

# 85010014 error – resolved – ActiveSync Exchange Server

---

*Source:*

<http://www.tech-archive.net/Archive/PocketPC/microsoft.public.pocketpc.activesync/2006-02/msg00256.html>

---

- *From:* "Ryan Hardin" <[RyanHardin@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:RyanHardin@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 9 Feb 2006 01:01:26 -0800
- 

I have a working solution.

My desire to synch successfully 100% of the time became very frustrating. ASync would work for about 2 hours, then quit for 2 hours, then start again for 2 hours...sometimes longer. Off and on. And when it didn't work, the Application Log on our server would be filled with those ActiveSync 3005: HTTP 400 errors...then they would disappear. On my phone (WM5.0), the error was 85010014 (problem on Server), and after seeing that number many times, memorization of that number so I could google the Internet for new insight of the problem was a breeze. But still, no one seemed to have a solution other than what was already in the MSKB, which is still lacking any results of the search. It basically all comes down to the basics of Windows DNS, how SSL technology works, and how having multiple IPs on your Exchange server (and subsequently DNS Host records) can jack things up for accurate synching. I'm still ecstatic as to my solution. EVERY SINGLE SYNC has been successful since my change. I have not received any 85010014 errors or 3005 errors in the Application Log since.

The solution that worked for me (granted that you still read and apply both KB# 817379 and KB 886346):

On Exchange Server, go to C:/windows/system32/drivers/etc/hosts.

Add Public IP and FQDN of your Exchange server (the Windows server name + domain name, NOT the Internet domain)

e.g. 1.1.1.1 server.contoso.local

Why Hosts file and not DNS?

Editing DNS to remove A records for server.contoso.local that didn't match the Default Website IP address didn't work because Windows would change the DNS information automatically, even when unchecking "Register this connection's address in DNS" on the DNS tab in Advanced TCP/IP Settings. The Hosts file gets priority over DNS, and since the server responds to any IP address that it possesses to perform Windows/Domain/ActiveDirectory/Kerberos functions, I don't think adding this to your Hosts file will cause any harm, since it will still be able to resolve the name to an IP address that responds to necessary services.

More insight and my scenario (read to see if it fits yours):

A mockup of my setup (I'm hiding my info here):

Network Environment:

- First off: Multiple IP addresses on the Server
- Windows Server Name (FQDN): server.contoso.local
- Public Internet Domain Name (for OMA, OWA–HTTPS, RPC–HTTP, ExchAS, etc): secure.contoso.com

System Environment:

- Exchange 2003 SP2
- Enabled FBA OWA (we use Public SSL cert from Thawte for secure.contoso.com)
- Created additional Exchange Virtual Directory: /ExchDAV (see KB 817379)
- Modified registry to have SMTPProxy value for SMTP domain of our default Recipient Policy listed in ESM: contoso.local (see KB 886346)
- In Active Directory, we have multiple SMTP addresses for users, with the Primary SMTP address being firstname.lastname@xxxxxxxxxxxx

Our specific error in the Exchange Server's Event Log is as follows:

Event ID: 3005

Source: Server ActiveSync

Unexpected Exchange mailbox Server error: Server: [server.contoso.local]

User: [firstname.lastname@xxxxxxxxxxxx] HTTP status code: [400]. Verify that the Exchange mailbox Server is working correctly.

The regular intervals of the Application Log entry described above that would occur in a somewhat group of time led me to think that some type of caching was involved, so I then checked out our DNS server. Apparently each IP address listed on the server was entered into our internal DNS server. And from what I understand, Exchange ActiveSyncing requires the server to "find itself" internally (from the HTTP 400 errors that were generated) so it can grab the data from the Exchange system and then send it back out via HTTP(s) to the phone. In doing a query on itself, it was using the FQDN of its own machine name + windows domain name -- server.contoso.local. Since DNS had multiple host record (A) entries for [server] in the contoso.local DNS zone, and our Default Website could only properly respond to ONE IP address (because of how SSL technology works), problems would occur when the DNS cache would resolve server.contoso.local to an IP address that was not the correct IP address for SSL website (or Default Website).

In IIS, I have the following host headers on the Default Website, which contains the /exchange et al. virtual directories.

- A blank host header with port 80. (you MUST leave this one – do not delete)
- A host header for secure.contoso.com for port 80.
- SSL host header using port 443 with our SSL cert.

Note: This probably doesn't apply to anyone many out there, but we have two SSL certs on this machine. So we specify the IP for our Cert here on the Host Headers instead of (All Unassigned). This is a great trick to allow multiple SSL certs and secure websites on your server. Each website in IIS must explicitly have the IP address on the host header values instead of (All Unassigned). Read the bottom of this post for how this all works. It is a

fantastic tip and explanation.

We also have multiple IPs on our server, so it is multihomed. (Can be same NIC or different NICs, doesn't matter):

1.1.1.1 – this is specific to our Default Website in IIS (secure.contoso.com – resolution from Internet DNS Servers)

1.1.1.2 – this is specific to our 2nd SSL Website in IIS (somethingelse.msft.com goes here)

192.168.1.1 – private IP of machine for LAN/VPN sitting inside firewall (we use PAT for ports 80 and 443 to point to this machine)

On our Internal DNS Server on our network, there are multiple entries for the SERVER hostname in the [contoso.local] zone file, one for each IP address on the server as listed above.

server.contoso.local – 1.1.1.1

server.contoso.local – 1.1.1.2

server.contoso.local – 192.168.1.1

This is where the problem lies. Multiple entries for the Exchange Server Windows FQDN in DNS.

We need to consistently and accurately resolve SERVER.CONTOSO.LOCAL to 1.1.1.1, as seen in event log entry. The HTTP 400 error is because the server cannot find the website. Remember, we have multiple SSL certificates and we specify the Default Website to answer to the IP address 1.1.1.1, so it will not answer to (All Unassigned) (that is, if SERVER.CONTOSO.LOCAL resolves to 1.1.1.2 or 192.168.1.1, it will fail). We tried removing the DNS host record, but it would always be regenerated by the server (even after changing the TCP/IP DNS properties to not register its address in DNS.)

So if you have a multihomed server, using private and public IPs on the machine, or even using two IPs on the same NIC, (or oddly have multiple SSL certs on your machine) you may want to try my workaround.

Again, the workaround: The good 'ole /system32/drivers/etc/hosts file on the Exchange Server.

I created an entry for SERVER.CONTOSO.LOCAL to be the IP 1.1.1.1. The server resolved the name correctly, Exchange workings "found themselves," and my Windows Mobile 5.0 Smartphone has been synching 100% ever since.

All the best,

– Ryan Hardin, CCNA, MCSE (MCP ID No: 2529921 –> that's for you, Microsoft.

Look me up and hook a brother up now. I just increased the success of your Exchange ActiveSync users. I know you can definitely afford a little compensation. => I'll buy more of your stock. Or if you are interested in acquisition of my software company, that will work too. Seriously, look and hook me up. Shoot an email.)

For multiple SSL websites on a single box:

=====

Basically you need the following for it to work:

## 85010014 error – resolved – ActiveSync Exchange Server

- 1) Two SSL certs entered into IIS, one for each website
- 2) Two IP addresses on the box (either on multiple NICs or a single NIC, doesn't matter)
- 3) In IIS, you must specify different IP addresses for the host headers used for each website.

The reason for this is because SSL traffic actually encrypts the host headers, so the only way for IIS to know which website to route website requests to is via the IP address information in each packet that surrounds the SSL encryption.

To explain logically...

Basically on the network level, all traffic is destined for an IP address and a port. We all know that. And we all know that for web traffic, the port number is 80. And we all know that you can only have a single application respond to the same IP address and port. So how does IIS allow multiple websites? Simple: Host Headers.

If there are multiple websites in IIS, each responding to (All Unassigned) IP address and port 80, IIS reads and uses host header information at the application layer to know where to route website traffic to the correct website. This is how host headers work, and the host header information is carried in the application layer, which is inside the network routing and port data (that is, inside the destination IP address and inside the destination port in the data packet).

If you were to map out the TCP traffic data, it would look something like this:

```
--> {start packet 1} [ destination IP address ] + [ destination port ] + [ application data (host header + http get + blahblahblah) ] {end packet 1}
```

```
--> {start packet 2} [ destination IP address ] + [ destination port ] + [ application data (host header + http get + blahblahblah) ] {end packet 2} -->
```

However, when using SSL and port 443, the application layer is encrypted. If it is not, then it defeats the purpose of the Secure in SecureSocketsLayer (SSL). This encryption of the application layer also includes the encryption of the host header information. More importantly, this data destined for a secure website can only be decrypted by the website in IIS that holds the SSL certificate assigned TO the secure website. This is how SSL certs work.

So, let's state our scenario again, this time using SSL and port 443 instead of port 80:

If there are multiple websites in IIS, each responding to (All Unassigned) and port 80, IIS then reads the host header information and then reroutes the traffic to the correct website. But wait, how can IIS read the host header information if it is encrypted? It can't. This is a limitation on all web servers that I am aware of, and it really isn't a limitation. It is how the protocol is designed for the application and security of the protocol. Also,

## 85010014 error – resolved – ActiveSync Exchange Server

IIS (or other web servers) are not smart enough to know which SSL cert to use in order to decrypt the host header information, and therefore, it cannot route the traffic using the host headers to the correct website.

So the solution?

In IIS on each website that uses in SSL certificate, you must specify the IP address to which the website in IIS is to respond. That is, instead of using (All Unassigned) on your SSL websites, specify the IP address to which the domain name on the SSL certificate resolves. Using this method, IIS will see the traffic and say "OK. You have this IP. Go here. This website over here is supposed to respond to you." This basically circumvents the need to read the encrypted host header in order to send it to the right website. The IP address specified allows it to be sent to the correct website, without using host headers. So if IIS simply sends the traffic to the website that specifically responds to an IP address, it can then use the Cert to decrypt the data, and now you're rockin'.

Basically, if IIS has multiple websites responding to the same IP and port, it uses Host Headers. Since Host Headers are encrypted by SSL, having multiple SSL websites requires the use of different IP addresses.

This is something that Microsoft should consider in their next version of IIS:

If there are multiple SSL certs on multiple websites in IIS, when a website request arrives on the machine, IIS should query all SSL certificates stored on the websites. IIS should continue down this list trying to decrypt the application data until it can find a match of an unencrypted host header with a host header that is listed on a website stored in IIS. If this were the case, multiple SSL certs in IIS would not require you to have an IP address for each cert. I'm not sure what security risks are imposed, but this seems like a logical thing to me.

– Good luck!  
– Ryan Hardin