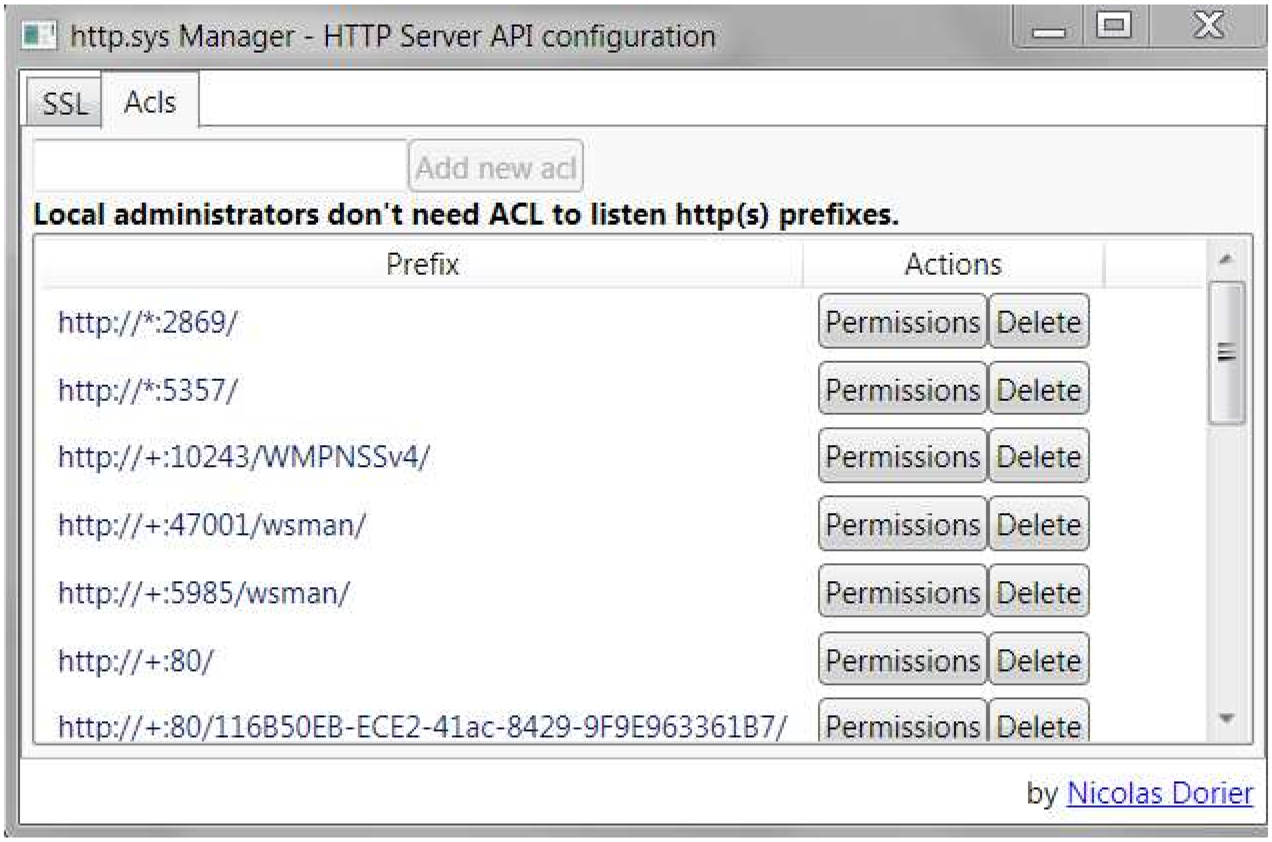
Demystify http.sys with HttpSysManager

Configure the HTTP traffic on your local machine. A nice alternative for netsh http.

- **Go to HttpSysManager on Codeplex**



Local administrators don't need ACL to listen http(s) prefixes.

Prefix	Actions
http://*:2869/	Permissions Delete
http://*:5357/	Permissions Delete
http://+:10243/WMPNSSv4/	Permissions Delete
http://+:47001/wsman/	Permissions Delete
http://+:5985/wsman/	Permissions Delete
http://+:80/	Permissions Delete
http://+:80/116B50EB-ECE2-41ac-8429-9F9E963361B7/	Permissions Delete

by Nicolas Dorier

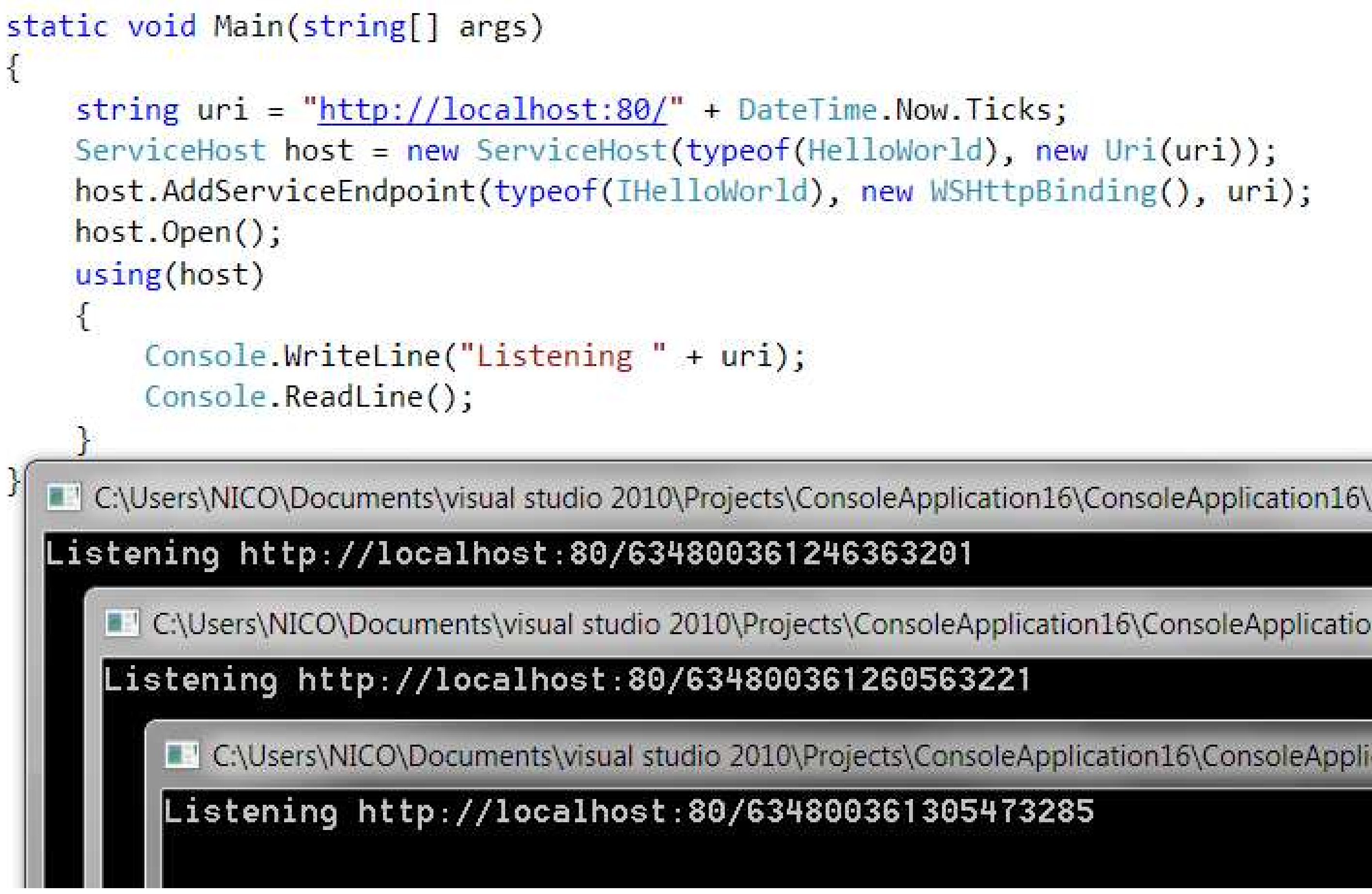
Demystify http.sys with HttpSysManager

Spotlight on HTTP.SYS

Before Windows server 2003, life was simple for http servers creator : just open a socket on an endpoint (IP:Port), listen incoming traffic and parse it.
All was fine and good, except that all applications, for firewall reason, wanted to use port 80 (http) and 443 (https).

As you might know, with the socket model, only one application can listen an endpoint at any given time (IP:Port).
Everybody claimed port 80 and 443, and terrible wars started on internet to know what the ruler would be.

Then Windows server 2003 server released, and a new kernel driver borned, Microsoft gave him the name of `http.sys`.
This driver is meant to listen http traffic and dispatch based on the URL to processes : now multiple processes will be able to listen HTTP traffic on the same port.
Here is a simple proof :



```
static void Main(string[] args)
{
    string uri = "http://localhost:80/" + DateTime.Now.Ticks;
    ServiceHost host = new ServiceHost(typeof(HelloWorld), new Uri(uri));
    host.AddServiceEndpoint(typeof(IHelloWorld), new WSHttpBinding(), uri);
    host.Open();
    using(host)
    {
        Console.WriteLine("Listening " + uri);
        Console.ReadLine();
    }
}
```

C:\Users\NICO\Documents\visual studio 2010\Projects\ConsoleApplication16\ConsoleApplication16\

```
Listening http://localhost:80/634800361246363201
```

C:\Users\NICO\Documents\visual studio 2010\Projects\ConsoleApplication16\ConsoleApplicatio

```
Listening http://localhost:80/634800361260563221
```

C:\Users\NICO\Documents\visual studio 2010\Projects\ConsoleApplication16\ConsoleAppli

```
Listening http://localhost:80/634800361305473285
```

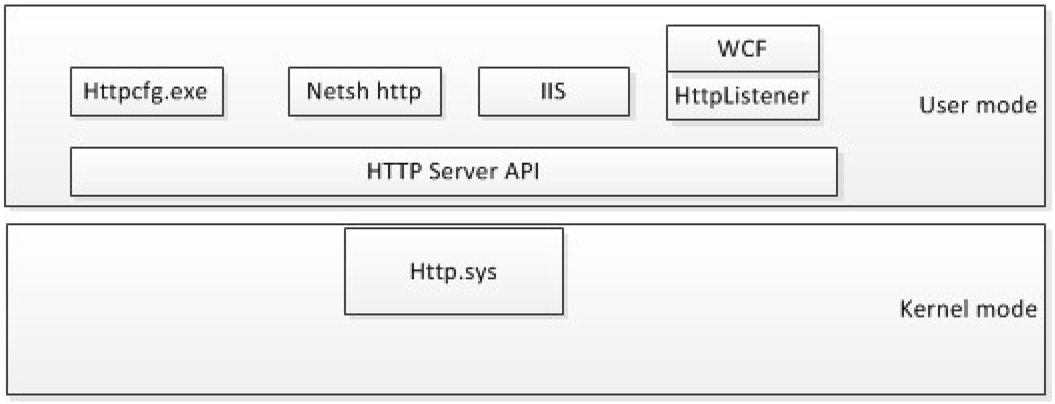
A public windows API was created around the new born called HTTP Server API, and also a set of tools making use of it like

Hide Copy Code

```
httpcfg (the
                                                  old way)
```

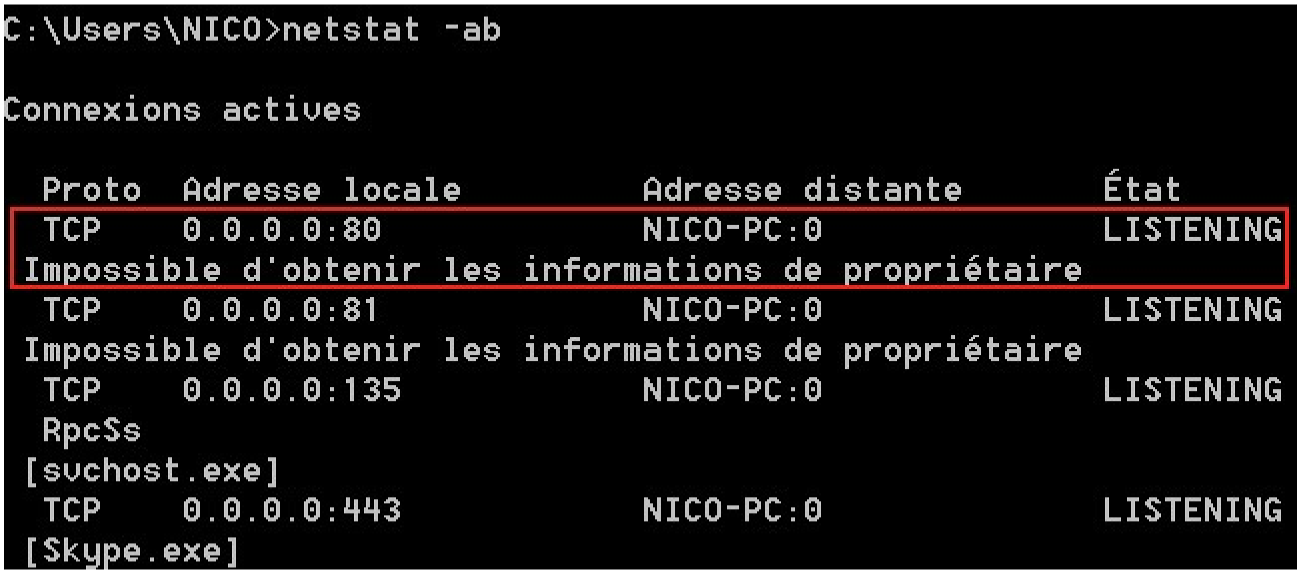
or `netsh http (the new way)`.

IIS depends on `http.sys`, as well as our beloved class `HttpListener` which is in really a simple wrapper around the HTTP Server API and so is WCF `Http(s)TransportBinding`.



You don't believe me ? You think it is a conspiracy ? Check it yourself.

Translation : Impossible to get any information about the owner. (because a kernel driver, `http.sys`, is not a process)

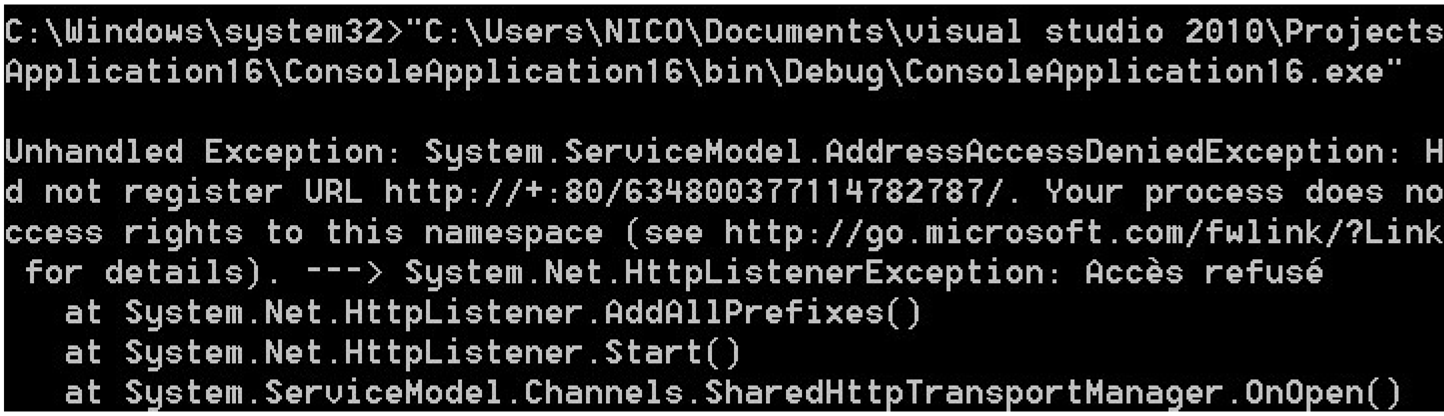


```
C:\Users\NICO>netstat -ab

Connexions actives

  Proto  Adresse locale         Adresse distante       État
  TCP    0.0.0.0:80             NICO-PC:0              LISTENING
 Impossible d'obtenir les informations de propriétaire
  TCP    0.0.0.0:81             NICO-PC:0              LISTENING
 Impossible d'obtenir les informations de propriétaire
  TCP    0.0.0.0:135            NICO-PC:0              LISTENING
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:443            NICO-PC:0              LISTENING
 [Skype.exe]
```

And what are the implication for you, fellow developer ? The implication is wierd production deployment crash like this one.



```
C:\Windows\system32>"C:\Users\NICO\Documents\visual studio 2010\Projects
Application16\ConsoleApplication16\bin\Debug\ConsoleApplication16.exe"

Unhandled Exception: System.ServiceModel.AddressAccessDeniedException: H
d not register URL http://+:80/634800377114782787/. Your process does no
ccess rights to this namespace (see http://go.microsoft.com/fwlink/?Link
 for details). ---> System.Net.HttpListenerException: Accès refusé
   at System.Net.HttpListener.AddAllPrefixes()
   at System.Net.HttpListener.Start()
   at System.ServiceModel.Channels.SharedHttpTransportManager.OnOpen()
```

For google referencement and all desperate developers that will ask google to save their soul, I copy the message for you, in plain english :
Unhandled Exception: System.ServiceModel.AddressAccessDeniedException : HTTP could not register URL http://+:80/634800377949733185/. Your process does not have access rights to this namespace (see http://go.microsoft.com/fwlink/?LinkId=70353 for details). ---> System.Net.HttpListenerException: Access is denied

The question I will first respond, is the one you ask just after seeing this message. This is a very famous question among developers, and an endless source of debate: **Why in the hell did it work on my machine ?**

The show answer is also very common: **It worked because you was in the local administrators group of your machine**.

The long answer is that it does not work because the user in production environment was not admin **and the user in production environment did not have right on the URL http://+:80/**.

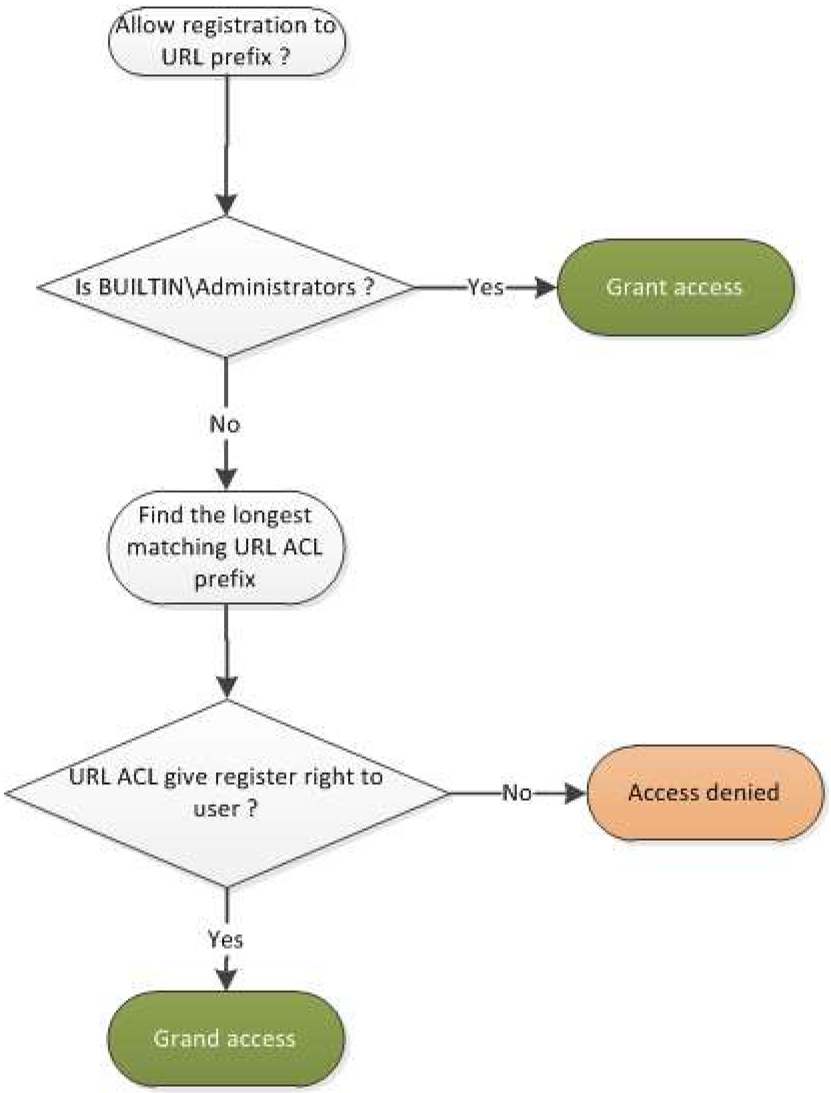
Routing, registrations, URL ACL and delegation

When IIS, or any application that use HTTP Server API listen on some HTTP request path, they need to **register** a url prefix on `http.sys`, we call this process **registration**.

When an incoming request is picked by `http.sys`, it needs to deliver the message to the right **registered**application, we call that **routing**.

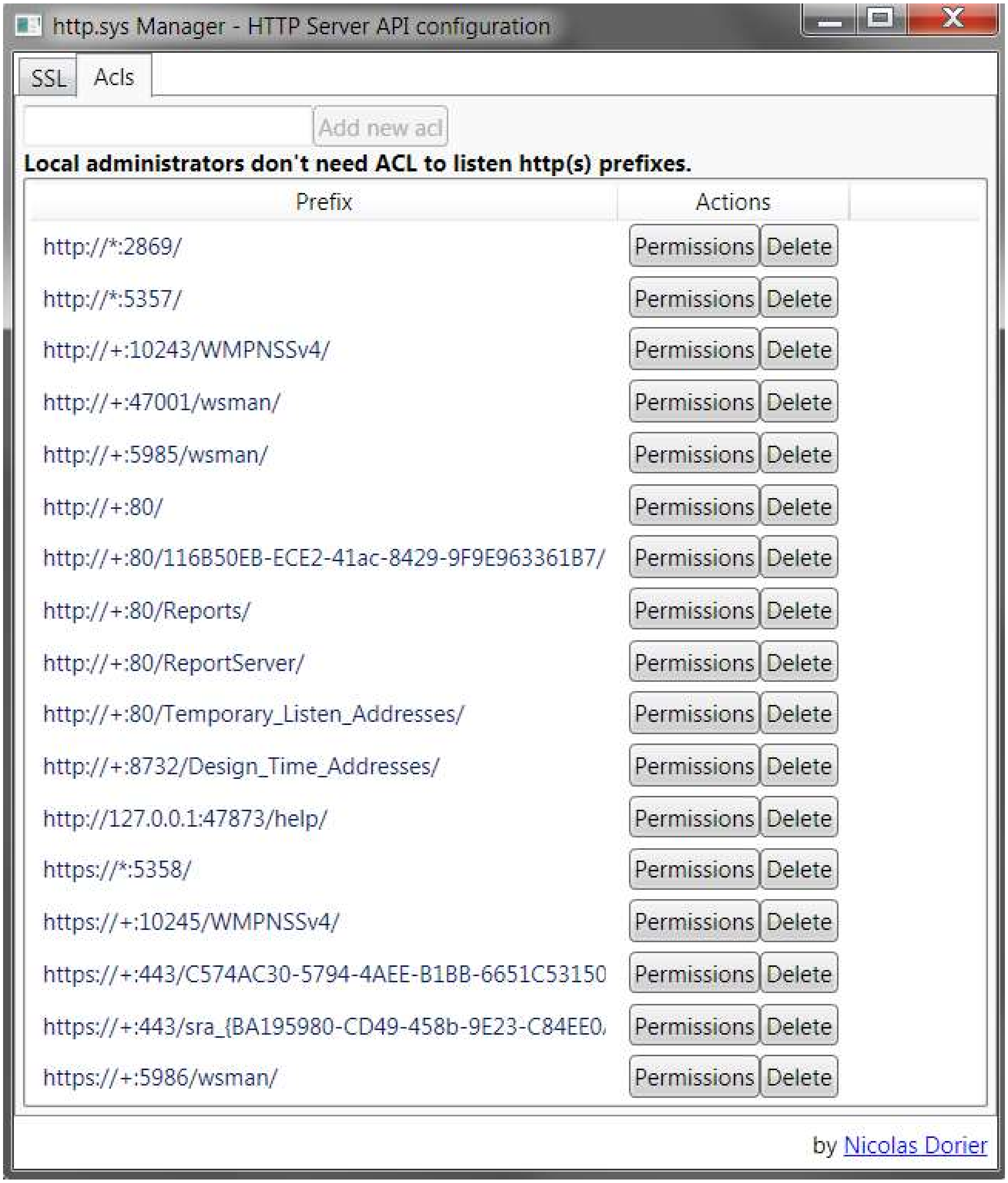
As you have seen, when the application register every thing works fine if you are local administrator. If you are not, `URL ACL` are checked.

An URL ACL is just a URL prefix associated to an ACL.


Allow registration to URL prefix ?
Is BUILTIN\Administrators ?
Yes
Grant access
No
Find the longest matching URL ACL prefix
URL ACL give register right to user ?
No
Access denied
Yes
Grand access

.

You can play with it with command line `netsh http show|add|delete urlacl`, but I don't recommend it, because I created a -almost- nice UI around HTTP Server API : **HttpSysManager**.



http.sys Manager - HTTP Server API configuration

SSL Acls

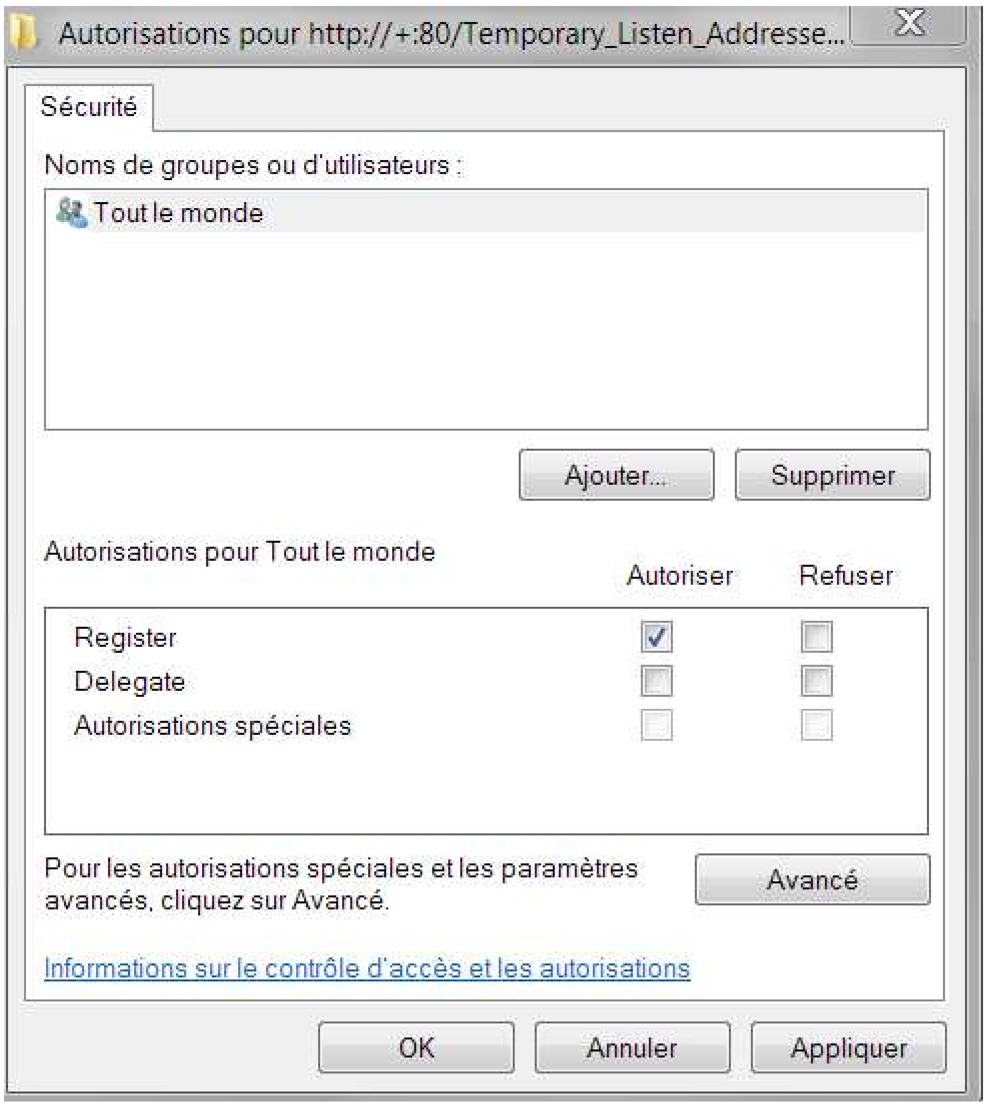
Add new acl

Local administrators don't need ACL to listen http(s) prefixes.

Prefix	Actions
http://*:2869/	Permissions Delete
http://*:5357/	Permissions Delete
http://+:10243/WMPNSSv4/	Permissions Delete
http://+:47001/wsman/	Permissions Delete
http://+:5985/wsman/	Permissions Delete
http://+:80/	Permissions Delete
http://+:80/116B50EB-ECE2-41ac-8429-9F9E963361B7/	Permissions Delete
http://+:80/Reports/	Permissions Delete
http://+:80/ReportServer/	Permissions Delete
http://+:80/Temporary_Listen_Addresses/	Permissions Delete
http://+:8732/Design_Time_Addresses/	Permissions Delete
http://127.0.0.1:47873/help/	Permissions Delete
https://*:5358/	Permissions Delete
https://+:10245/WMPNSSv4/	Permissions Delete
https://+:443/C574AC30-5794-4AEE-B1BB-6651C53150	Permissions Delete
https://+:443/sra_{BA195980-CD49-458b-9E23-C84EE0/	Permissions Delete
https://+:5986/wsman/	Permissions Delete

by Nicolas Dorier

With some familiar UI to set ACLs… (Tout le monde = Everyone)


Autorisations pour http://+:80/Temporary_Listen_Addresse...

Sécurité

Noms de groupes ou d'utilisateurs :

Tout le monde

Ajouter... Supprimer

Autorisations pour Tout le monde

	Autoriser	Refuser
Register	✓	
Delegate		
Autorisations spéciales		

Pour les autorisations spéciales et les paramètres avancés, cliquez sur Avancé.

Avancé

Informations sur le contrôle d'accès et les autorisations

OK Annuler Appliquer

Some of you may recognize that this is the ACL dialog for the callback url for duplex HTTP scenario in WCF… this ACL is set during WCF installation, so that every user can register callback addresses.

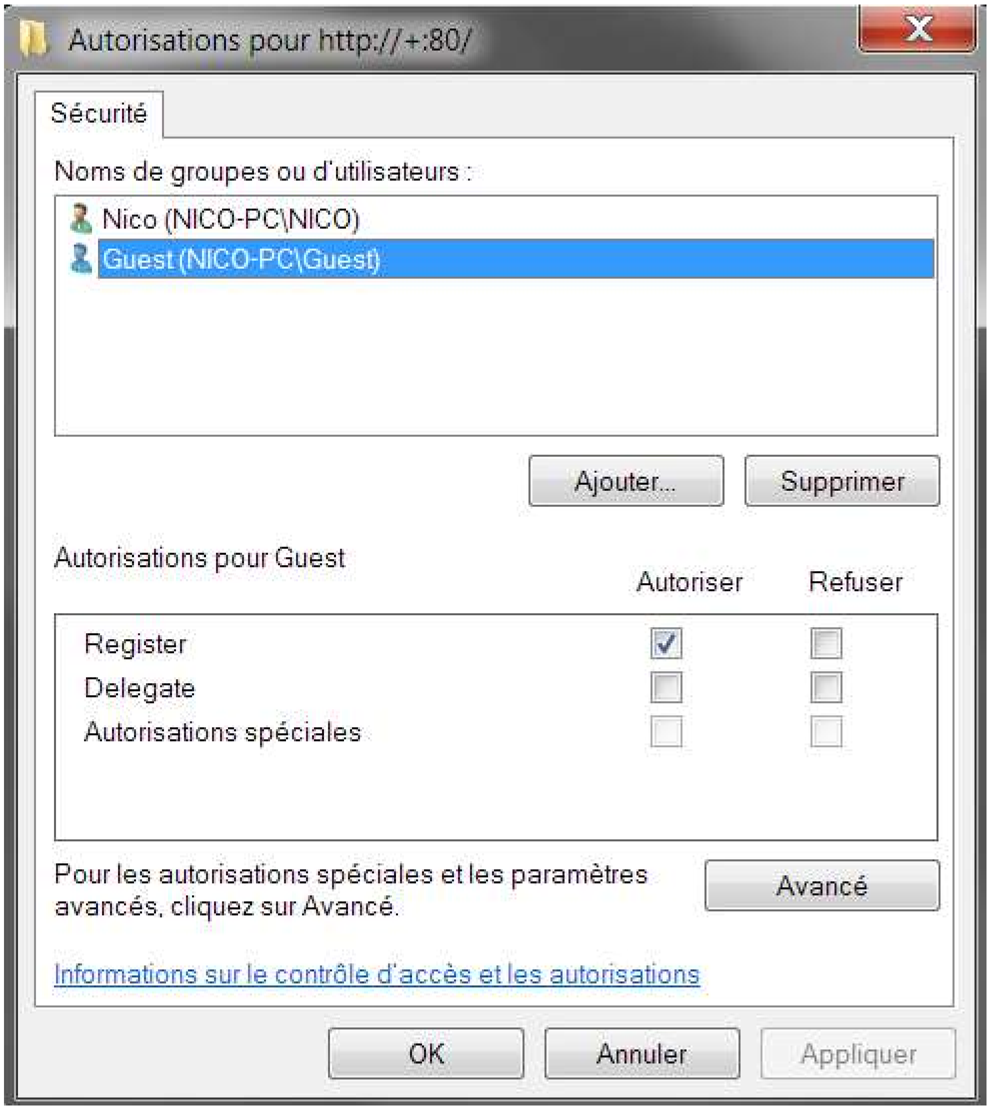
Given my current ACLs, a registration to **http://+:80/ReportServer/test/blabla** will use the ACL **http://+:80/ReportServer**.
On the other hand a registration to **http://+:80/test/blabla** will use the ACL of **http://+:80/**.

Always remember : **the longest match rule** applies.

You can see there is two different rights in the ACL window : **register** and **delegate** (Special authorization does not apply), and we covered **register**.

Delegation means that you give rights to group or user to add, modify and remove url acl for sub request path.

Example, here is the rights for **http://+:80/** and Guest user :


Autorisations pour http://+:80/

Sécurité

Noms de groupes ou d'utilisateurs :

Nico (NICO-PC\NICO)

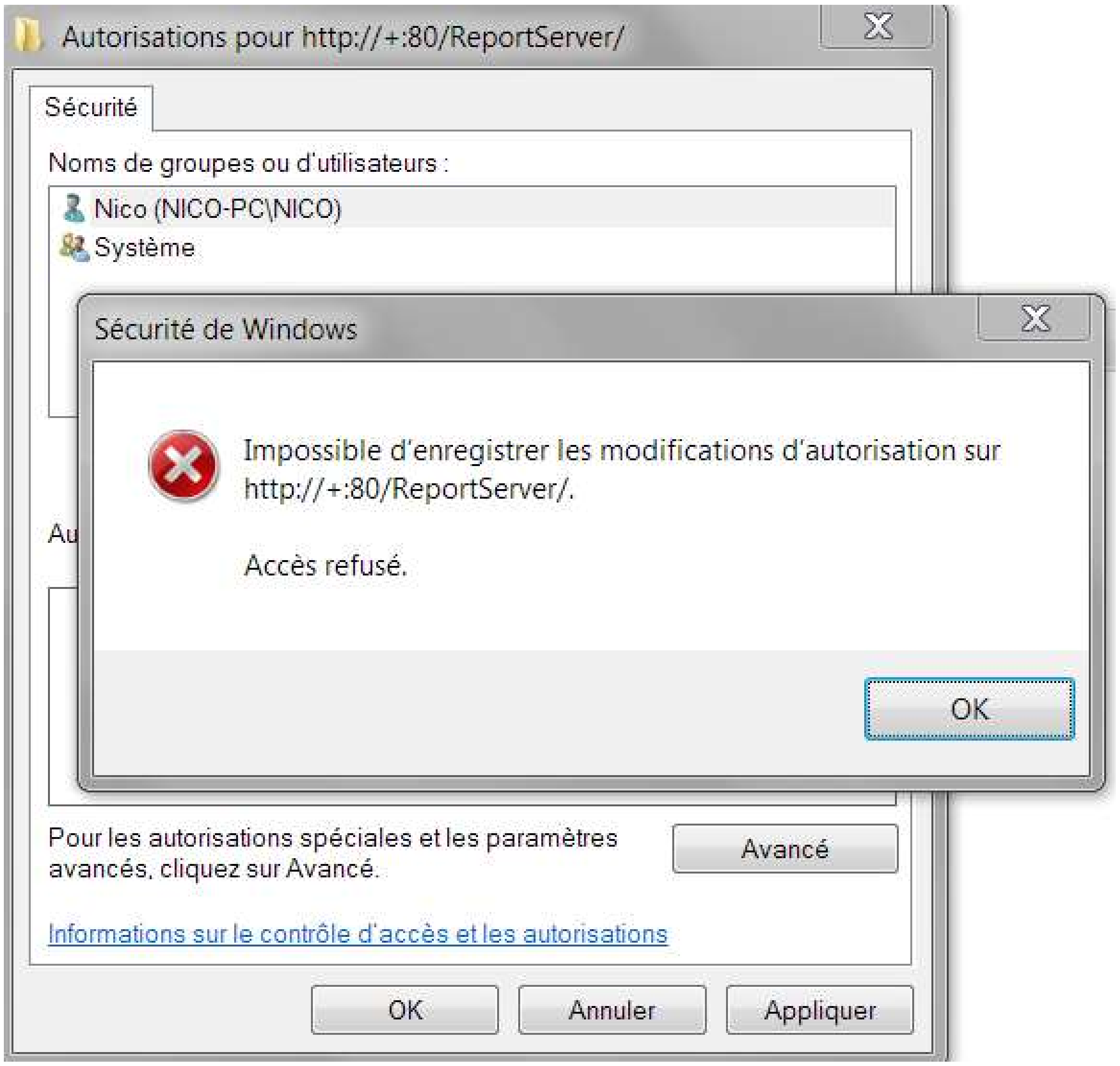
Guest (NICO-PC\Guest)

Autorisations pour Guest

	Autoriser	Refuser
Register	✓	
Delegate		
Autorisations spéciales		

Pour les autorisations spéciales et les paramètres avancés, cliquez sur Avancé.

If I run **HttpSysManager** as Guest and try to add the user NICO to **http://+:80/ReportServer**'s acl, here is the result :


Autorisations pour http://+:80/ReportServer/

Sécurité

Noms de groupes ou d'utilisateurs :

Nico (NICO-PC\NICO)

Système

Sécurité de Windows

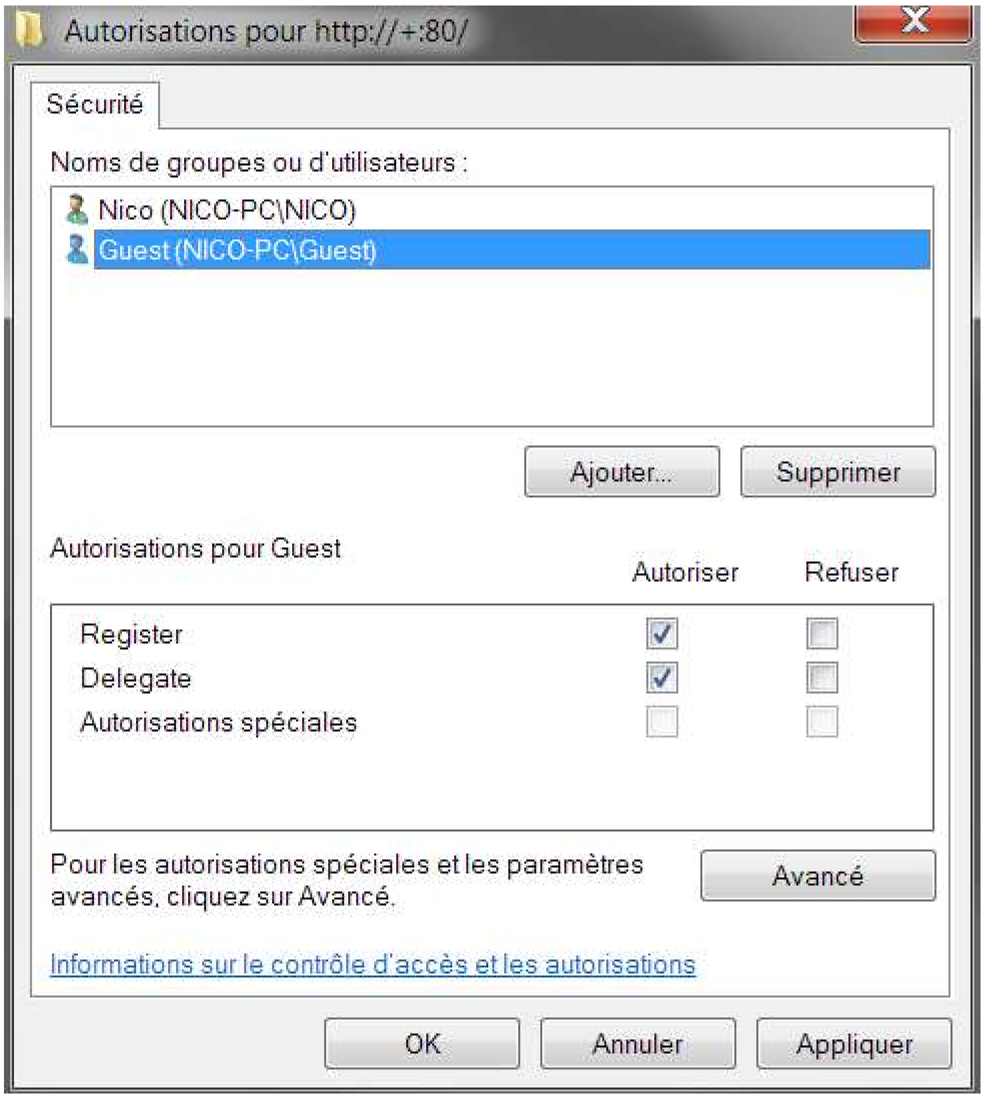
Impossible d'enregistrer les modifications d'autorisation sur http://+:80/ReportServer/.

Accès refusé.

OK
Au
Pour les autorisations spéciales et les paramètres avancés, cliquez sur Avancé.
Avancé
Informations sur le contrôle d'accès et les autorisations
OK
Annuler
Appliquer

In plain english : it does not like it, and I can say that's the best case... My application did not like and crashed !   " src="http://www.codeproject.com/script/Forums/Images/smiley_smile.gif" />

Go back in admin user, and give delegation rights to Guest on **http://+:80/**:


Autorisations pour http://+:80/

Sécurité

Noms de groupes ou d'utilisateurs :

- Nico (NICO-PC\NICO)
- Guest (NICO-PC\Guest)

Autorisations pour Guest

	Autoriser	Refuser
Register	✓	
Delegate	✓	
Autorisations spéciales		

Pour les autorisations spéciales et les paramètres avancés, cliquez sur Avancé.

Informations sur le contrôle d'accès et les autorisations

Run again in Guest user, and now you can add Nico to acl of **http://+:80/ReportServer**.

The case of HTTPS

`http.sys` can listen both, http and https traffic. And it ships with some really nice feature specific for https.

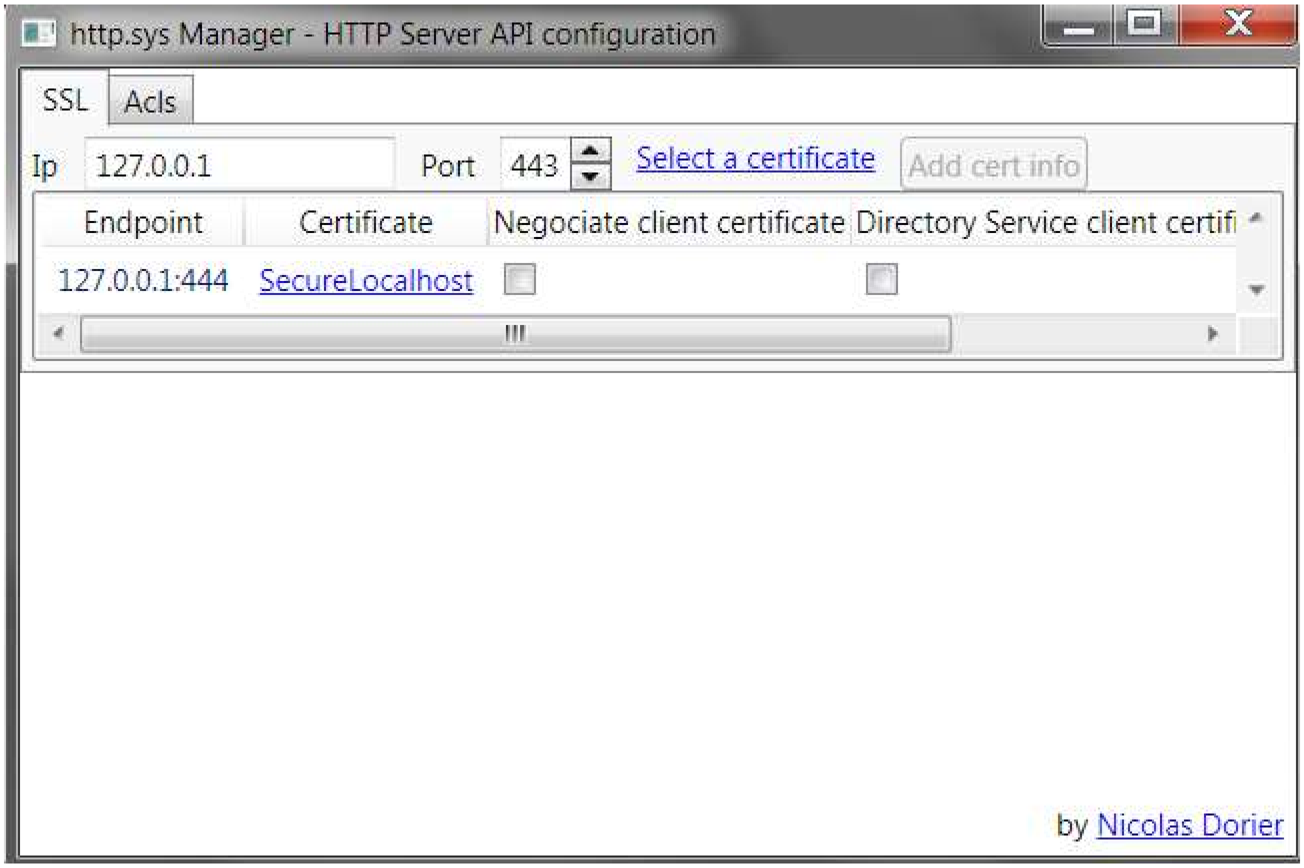
Similar to the HTTP case, IIS, WCF and the world relies on HTTP Server API for HTTPS. (except people that just use good old socket)
The most obvious feature that http.sys expose is the possibility to bind a certificate to an endpoint.(IP:Port)

The certificate should be installed to the LocalMachine certificate location, because http.sys does not run in the context of any user.

Then you can use `netsh http add|delete|show sslcert certhash=` to bind the endpoint to the server certificate, but **HttpSysManager** makes it easier.

Here is how to import or select the certificate in a store and bind to endpoint with **HttpSysManager**.

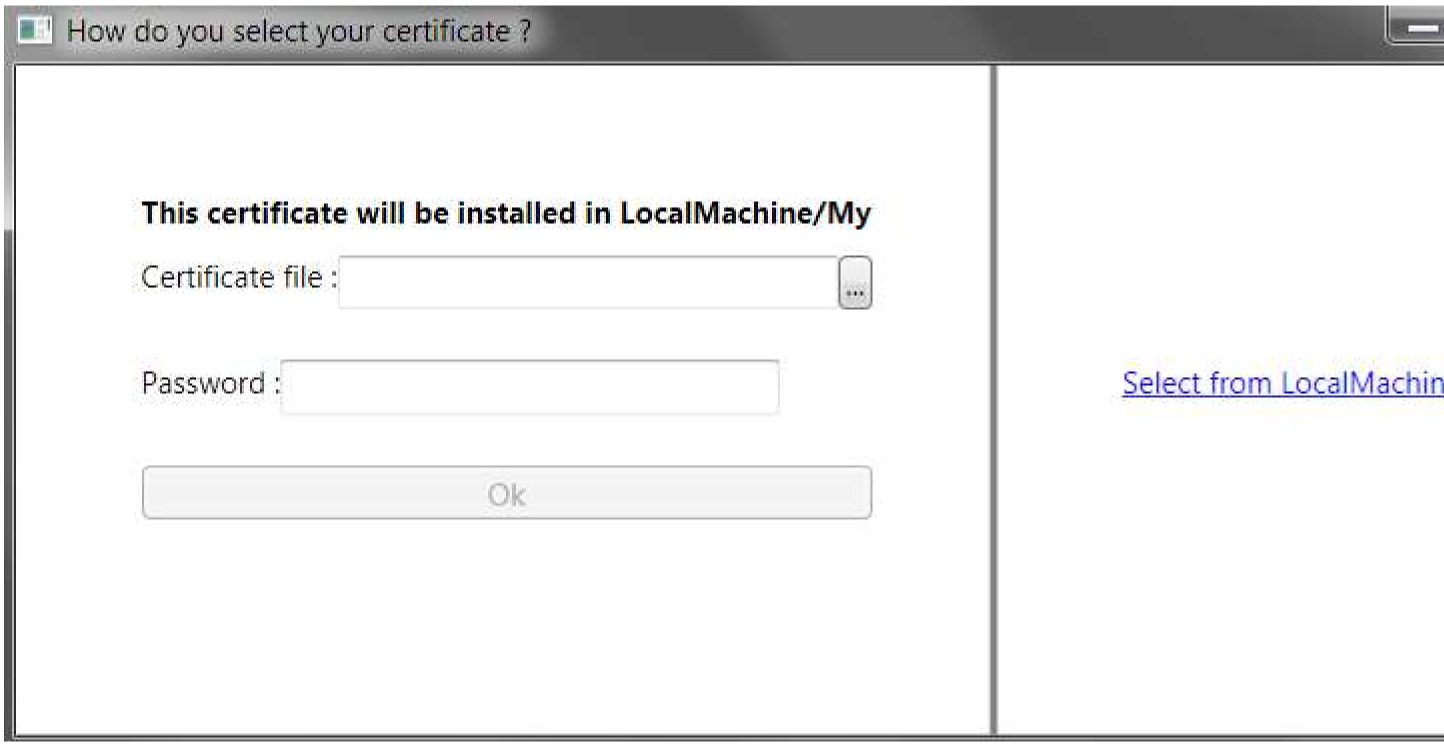
You can see in this screenshot an ssl binding already registered by IIS on port 444, and I am adding a new one on port 443.



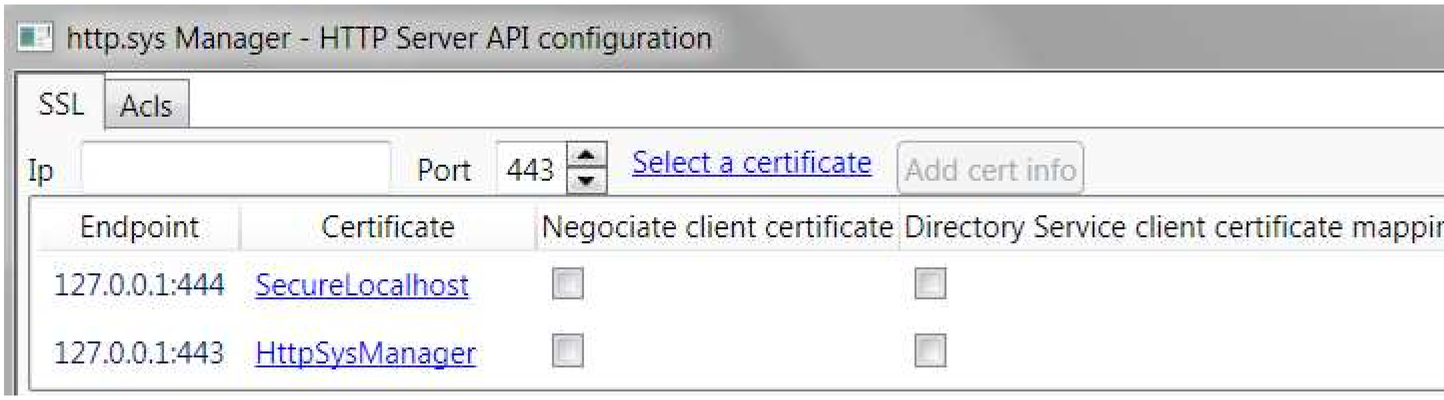
Ip: 127.0.0.1
Port: 443

Endpoint	Certificate	Negociate client certificate	Directory Service client certifi
127.0.0.1:444	SecureLocalhost	☐	☐

Click on **Select certificate**, and choose your certificate (with private key) from file or from your machine's store.


How do you select your certificate ?
This certificate will be installed in LocalMachine/My
Certificate file :
...
Password :
Ok
Select from LocalMachin

And that's done, now all https traffic from this endpoint will use the certificate HttpSysManager to authenticate and encryption.



Endpoint	Certificate	Negociate client certificate	Directory Service client certificate mappin
127.0.0.1:444	SecureLocalhost	☐	☐
127.0.0.1:443	HttpSysManager	☐	☐

A note on the **Negotiate client certificate** checkbox :

Once an ssl session is established, the application can choose exactly when to ask client's certificate for mutual authentication.
For example, you can configure IIS to require client certificates only when they are accessing the virtual folder Secure_Data...

At this point, the server and client negociate a new SSL session (**re**negociation) and the client sends its certificate.

All is well, except in some case, some of your clients might not support to **re**negociate the SSL session.

In this case, you can ask http.sys to always require the client to send certificate when an SSL session is establishing by checking **Negociate Client Certificate**.

A note for the second option : **Directory Service client certificate Mapping**

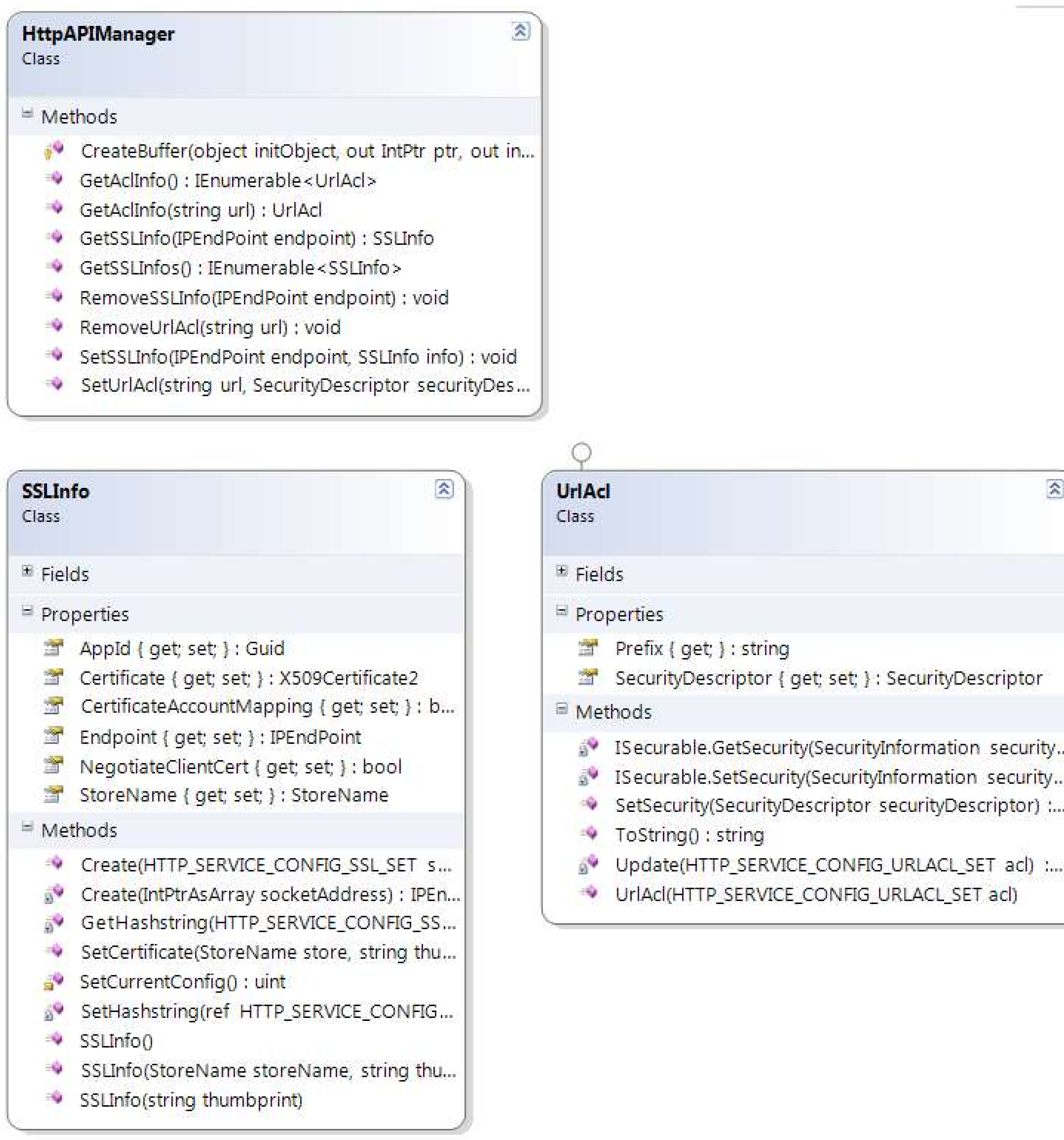
If checked, http.sys will ask the AD to give the access token (Windows identity) corresponding to the client certificate, and will pass it to the application. (For impersonation, or authorization purpose)

For more information on that, check this link.

Show me the code

The code makes heavy use on pInvoke to interact with the HTTP Server API. For the Url Acl part I was lucky enough to find a code snippet on google that wrapped all `SecurityDescriptor` stuff from Process Hacker V1.
I end up referencing all the library, and modifying the sources to add some methods.

The high level class that wrap the HTTP Server API is `HttpApiManager` along with `UrlAcl` and `SSLInfo`.



```
HttpAPIManager
Class

Methods
  CreateBuffer(object initObject, out IntPtr ptr, out in...
  GetAclInfo() : IEnumerable<UrlAcl>
  GetAclInfo(string url) : UrlAcl
  GetSSLInfo(IPEndPoint endpoint) : SSLInfo
  GetSSLInfos() : IEnumerable<SSLInfo>
  RemoveSSLInfo(IPEndPoint endpoint) : void
  RemoveUrlAcl(string url) : void
  SetSSLInfo(IPEndPoint endpoint, SSLInfo info) : void
  SetUrlAcl(string url, SecurityDescriptor securityDes...
```

```
SSLInfo
Class

Fields
Properties
  AppId { get; set; } : Guid
  Certificate { get; set; } : X509Certificate2
  CertificateAccountMapping { get; set; } : b...
  Endpoint { get; set; } : IPEndPoint
  NegotiateClientCert { get; set; } : bool
  StoreName { get; set; } : StoreName
Methods
  Create(HTTP_SERVICE_CONFIG_SSL_SET s...
  Create(IntPtrAsArray socketAddress) : IPEn...
  GetHashstring(HTTP_SERVICE_CONFIG_SS...
  SetCertificate(StoreName store, string thu...
  SetCurrentConfig() : uint
  SetHashstring(ref HTTP_SERVICE_CONFIG...
  SSLInfo()
  SSLInfo(StoreName storeName, string thu...
  SSLInfo(string thumbprint)
```

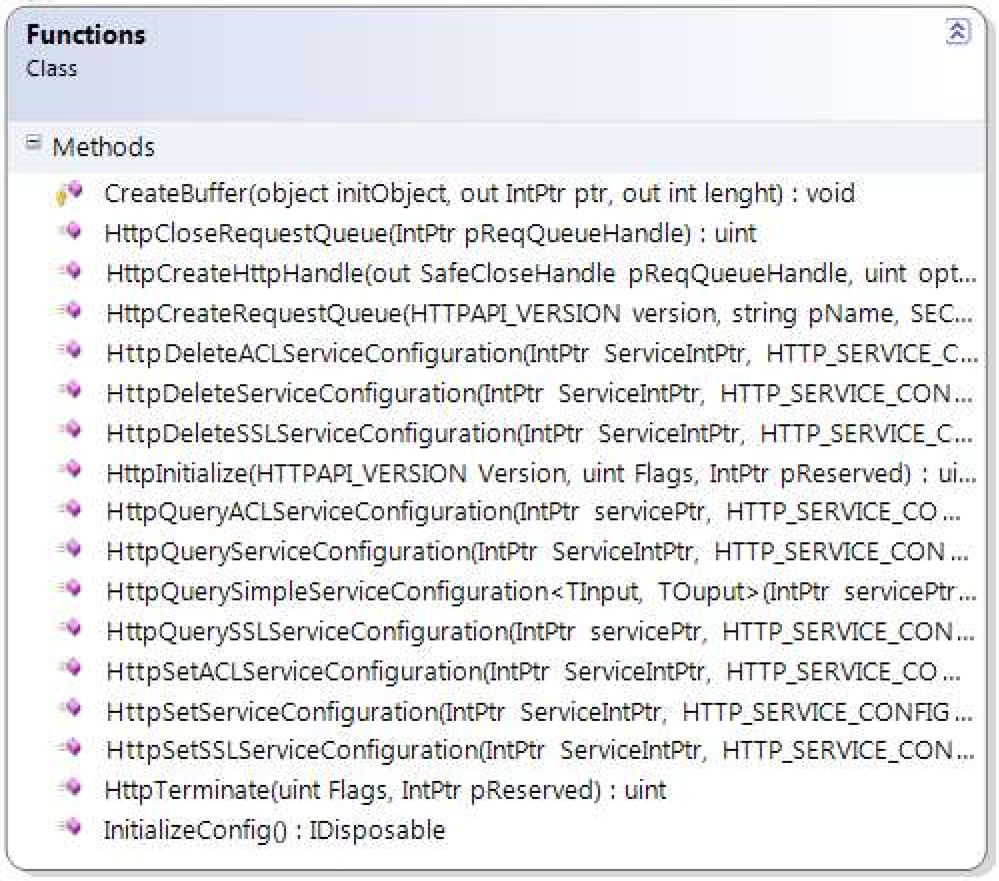
```
UrlAcl
Class

Fields
Properties
  Prefix { get; } : string
  SecurityDescriptor { get; set; } : SecurityDescriptor
Methods
  ISecurable.GetSecurity(SecurityInformation security..
  ISecurable.SetSecurity(SecurityInformation security..
  SetSecurity(SecurityDescriptor securityDescriptor) :...
  ToString() : string
  Update(HTTP_SERVICE_CONFIG_URLACL_SET acl) :...
  UrlAcl(HTTP_SERVICE_CONFIG_URLACL_SET acl)
```

No surprise on this part, properties of these object reflects what you see in the UI.

Note that `UrlAcl.SecurityDescriptor`, hold informations about the ACL,
the `SecurityDescriptor` is a beat on itself... I borrowed the class created by Process Hacker V1

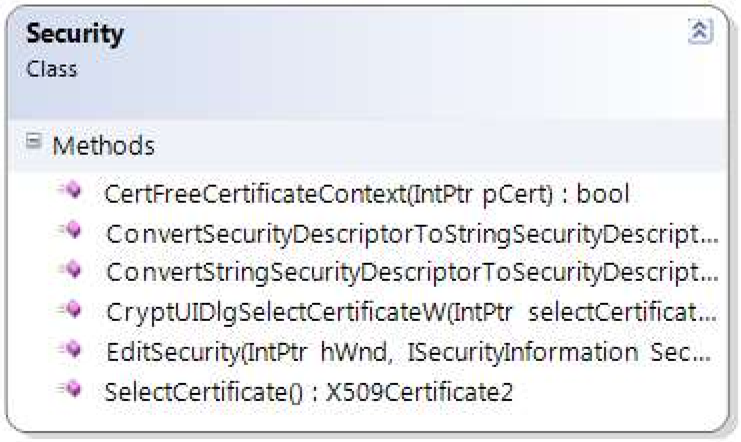
The HttpApiManager just depends on tons and tons and tons of pInvoke wrapper functions around Win API.



Functions
Class

Methods

```
CreateBuffer(object initObject, out IntPtr ptr, out int lenght) : void
HttpCloseRequestQueue(IntPtr pReqQueueHandle) : uint
HttpCreateHttpHandle(out SafeCloseHandle pReqQueueHandle, uint opt...
HttpCreateRequestQueue(HTTPAPI_VERSION version, string pName, SEC...
HttpDeleteACLServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_C...
HttpDeleteServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_CON...
HttpDeleteSSLServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_C...
HttpInitialize(HTTPAPI_VERSION Version, uint Flags, IntPtr pReserved) : ui...
HttpQueryACLServiceConfiguration(IntPtr servicePtr, HTTP_SERVICE_CO...
HttpQueryServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_CON...
HttpQuerySimpleServiceConfiguration<TInput, TOuput>(IntPtr servicePtr...
HttpQuerySSLServiceConfiguration(IntPtr servicePtr, HTTP_SERVICE_CON...
HttpSetACLServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_CO...
HttpSetServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_CONFIG...
HttpSetSSLServiceConfiguration(IntPtr ServiceIntPtr, HTTP_SERVICE_CON...
HttpTerminate(uint Flags, IntPtr pReserved) : uint
InitializeConfig() : IDisposable
```



Security
Class

Methods

```
CertFreeCertificateContext(IntPtr pCert) : bool
ConvertSecurityDescriptorToStringSecurityDescript...
ConvertStringSecurityDescriptorToSecurityDescript...
CryptUIDlgSelectCertificateW(IntPtr selectCertificat...
EditSecurity(IntPtr hWnd, ISecurityInformation Sec...
SelectCertificate() : X509Certificate2
```

I pass the details, it is not very interesting and was a pain in the ass to code.

The interesting part is how I reused windows UI for my puropse again, for example, for Url ACL, I reused Process Hacker V1 classes, especially the `SecurityEditor.EditSecurity` method.

Hide Copy Code

```
public static void EditSecurity(IWin32Window owner, ISecurable securable, string
name, IEnumerable<AccessEntry> accessEntries);
```

Hide Copy Code

```
private void Permissions_Click(object sender, RoutedEventArgs e)
{
        var acl = (UrlAcl)((FrameworkElement)sender).DataContext;
        SecurityEditor.EditSecurity(null, acl, acl.Prefix, GetAccessEntries());
}
```

The `ISecurable` interface are callback functions to implement that the ACL window use to get current ACLs, and set new ACLs. That is a fully managed one from Process Hacker that I implemented on `UrlAcl`...

Here is what I would have to implement without Process Hacker V1 :

Hide Copy Code

```
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
[Guid("965fc360-16ff-11d0-91cb-00aa00bbb723")]
public interface ISecurityInformation
{
        HResult GetAccessRights(ref Guid ObjectType, SiObjectInfoFlags Flags, out
IntPtr Access, out int Accesses, out int DefaultAccess);
        HResult GetInheritTypes(out IntPtr InheritTypes, out int InheritTypesCount);
        HResult GetObjectInformation(out SiObjectInfo ObjectInfo);
        HResult GetSecurity(SecurityInformation RequestedInformation, out IntPtr
SecurityDescriptor, bool Default);
        HResult MapGeneric(ref Guid ObjectType, ref AceFlags AceFlags, ref int
Mask);
        HResult PropertySheetPageCallback(IntPtr hWnd, SiCallbackMessage Msg,
SiPageType Page);
        HResult SetSecurity(SecurityInformation SecurityInformation, IntPtr
SecurityDescriptor);
}
```

I am certainly confident with my coder's skill, it is feasable, but I am not masochist.
All the classes in this interface are also composed of IntPtr... It is not plain old C# object, but plain old C object.

Here is what I needed to implement thanks to Process Hacker.

Hide Copy Code

```
public interface ISecurable
{
        SecurityDescriptor GetSecurity(SecurityInformation securityInformation);
        void SetSecurity(SecurityInformation securityInformation, SecurityDescriptor
securityDescriptor);
}
```

And that's all plain old .NET object my friend !

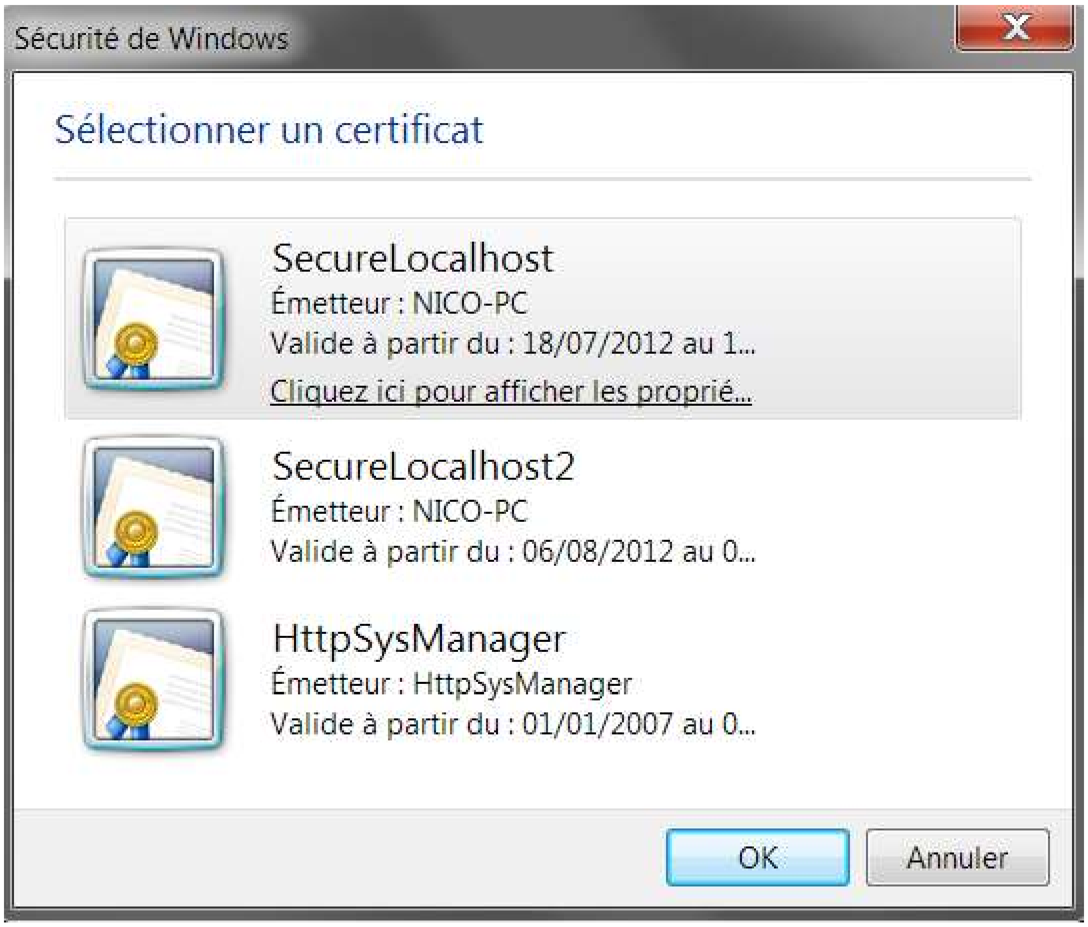
Here is the implementation in `UrlAcl`:

Hide   Copy Code

```
SecurityDescriptor
ISecurable.GetSecurity(ProcessHacker.Native.Api.SecurityInformation
securityInformation)
{
        return SecurityDescriptor;
}

public void SetSecurity(SecurityDescriptor securityDescriptor)
{
        ((ISecurable)this).SetSecurity(SecurityInformation.Dacl,
securityDescriptor);
}
void ISecurable.SetSecurity(ProcessHacker.Native.Api.SecurityInformation
securityInformation, SecurityDescriptor securityDescriptor)
{
        if(securityDescriptor.ToString() == SecurityDescriptor.Empty.ToString())
                return;
        var manager = new HttpAPIManager();
        manager.SetUrlAcl(Prefix, securityDescriptor);
        Update(manager.GetAclInfo(Prefix)._Acl);
}
```

The other interesting part is how I reused the certificate selection windows :



Sécurité de Windows

Sélectionner un certificat

SecureLocalhost
Émetteur : NICO-PC
Valide à partir du : 18/07/2012 au 1...
Cliquez ici pour afficher les proprié...

SecureLocalhost2
Émetteur : NICO-PC
Valide à partir du : 06/08/2012 au 0...

HttpSysManager
Émetteur : HttpSysManager
Valide à partir du : 01/01/2007 au 0...

This one was also simple, .NET provide a full manager wrapper around certificates structures : `X509Certificate2`.

I created a method : `Security.SelectCertificate()`, to show certificates from the LocalMachine/My store.

Take a look at how I use `X509Store` and `X509Certificate2`, to get informations that the unmanaged function `Security.CryptUIDlgSelectCertificateW(intputPtr);` uses.
I commented the interesting parts.

Hide Shrink ▲ Copy Code

```
public static X509Certificate2 SelectCertificate()
{
        bool storeOpened = false;
        var store = new X509Store(StoreName.My, StoreLocation.LocalMachine);
        IntPtr result = IntPtr.Zero;
        IntPtr intputPtr = IntPtr.Zero;
        var array = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(IntPtr)));
        try
        {
                store.Open(OpenFlags.ReadOnly);
                storeOpened = true;
                Marshal.WriteIntPtr(array, store.StoreHandle); //array hold the list
of Store to select from. I use X509Store.StoreHandle to get the handle that the
unmanaged function need.


                int size =
Marshal.SizeOf(typeof(CRYPTUI_SELECTCERTIFICATE_STRUCTW));
                CRYPTUI_SELECTCERTIFICATE_STRUCTW input = new
CRYPTUI_SELECTCERTIFICATE_STRUCTW();

                input.dwSize = (uint)size;
                input.cPropSheetPages = 0;
                input.dwDontUseColumn = 0;
                input.dwFlags = 0;
                input.hSelectedCertStore = IntPtr.Zero;
                input.hwndParent = IntPtr.Zero;
                input.pDisplayCallback = IntPtr.Zero;
                input.pFilterCallback = IntPtr.Zero;
                input.pvCallbackData = IntPtr.Zero;
                input.rghDisplayStores = array;
                input.cDisplayStores = 1;
                input.rghStores = IntPtr.Zero;
                input.cStores = 0;
                input.rgPropSheetPages = IntPtr.Zero;
                input.szDisplayString = null;
                input.szTitle = null;


                intputPtr = Marshal.AllocHGlobal(size);
                Marshal.StructureToPtr(input, intputPtr, false);
                result = Security.CryptUIDlgSelectCertificateW(intputPtr);
                if(result == IntPtr.Zero)
                        return null;
                return new X509Certificate2(result); //I create X509Certificate2
from the handle I got from Security.CryptUIDlgSelectCertificateW
        }
        finally
        {
                if(storeOpened)
                        store.Close();
                if(intputPtr != IntPtr.Zero)
                {
                        Marshal.DestroyStructure(intputPtr,
typeof(CRYPTUI_SELECTCERTIFICATE_STRUCTW));
                        Marshal.FreeHGlobal(intputPtr);
                }
                if(array != IntPtr.Zero)
                        Marshal.FreeHGlobal(array);
                if(result != IntPtr.Zero)
                        Security.CertFreeCertificateContext(result);
//X509Certificate2 duplicate the handle internally
        }
}
}
```

Roadmap

In fact, there is much more possibility with the HTTP Server API, and here are some of the next steps I will implement.

- script (.bat) generation
- ~~CTL (Certificate Trust List) Creation and binding to SSL endpoint~~
- HTTPS SNI (only windows 8)
- Support other store than MY