We have been seeing and trying to resolve SQL connectivity issue all the time. I guess it would be helpful if we can put some guidance on how to resolve connectivity issues. Here comes a proposal based on my experience.

Basically, when you failed to connect to your SQL Server, the issue could be:

1) Network issue,

2) SQL Server configuration issue.

3) Firewall issue,

4) Client driver issue,

5) Application configuration issue.

6) Authentication and logon issue.

Usually, customers see connectivity issue in their applications, but it would be great if you can follow the steps below to eliminate issues one by one and post a question on [SQL Server Data Access](http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=87&SiteID=1) forum if needed.

**Step 1: Network issue**

You might be able to make local connection without a working network, but that's a special case. For remote connection, a stable network is required. The first thing to trouble shoot SQL connectivity issues is to make sure the network we rely on is workable and stable. Please run the following commands:

**ping -a <your\_target\_machine>**(use -4 and -6 for IPv4 and IPv6 specifically)

**ping -a <Your\_remote\_IPAddress>**

**nslookup**(type your local and remote machine name and IP address multiple times)

Be careful to see any mismatch on the returned results. If you are not able to ping your target machine, it has high chance that either the network is broken or the target machine is not running. It's possible the target machine is behind a firewall and the firewall blocks the packets sent by ping, though. Windows firewall does not block ping (ECHO) packet by default. The correctness of DNS configuration on the network is vital to SQL connection. Wrong DNS entry could cause of all sorts of connectivity issue later. See this link for example, ["Cannot Generate SSPI Context" error message, Poisoned DNS](http://blogs.msdn.com/sql_protocols/archive/2007/01/02/cannot-generate-sspi-context-error-message-poisoned-dns.aspx).

**Step 2: SQL Server configuration issue**

You need to make sure the target SQL Server is running and is listening on appropriate protocols. You can use SQL Server Configuration Manager (SCM) to enable protocols on the server machine. SQL Server supports Shared Memory, Named Pipes, and TCP protocols (and VIA which needs special hardware and is rarely used). For remote connection, NP and/or TCP protocols must be enabled. Once you enabled protocols in SCM, please make sure restart the SQL Server.

You can open errorlog file to see if the server is successfully listening on any of the protocol. The location of errorlog file is usually under:

%ProgramFile%Microsoft SQL Server/MSSQLxx.xxx/MSSQL/Log

If the target SQL instance is a named instance, you also need to make sure SQL Browser is running on the target machine. If you are not able to access the remote SQL Server, please ask your admin to make sure all these happen.

**Step 3: Firewall issue**

A firewall on the SQL Server machine (or anywhere between client and server) could block SQL connection request. An easy way to isolate if this is a firewall issue is to turn off firewall for a short time if you can. Long term solution is to put exception for SQL Server and SQL Browser.

For NP protocol, please make sure file sharing is in firewall exception list. Both file sharing and NP use SMB protocol underneath.

For TCP protocol, you need put the TCP port on which the SQL Server listens on into exception.

For SQL Browser, please put UDP port 1434 into exception.

Meanwhile, you can put sqlservr.exe and sqlbrowser.exe into exception as well, but this is not recommended. IPSec between machines that we are not trusted could also block some packets. Note that firewall should never be an issue for local connections.

**Step 4: Client driver issue**

At this stage, you can test your connection using some tools. The tests need to be done on client machine for sure.

First try:

**telnet <your\_target\_machine> <TCP\_Port>**

You should be able to telnet to the SQL server TCP port if TCP is enabled. Otherwise, go back to check steps 1-3. Then, use OSQL, SQLCMD, and SQL Management Studio to test sql connections. If you don't have those tools, please download SQL Express from Microsoft and you can get those tools for free.

OSQL (the one shipped with SQL Server 2000) uses MDAC.

OSQL (the one shipped with SQL Server 2005 & 2008) uses SNAC ODBC.

SQLCMD (shipped with SQL Server 2005 & 2008) uses SNAC OLEDB.

SQL Management Studio (shipped with SQL Server 2005 & 2008) uses SQLClient.

Possilbe command use be:

**osql -E -SYour\_target\_machineYour\_instance** for Windows Auth

**osql -Uyour\_user -SYour\_target\_machineYour\_instance** for SQL Auth

SQLCMD also applies here. In addition, you can use **“-Stcp:Your\_target\_machine, Tcp\_port”** for TCP,  **“-Snp:Your\_target\_machineYour\_instance”** for NP, and **“-Slpc:Your\_target\_machineYour\_instance”** for Shared Memory. You would know if it fails for all protocols or just some specific procotols.

At this stage, you should not see general error message such as error 26 and error 40 anymore. If you are using NP and you still see error 40 (Named Pipes Provider: Could not open a connection to SQL Server), please try the following steps:

a)       Open a file share on your server machine.

b)       Run “net view [\your\_target\_machine](file:///\\your_target_machine\)” and “net use [\your\_target\_machineyour\_share](file:///\\your_target_machine\your_share)”  (You can try Map Network Drive from Windows Explorer as well)

If you get failure in b), it's very likely you have OS/Network configuration issue, which is not SQL Server specific. Please search on internet to resolve this issue first.

You can try connection using both Windows Authentication and SQL Authentication. If the tests with all tools failed, there is a good chance that steps 1-3 were not set correctly, unless the failure is logon-related then you can look at step 6.

If you succeeds with some of the tools, but fails with other tools, it's probably a driver issue. You can post a question on our forum and give us the details.

You can also use “windowssystem32odbcad32.exe” (which ships with Windows) to test connection by adding new DSN for various drivers, but that's for ODBC only.

**Step 5: Application issue**

If you succeed with steps 1-4 but still see failure in your application, it's likely a configuration issue in your application. Think about couple of possible issues here.

a) Is your application running under the same account with the account you did tests in step 4? If not, you might want to try testing in step 4 under that account or change to a workable service account for your application if possible.

b) Which SQL driver does your app use?

c) What's your connection string? Is the connection string compatible to your driver? Please check <http://www.connectionstrings.com/> for reference.

**Step 6: Authentication and logon issue**

This is probably the most difficult part for sql connectivity issues. It's often related to the configuration on your network, your OS and your SQL Server database. There is no simple solution for this, and we have to solve it case by case. There are already several blogs in sql\_protocols talking about some special cases and you can check them see if any of them applies to your case. Apart from that, things to keep in mind:

a) If you use SQL auth, mixed authentication must be enabled. Check this page for reference <http://msdn.microsoft.com/en-us/library/ms188670.aspx>

b) Make sure your login account has access permission on the database you used during login ("Initial Catalog" in OLEDB).

c) Check the eventlog on your system see if there is more information

At last, please post question on our [forum](http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=87&SiteID=1). More people could help you over there. When you post question, you can refer to this link and indicate you see failure at which step. The most important things for us to troubleshoot are a) exact error message and b) connection string.