

* Go to the site pane
* Select the “Bindings…” action from the action pane
* Select the binding and click the “Edit” button
* Select the desired certificate



7d: Activating SSL

Now click the “SSL Settings” and place a mark for “Require SSL”.



**Step 8: Configuring the client certificates**

Now that we know how to setup SSL for a site, we can do this for the “SecureClientCert” site as well. And as you might guess, we can activate “Accept” on the previous step for that site (see figure above). But that is not enough. What’s missing is the information about which client certificates to trust. This information can only be added to the “Configuration Editor”

So to configure the client certificate for the “SecureClientCert” site, you must follow these steps:

* Go to the site pane for “SecureClientCert”
* Click on the “Configuration Editor” icon
* Select the path under
  + System.WebServer
  + Security
  + Authentication
  + iisClientCertificateMappingAuthentication
* Set the enabled option to “True”
* Select the last field (“OneToOneMappings) and click the […] symbol at the right
* Now you are in the Certificate Collection Editor. Here you can ‘Add’ a client certificate
* (See the figure on the next page).

**However:**

Before you can add the certificate you must export it to a Base64 representation. To do this you must open the certificate in the Microsoft Console (mmc) as described in the documentation in the “Certificates” directory.

Now export the certificate to a \*.cer file, and open that file in Notepad or better Notepad++.

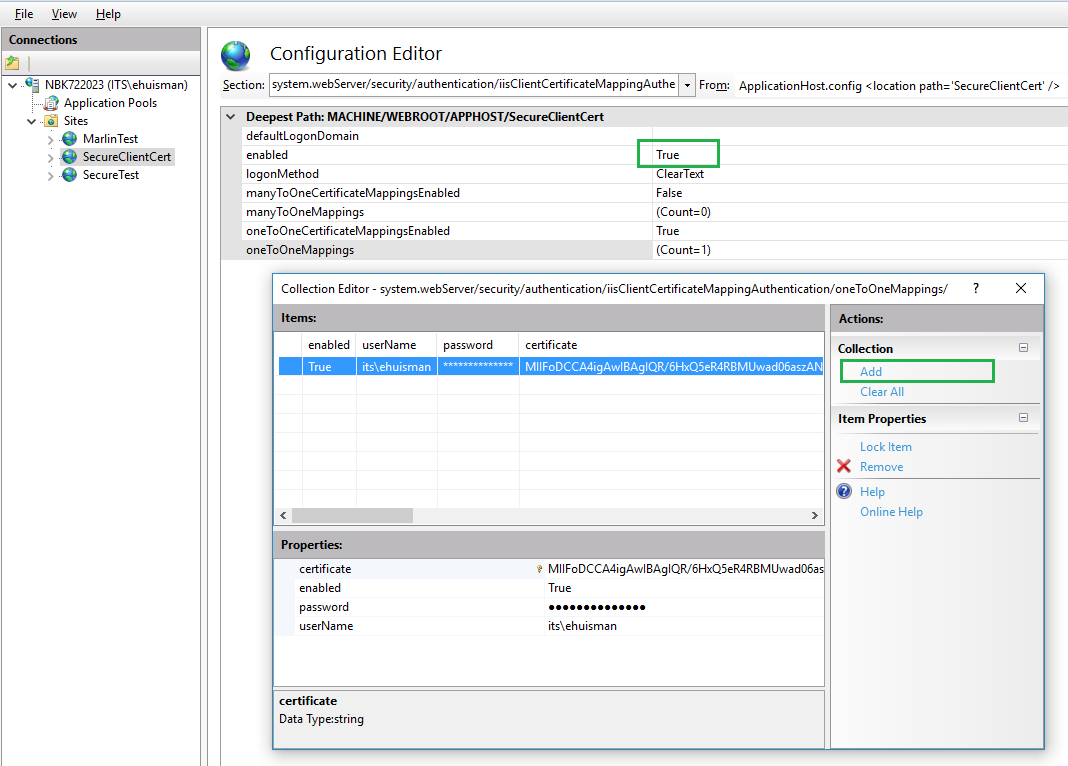
Reformat all the lines of the certificate to just 3 lines, where the base64 representation is on 1 (one !!) line, somewhat like this

-----BEGIN CERTIFICATE-----

MIIFoDCCA4igAwIBAgIQR/6H … very long line …… U01bE8kvERgaLg==

-----END CERTIFICATE-----

Now you can cut and paste \*\*\***the middle line only\*\*\*** to the IIS-Admin application



So that certificate that you see here is NOT the thumbprint of the certificate but the COMPLETE certificate in Base64 representation. The point here is to bind the certificate to an explicit AD user.

**Step 9: Placing the static content**

Now that we have configured our sites, we can place the static content (files) in those sites. The testing framework of Marlin uses just a few. You can use the files from the documentation directories, or you can make up your own, as long as the file names stay the same

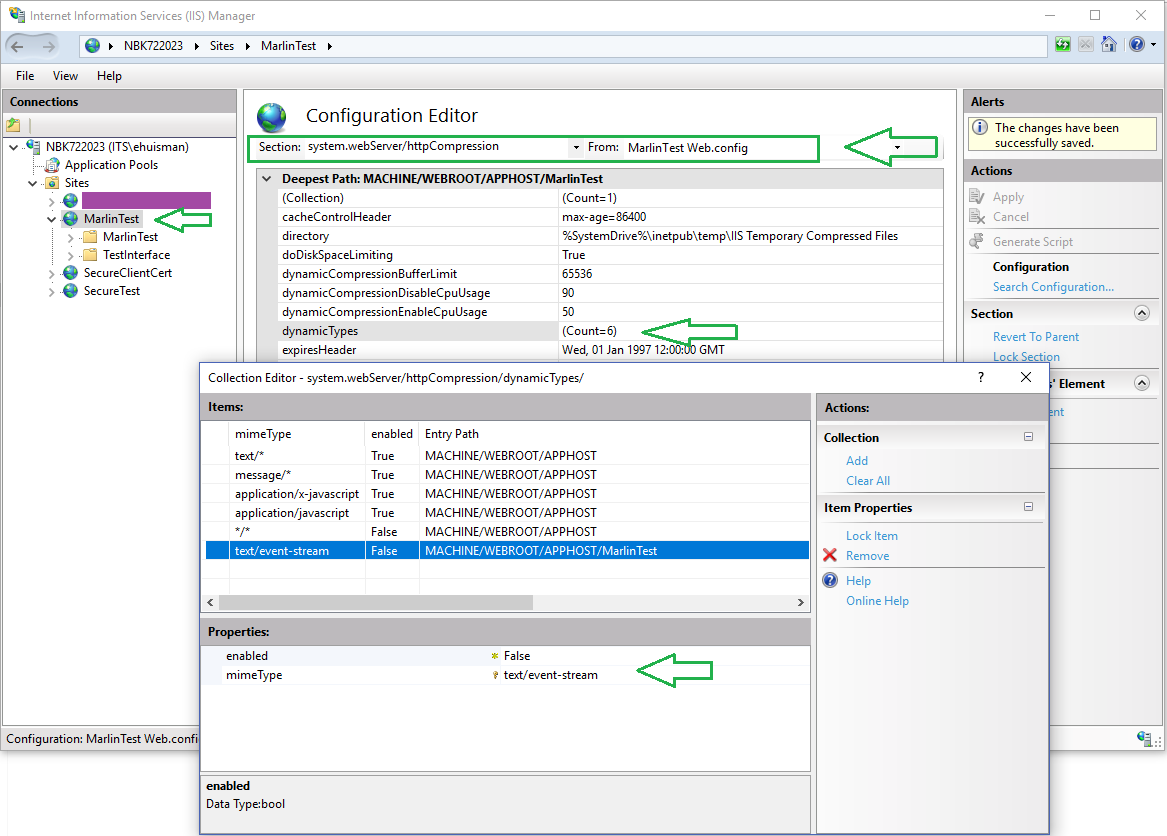
|  |  |
| --- | --- |
| Site | File |
| MarlinTest | C:\inetpub\wwwroot\MarlinTest\MarlinTest\Site\FileOne.html |
| MarlinTest | C:\inetpub\wwwroot\MarlinTest\MarlinTest\Site\FileTwo.html |
| SecureClientCert | C:\inetpub\wwwroot\SecureClientCert\SecureClientCert\codes.html |
| SecureTest | C:\inetpub\wwwroot\SecureTest\SecureTest\codes.html |

After the first run, the following file will appear:

|  |  |
| --- | --- |
| Site | File |
| MarlinTest | C:\inetpub\wwwroot\MarlinTest\TestInterface\MarlinWeb.wsdl |
|  |  |

**Step 10: Enabling the event-streams**

In order to let the even stream (SSE = Server Sent Events) to work correctly, you should disable the dynamic compression of a request’s response. This can best be done on the website level where you expect those event streams for just the “text/event-stream” content type. The quickest way to configure this is via the configuration editor under “WebServer/httpCompression” and then adding a dynamic content type.



**Step 11: Bringing it all together**

Now we can be about our business and run the application. To do this we must take the following steps

1. Start the IIS service

* Go to the computer pane
* Select “Start” from the actions pane

1. Start the application pool

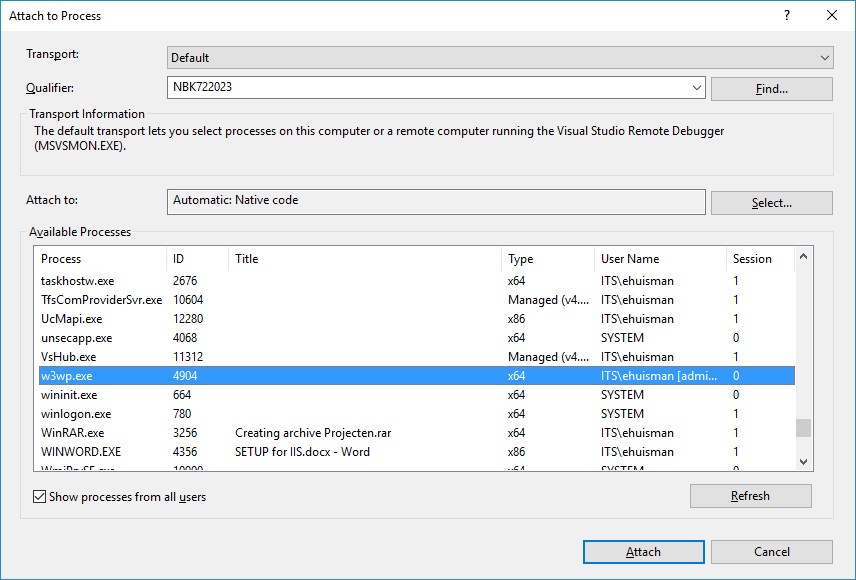
* Go to the application pool pane
* Select the “MarlinPool” (or whatever you named your pool)
* Select “Start” from the actions pane

1. Start the websites

* Go to the pane of the website
* Select “Start” from the actions pane

To debug your website application, follow these steps after starting the application pool:

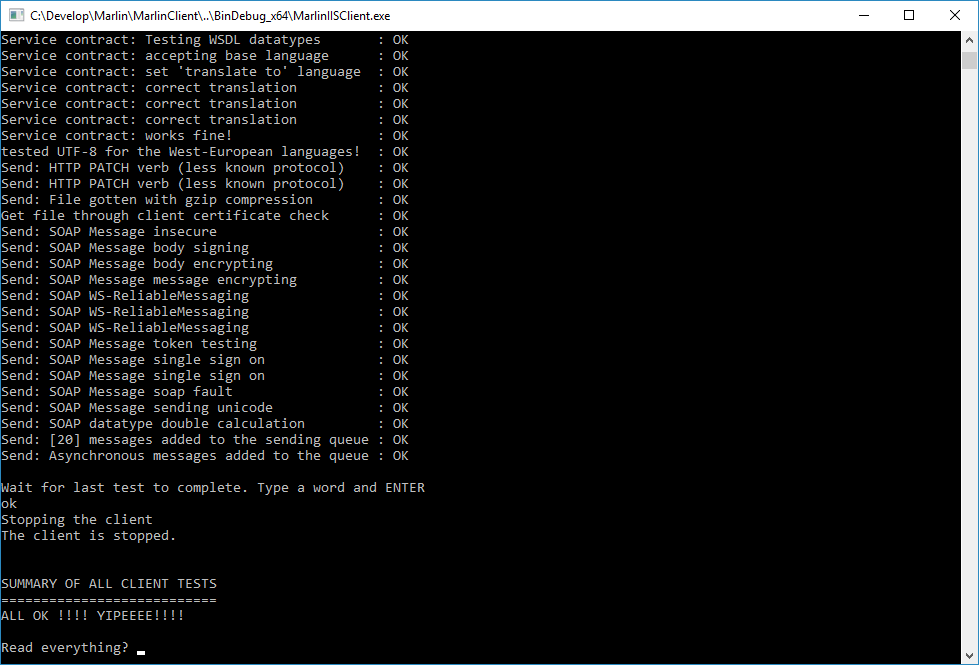
1. Start Visual Studio ‘in administrator mode’
2. Load your Marlin application’s solution
3. Make sure everything is (re)compiled
4. Select “Debug” from the main menu
5. Select “Attach to process…”
6. Find the ‘w3wp.exe’ process and press “Attach”
7. Happy programming…



**Step 12: Extra step: Running the test suite**

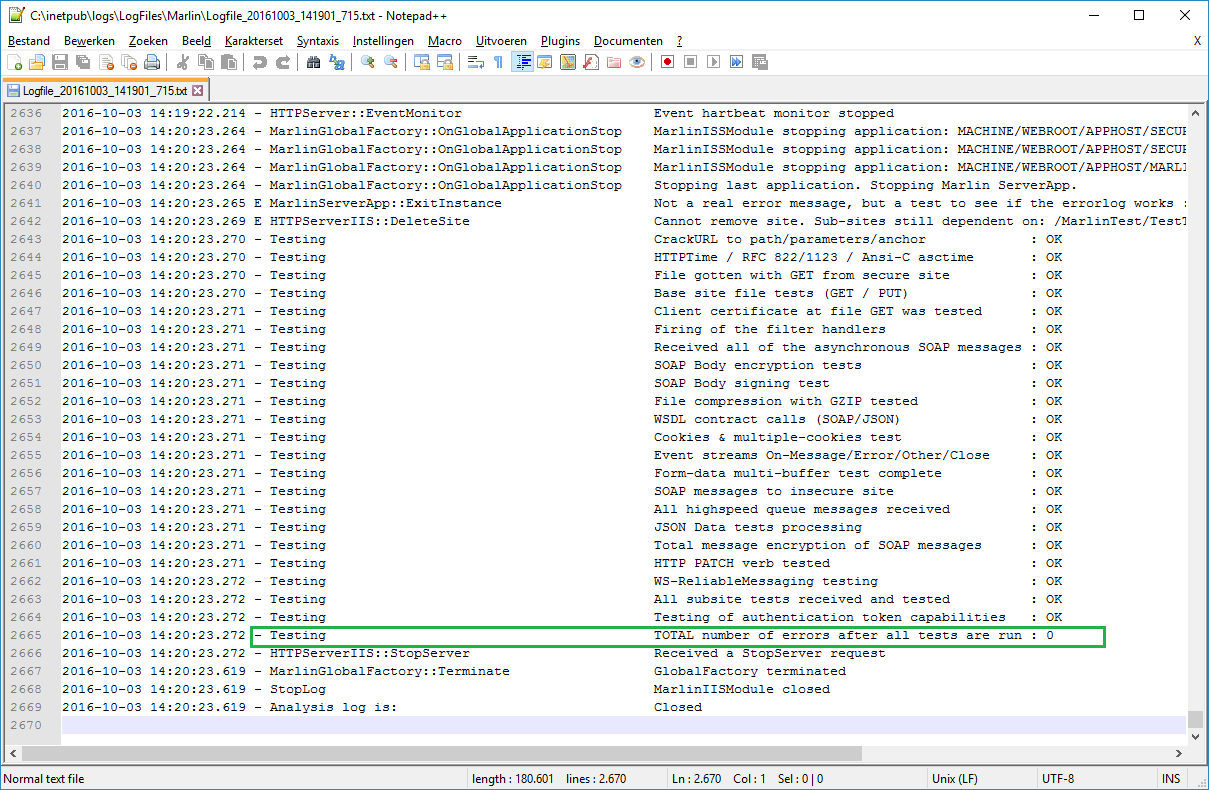
As an extra frosting on the pie in this manual: we will run the standard regression test for the Marlin framework.

After we have taken all the steps here, we start an extra Visual Studio and start the “MarlinClient” solution. Compile and run the “MarlinIISClient” project in this solution. You must see something like this:



Now go to the application pools pane and stop your Marlin pool.

After that you can check in the logfiles directory the end of the marlin logfile, to find the end result.



**Procedures: Password changed**

To save yourselves some troubles: in case you change the password of an account that the application pool is running under, you will only see the effect that an application pool is not starting. No warnings appear either in the IIS-Admin or the logfiles. When searching the Windows Event viewer, you can finally find the fact that the WAS service could not startup the application pool.

And of course: Changing the password of the application pool’s identity fixes the situation ☺

