Synopsis: Describes how Windows Authentication works in the context of eB Web Server and how to configure it.

Number: D006964
Updated: 27/03/2014
By: Joe Buckley, Bentley Systems
State: WIP
Scope: **eB v16**

Windows Authentication with eB Web Server

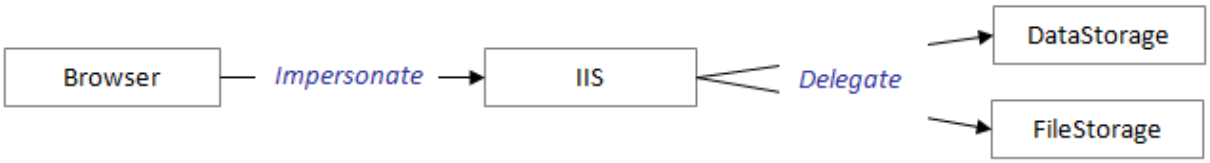
1. Overview

This document will not explain how Kerberos or AD works; simply how to setup eB so Windows authentication works across separate Web and App Servers. I do however need to point out two important concepts (1.1 and 1.2 below).

Wiki: You have to use Kerberos in order to do delegation and you have to use delegation if the AppServer is on a different machine from the WebServer.

1.1 Impersonation and Delegation

Impersonation allows the service to act as the client while performing the action. Delegation allows a front-end service to forward the client's request to a back-end service in such a way that the back-end service can also impersonate the client. Impersonation is most commonly used as a way of checking whether a client is authorized to perform a particular action, while delegation is a way of flowing impersonation capabilities, along with the client's identity, to a back-end service.



Delegation is a Windows domain feature that can be used when Kerberos-based authentication is performed. Delegation is distinct from identity flow and, because delegation transfers the ability to impersonate the client without possession of the client's password, it is a much higher privileged operation than identity flow.

Both impersonation and delegation require that the client have a Windows identity. *If a client does not possess a Windows identity, then the only option available is to flow the client's identity to the second service. (e.g. using Basic Authentication see: 3.3)*

1.2 Kerberos vs Ntlm

Windows makes use of both Kerberos and Ntlm for authentication. The default [when on a domain] is to use Kerberos but the authentication can fall back to Ntlm for a variety of reasons such as:

- The client is authenticating to a server using an IP address
- The client is authenticating to a server that belongs to a different Active Directory forest that has a legacy NTLM trust instead of a transitive inter-forest trust
- The client is authenticating to a server that doesn't belong to a domain
- No Active Directory domain exists (commonly referred to as "workgroup" or "peer-to-peer")
- Where a firewall would otherwise restrict the ports required by Kerberos (typically TCP 88)

Long story short; Ntlm as a fallback makes debugging difficult, this is aggravated by that fact that the error message that that one gets back from an incorrect configuration is misleading and uninformative. I will refer to this error as the "*PerplexedError*".

1.3 The "*PerplexedError*"

Whenever anything goes wrong between the WebServer and the AppServer as a result of delegation the error that comes back invariably says:

```
System.ServiceModel.CommunicationException: The socket connection was aborted. This could be caused by an
error processing your message or a receive timeout being exceeded by the remote host, or an underlying
network resource issue. Local socket timeout was '00:01:00'.

---> System.Net.Sockets.SocketException: An existing connection was forcibly closed by the remote host
   at System.Net.Sockets.Socket.Receive(Byte[] buffer, Int32 offset, Int32 size, SocketFlags socketFlags)
   at System.ServiceModel.Channels.SocketConnection.ReadCore(Byte[] buffer, Int32 offset, Int32 size,
TimeSpan timeout, Boolean closing)
   --- End of inner exception stack trace ---
   at System.ServiceModel.Channels.SocketConnection.ReadCore(Byte[] buffer, Int32 offset, Int32 size,
TimeSpan timeout, Boolean closing)
   at System.ServiceModel.Channels.SocketConnection.Read(Byte[] buffer, Int32 offset, Int32 size,
TimeSpan timeout)
   at System.ServiceModel.Channels.DelegatingConnection.Read(Byte[] buffer, Int32 offset, Int32 size,
TimeSpan timeout)
   at System.ServiceModel.Channels.ConnectionStream.Read(Byte[] buffer, Int32 offset, Int32 count,
TimeSpan timeout)
   at System.ServiceModel.Channels.ConnectionStream.Read(Byte[] buffer, Int32 offset, Int32 count)
   at System.Net.FixedSizeReader.ReadPacket(Byte[] buffer, Int32 offset, Int32 count)
   at System.Net.Security.NegotiateStream.StartFrameHeader(Byte[] buffer, Int32 offset, Int32 count,
AsyncProtocolRequest asyncRequest)
   at System.Net.Security.NegotiateStream.StartReading(Byte[] buffer, Int32 offset, Int32 count,
AsyncProtocolRequest asyncRequest)
   at System.Net.Security.NegotiateStream.ProcessRead(Byte[] buffer, Int32 offset, Int32 count,
AsyncProtocolRequest asyncRequest)
   --- End of inner exception stack trace ---
   at System.Net.Security.NegotiateStream.ProcessRead(Byte[] buffer, Int32 offset, Int32 count,
AsyncProtocolRequest asyncRequest)
   at System.Net.Security.NegotiateStream.Read(Byte[] buffer, Int32 offset, Int32 count)
   at System.ServiceModel.Channels.StreamConnection.Read(Byte[] buffer, Int32 offset, Int32 size,
TimeSpan timeout)
```

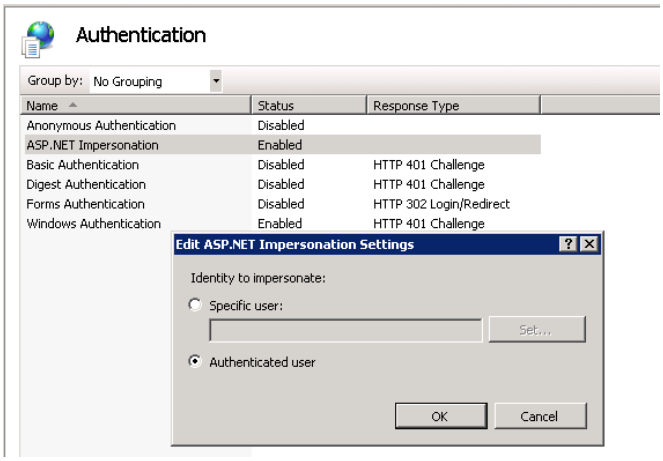
The giveaway is that it happens immediately and not after the 1 minute timeout. Even so it is not very helpful so the best way to debug is to enable Kerberos tracing on the IIS server.

Unfortunately, because it is so generic, we cannot really translate this message into anything useful to the user.

2. Configuring the Web Server

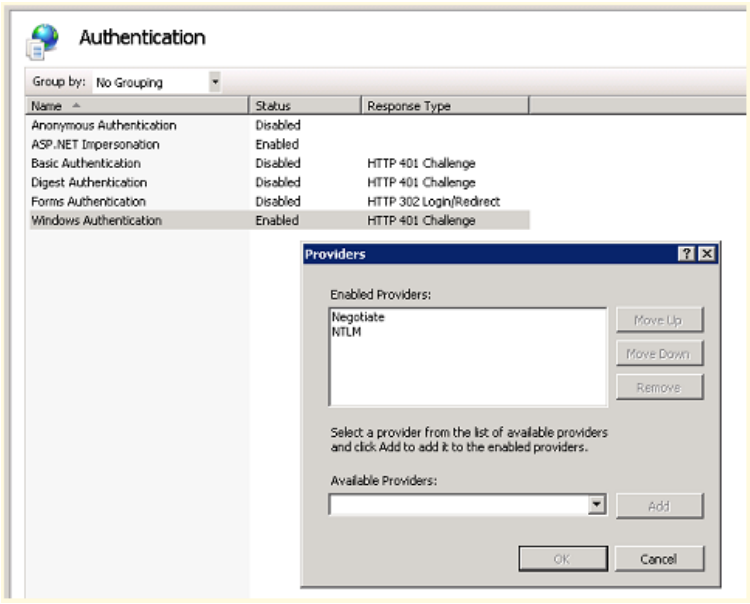
2.1 Configure ASP.NET Impersonation

On the web application under ""Authentication"; enable "ASP.NET Impersonation" and make sure that the setting is for the "Authenticated User" e.g:



2.2 Configure Windows Authentication

Also under ""Authentication", Disable all authentication except for "Windows Authentication" (we obviously also keep the ASP.NET Impersonation enabled). The default is to have both the Negotiate and NTLM providers available. You may keep those defaults even though we will not use the NTLM part.



Note, we also keep the default "Kernel Mode" (under advanced settings)

2.3 Disable Ntlm to AppServer [suggestion]

Although this is not required, it does simplify debugging. The Negotiate protocol will pick Kerberos if all mutual trust requirements are satisfied but if not it will silently failover to Ntlm and you will get the *PerplexedError.*

To disable it, add the following to the web.config file (under appSettings):

```
<appSettings>
   <add key="Bentley.eB.AllowNtlm" value="false"/>
</appSettings>
```

2.4 Disable Https to AppServer [suggestion]

All eB clients support both Tcp and Https; the Tcp channel uses Windows credentials on the transport as default security. If anything prevents communication over Tcp, the eB client will fail-over to the https channel. This behavior can mask some errors and since we know that we need Tcp with windows credentials for our delegation scenario it simplified debugging it we turn of Https. To do this, add the following to the web.config file (under appSettings):

```
<appSettings>
   <add key="Bentley.eB.TcpOnlyDomains" value="*"/>
</appSettings>
```

This means; use Tcp when connecting to any server.

2.5 Upn, when Application Services is running as a user account

If the application server is running as a specific user account (i.e. bob@acme.com) then you have to set the User Principal Name as the client endpoint identity. To do this add the following to the web.config file (under appSettings):

```
<appSettings>
     <add key="Bentley.eB.DefaultTcpEndpointIdentity" value="upn:bob@acme.com"/>
</appSettings>
```

Note: to find the identity of the service you can look at the Wsdl file for the service e.g. type this in the browser: https://<server>/Bentley/eB/Service/ServiceRegistry?wsdl and you will see towards the bottom of the results:

```
<wsdl:port name="NetTcpBinding_IServiceRegistry" binding="tns:NetTcpBinding_IServiceRegistry">
  <soap12:address location="net.tcp://localhost:18377/Bentley/eB/Service/ServiceRegistry/Tcp" />
  <wsa10:EndpointReference>
     <wsa10:Address>net.tcp://localhost:18377/Bentley/eB/Service/ServiceRegistry/Tcp</wsa10:Address>
     <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
        <Upn>bob@acme.com</Upn>
     </Identity>
   </wsa10:EndpointReference>
</wsdl:port>
```

2.6 Spn, when Application Services is running as a machine account

If the application server is running under a machine account (like Local System or Network Services) then you need to set the identity to a Service Principal Name or explicitly set it to 'None' – because None will default to the host name. E.g.:

```
<appSettings>
     <add key="Bentley.eB.DefaultTcpEndpointIdentity" value="spn:HOST/myservername"/>
</appSettings>
```

Or

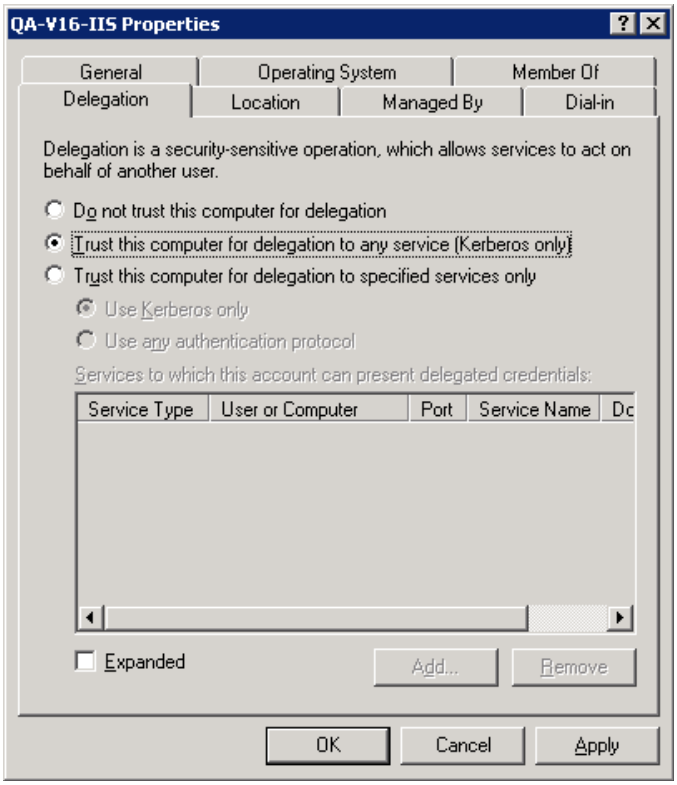
```
<appSettings>
     <add key="Bentley.eB.DefaultTcpEndpointIdentity" value="None"/>
</appSettings>
```

You can verify that the spn is correct by looking at AD like this:

```
C:\> setspn –L myservername
```

2.7 Set the “Trust for Delegation” on the IIS AppPool account

The IIS AppPool account is, by default, associated with the machine account. E.g. the DefaultAppPool on “MyIISServer” will use the spn “HOST/MyIISServer” to request the Kerberos ticket for the AppServer. In order for this to succeed the spn account i.e. the MyIISServer machine account required the “Trust this computer for delegation to any service (Kerberos only)” permission. You set this up on the AD using the “Active Directory Users and Computers” mmc snap in. e.g.

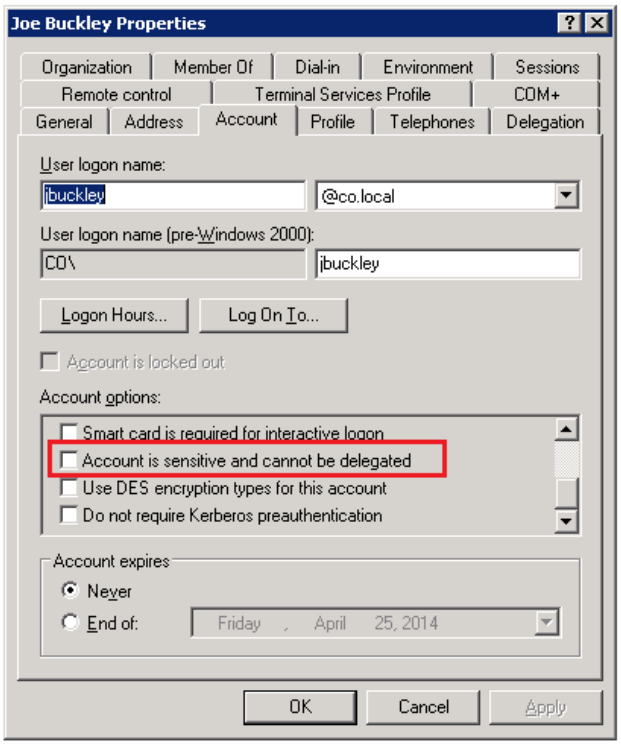


Please note that this setting is cached in the client token – so the client [using the browser] will need to purge their Kerberos tickets or [easier] logoff and back in again. Also note that this setting will have to be replicated between domain controllers – so there could be a 15 min delay before it is available everywhere.

If this is not the case the user will get the “*PerplexedError”.*

2.8 Verify that all the users are trusted for delegation

All users must be allowed to have their credentials delegated. This is true by default, but it will be worthwhile to verify that this is the case by checking in “Active Directory Users and Computers” that the setting “Account is sensitive and cannot be delegated” isn’t set. E.g.



If this is not the case the user will get the "*PerplexedError*".

2.9 When using a specific user account for the IIS AppPool

If the using a specific user account for the IIS AppPool then you also need to assign a specific HTTP/MyIISServer spn to the user account. Without this IIS will not be able to decrypt the Kerberos ticket. To add the spn you would do this e.g:

```
setspn -A HTTP/<bios name>  <domain>\<username>
```

You also need to set the SPN for the FQDN\

```
setspn -A HTTP/<fqdn>  <domain>\<username>
```

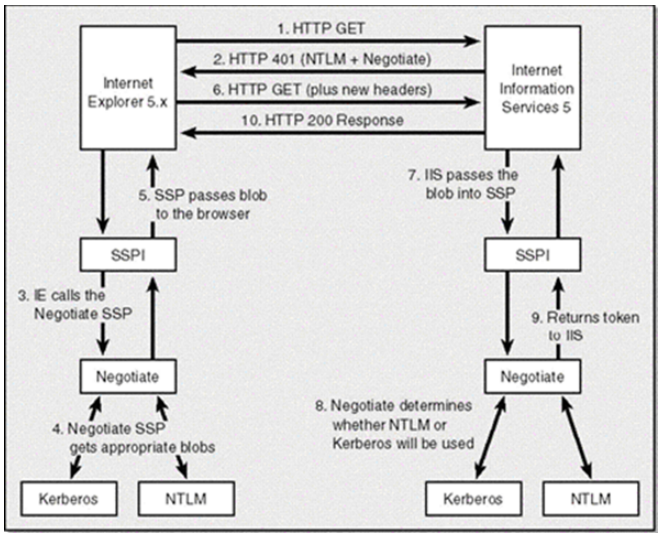
Also make sure that this user is trusted for delegation (similar to 2.7).

Again, if the spn is invalid or the user isn't trusted for delegation you will get the "*PerplexedError*".

3. Current limitation and future work

3.1 Limited to a single domain

When a client from one domain [say A] logs into an IIS server from another [say B], then both domains require "transitive inter-forest trust" before delegation will work. (You will get the "*PerplexedError" if this is not the case).* The reason for this is because the browser will talk to the TGS on one domain and the IIS will talk to a TGS on another e.g:



PLEASE NOTE: I have not verified cross domain delegation – I simply don't have the infrastructure to do this

3.2 WS-Trust i.e. ADFS and IMS

We will be moving to WS-trust in future (instead of relying on the SSPI). We already support delegation based on 'ActAs' channels using tokens obtained from IMS. This will work the same for ADFS.

3.3 Using Basic Authentication

For cross domain authentication one can consider using basic authentication. The credentials will be passed along so IIS can flow those to the AppServer. Keep in mind that the credentials will be in plaintext – so you will need to ensure that the WebServer uses Https. Also, you cannot mix Basic Authentication with Windows Authentication within the same web app.

3.4 S4U-Based Impersonation

There is no support for S4U-Based Impersonation yet. We will be looking into that in conjunction with ADFS.

4. Appendix

4.1 Revision History

This is the original revision

4.2 References

- D005091 - Communication Overview

External

- http://en.wikipedia.org/wiki/Kerberos_(protocol)

- http://en.wikipedia.org/wiki/NT_LAN_Manager
- http://blogs.msdn.com/b/spatdsg/archive/2007/11/14/kerberos-delegation-end-to-end-part-i.aspx
- http://blogs.msdn.com/b/spatdsg/archive/2007/11/20/kerberos-delegation-end-to-end-part-ii.aspx
- http://blogs.msdn.com/b/spatdsg/archive/2007/11/26/kerb-part-3.aspx