.NET Family of Products Installation Guide

Table of Contents

1. Introduction

The purpose of this document is to enable customers and their IT teams install Adserversolutions.NET products. This guide covers:

1. Technologies and versions supported
2. Standard deployment on shared and dedicated hosting
3. Enterprise deployment on virtual dedicated and dedicated hosting

The guide also covers how to install Geo database. Customers who do not wish to purchase Geo database may opt for our Geo web service where their systems can connect to our central Geo database for Geo targeting.

2. Technologies & Versions Supported

Operating Systems	Any windows OS, 32 or 64 bit that runs .NET framework 2.0 and upwards
Scripting	Asp.net, c# code behind, server side includes
Databases	SQL Server 2000 and 2005, Oracle 10g upwards. Works with express version of both SQL Server and Oracle.
Hosting	Shared, virtual dedicated, dedicated
System Architecture	Multiple server support.
SMTP Server	Standard .net SMTP components used. Will work with any SMTP server.
Third party components	Graphical components, pre-packaged
Minimum space requirements	1GB disk space, 500 mb MSSQL space, 1GB if using Geo database.
Other requirements	SMTP server, outbound webservices calls (if using geo webservice),

3. Packages Supplied

Adserversolutions_dep_x.y.z.zip	This package contains files and components for both standard and enterprise versions. This is a deployment package for live installations. X.y.x represents version number.
Adserversolutions_dev_x.y.z.zip	This package contains source associated with the deployment package. This package is not for installation and is only for customers who have purchased source and who wish to extend the product. Source code includes code for core manager programs and APIs for business components. X.y.z represents version number.

4. Standard Deployment

Standard version of Adserversolutions product can be deployed on shared and dedicated servers. Given below are step by step instructions on how to deploy the application. Steps mentioned apply to both shared and dedicated servers, unless otherwise stated.

Step 1	Install Adserver Package	Unzip the supplied package into a directory. For example Adserver. Ensure that project directory has write permission. Also ensure that after you right click on directory and click on properties and then security tab, Users on the machine must have full control to the Adserver project directory.
Step 2	IIS set up	• Create a project in IIS. This could be a website or a virtual directory. • Point this IIS directory to a directory physically located on the file system. E.g. d:\Adserver. • Ensure that IIS domain or sub-domain points to the physical directory on the file system. E.g. http://www.myadserver.com will map to d:\Adserver. • Ensure that index.aspx is the default page when domain is accessed. This can be set by going into project properties and then documents tab. • Ensure that under Home Directory tab IIS project properties, following is set: read/write access to directory and execute permissions set to Scripts.
Step 3	Verify IIS set up	Ensure that the IIS directory is able to serve simple HTML and aspx pages. E.g. http://www.myadserver.com/test.html and http://www.myadserver.com/test.aspx. To create an html and aspx file yourself, refer to Appendix A. If aspx page gives “access denied/forbidden” error, go to project properties in IIS and then to Home Directory tab and set Execute Permissions to “Scripts Only”.
Step 4	Database Setup	Create a database on your SQL server installation. E.g. Adserver. Execute following scripts on the database using Query Analyzer or equivalent database tools. • Run ss_ads_create_tables.sql (ignore warnings given out by SQL server) • Run ss_ads_create_sp.sql • Run ss_ads_create_udf.sql • Run ss_ads_create_data.sql • Run ss_ads_create_data_help.sql * if this script times out, it can be run in parts As per your organizations security policy, create a SQL userid. Ensure that this sql id has all permissions to the database, especially the db_owner role. Customers with oracle database can follow the same steps described above. All files prefixed with ora_ will need to be run on an oracle database. Order in which files should be run is: • ora_ads_create_packages.txt • ora_ads_create_tables_and_sp.sql • ora_ads_create_data_sp1.txt • ora_ads_create_data_sp2.txt

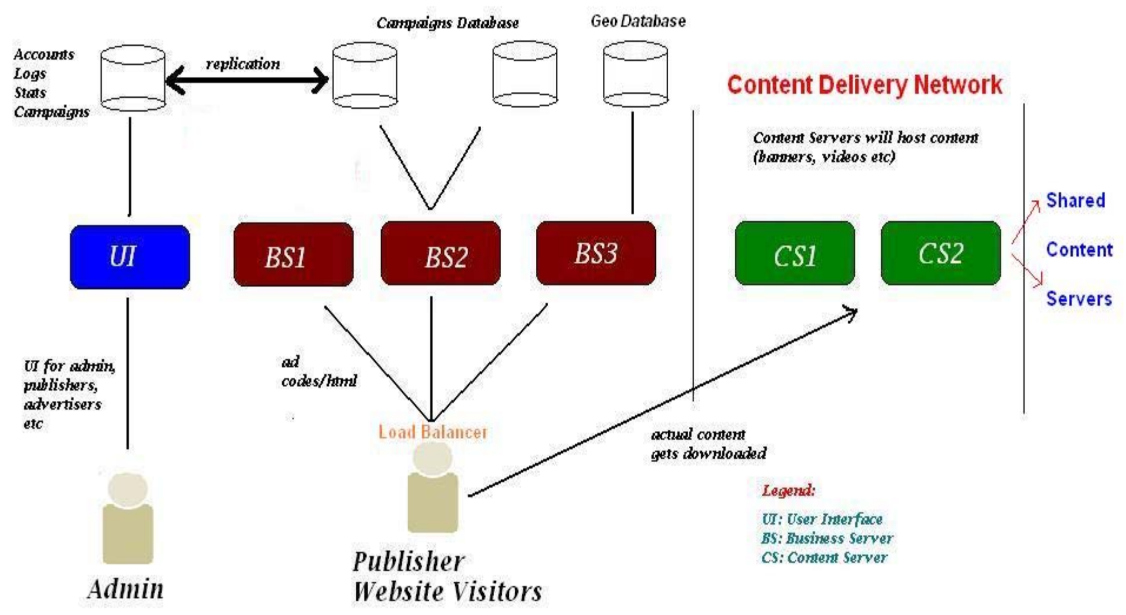
		• ora_ads_create_data_sp3.txt • ora_ads_create_data_sp4.txt • ora_ads_create_data_sp5.txt To install Geo database, please see instructions in Section named “Geo Database Installation”.
Step 4	Adserver Configuration	A web based and a desktop application interface has been provided for configuring the product. To configure using web based interface, open link http://www.myadserver.com/configure.aspx. This will open a page that will be used to configure the application. To configure using desktop based interface (currently works only on 32 bit OS), remote desktop connection to the server is required. ApplicationBuilder.exe can be started from ApplicationBuilder directory. After initial set up of database connection string, enter userid = Adserver and password = temp and select Production Support option. NOTE: This step may take a few minutes to load since it will look for sql server installations with sample connection strings supplied with the package. Given below are mandatory fields that must be configured: • Server IP • Application root. E.g. http://www.yourdomain.com when project set up as website in IIS or http://www.mydomain.com/Adserversolutions when project set up as virtual directory in IIS. • Application path. Note that path must end in a \\ • Log Path: Specify a path on the server where log files will be created. • SMTP details: Specify your SMTP server details like smtp server name, userid and password. • Connection Strings: Specify connection string by changing database ip address, sql id and password. Click on Test Connection to ensure that database connection is working. If you have purchased Geo database, set up the connection string for this database too. If you have not purchased Geo database, we recommend that you copy the same connection string for both databases. Example SQL Server connection string: Provider=sqloledb;Data Source=yourserver;Initial Catalog=yourdatabase;User Id=youruserid;Password=yourpassword;Min Pool Size=20;Max Pool Size=500;Connection Timeout=30 Example Oracle connection string: Provider=MSDAORA.1;User ID=;Password=;Data Source=XE;Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;Incr Pool Size=5; Decr Pool Size=2; Please see appendix C on SQL server connection pooling details if you wish to fine tune your connection tuning parameters.

		• Save. If you see a message indicating that connection to database has been successful, set up it complete. If you see a yellow screen, it indicates that write permission on the Adserver directory must be granted. You may need to contact your hosting provider for permissions.
Step 5	Activate Product	Open link http://www.myadserver.com/activate.aspx. This will open a page that will be used to activate the product. Enter the license key provided. Upon activation, you should see license key details on the same page. This indicates that product has been successfully activated. Otherwise, please contact our product support.
Step 6	Sign On	Open link http://www.myadserver.com/index.asp. This will open the sign on page. Following ids will be used based on product purchased: • Userid: netadmin, password: temp This id will be used if you have purchased an adnetwork product. • Userid: pubadmin, password: temp for default publisher id. • Userid: mercadmin, password: temp for default merchant • Userid: affadmin, password temp for default affiliate admin. Note that modules you will be able to access will depend on the license key so these ids will be applicable only to specific modules.

5. Enterprise Deployment

5.1. Architecture Planning

Enterprise deployment requires planning the network architecture based on capacity requirements. Typical enterprise architecture for a high volume operation is presented below.



Definitions of proposed logical servers

UI Server (UI):
Reserve a server for the main portal that will serve as a point of access for all users. This server will use its own database for persisting information like ads, campaigns, orders and statistics. You may use multiple UI servers that may be load balanced.

Business Servers (BS1, BS2 and so on):
Business servers will be used for ad selection. These servers will be the busiest servers that will select ads from database based on supplied criteria. Business servers will have to be load balanced for high throughput and failover. Each business server will work with its own database.

Content Servers (CS1, CS2 and so on):
You may reserve separate servers for serving content. While business servers select ads to stream, it will be content servers that will actual stream content down to client browser and devices. Linux servers with high bandwidth availability are most ideally suited for content servers.

Companies that do not wish to segregate roles of business and content servers may stream content from business servers themselves.

If you wish to use our proprietary hosted CDN, all you will need is business servers and one entry into CDNServers table to point to our CDN server. This model can enable customers with virtual dedicated or even shared servers to run high bandwidth application. Contact us for details.

Database Servers (DB):
For this high throughput and scalable architecture to work, all databases must work in merge replication mode.

5.2. Deployment Scenarios

1. **One physical server acting as UI, BS and CS on single database** – This is typically a standard product and clients will not derive real benefits of an enterprise set up. Only some benefits of enterprise product like memory based processing will be available.

2. **One server for UI and BS on single database and separate CS** – This enterprise set is useful for clients who wish to segregate their content streaming from core business logic. Clients with low throughput but high bandwidth requirements (video ads) can benefit from this scenario.

3. **Separate servers for UI and BS on single database and separate CS** – Good for high throughput and high bandwidth applications. Easy set up but no failover on database side and reliance on regular database backups.

4. **Separate servers for UI and BS on multiple databases and separate CS** – This is the most robust and scalable scenario. Merge replication required to keep all databases in sync with each other.

5.3. Deployment Details per Logical Server

User Interface Server

Product Set Up
Deploy the product as explained in “standard deployment” section.

CDN Servers Set Up

Additionally, a table called CDNServers is present in the database. To prevent accidental updates, this table cannot be edited from an online screen.

```
[ServerID] [int] IDENTITY(1,1) NOT NULL,
[Domain] [varchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[FTPId] [varchar](30) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[FTPPwd] [varchar](30) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[ServerStatus] [varchar](20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
```

Domain is the domain path where content resides. FTP id/pwd are required so that content broadcast program can broadcast content to all content servers.

Operational Notes:

- If a server is not active, set ServerStatus = N.
- If new entries are added to this table, website must be restarted in IIS.
- If the server that runs adserver product also needs to be used as a CDN server, then an entry must be added as follows:
    - Use the AppRoot value in web.config and append “Ads/” and store this in domain column.

Process Set Up

- ContentBroadcast.exe <mins> – this process will to be scheduled (windows or third party scheduler) every N minutes, as defined in the argument. This process will take all ads uploaded by users in the last N minutes and will broadcast (FTP) these ads to the designated content servers, as defined in CDNServers table.

  Note: This program is not required to be run if [a] all servers have capability to map to a common drive [b] external CDN system like our proprietary CDN is being used.

Business Servers

Each business server will have a copy of adserver product installed and configured the same way as described in Section 4 above.

For higher performance, we recommend that <AppRoles>, <CommandTypes> and <ViewID> tags be removed from web.config of business servers that will run only the rotation program. Server that will run the user interface programs must have these tags.

Process Set Up

Following processes will have to be set up for the enterprise version. These processes reside in the Processes directory. It is recommended that these exes are run on the business servers and instead run on another server (database server may be used). This will ensure that web servers are not overloaded with this processing.

- LogsArchive.exe <mins>, <path> – this program will archive all ad logs from the database into the file system. This program may be scheduled (windows or third party) and it takes two arguments.
  - <mins> - logs older than specified minutes to be deleted. So, if this parameter is 300, then all logs older than now minus 300 minutes will be archived and purged.
  - <path> - directory location where logs will be archived in txt files.

Database Servers

Depending on deployment scenario, ad server database will be set up on each designated database server and web.config will accordingly be configured with the connection string.

In the deployment scenario where multiple replicated databases are used, all databases configured with UI and BS servers will need to be set up for merge replication. All tables except logs table will be replicated.

You may choose to install one Geo database per database server. Geo database is required only for business servers. Geo database set up instructions are provided in the following section.

6. Geo Database Deployment

Customers who have purchased Geo database will need to follow the steps below. All files are supplied along with license key information.

- Create new database. E.g. AdserverGeo.
- To upload Geo information, you have two options. [a] import data from txt file using tools like DTS or [b] restore the full database backup.
- To restore the full database backup, unzip GeoIP.zip. This will extract a file called GeoIP.bak. Restore this.
- If you wish to import data, unzip GeoDBScripts.zip. Run the scripts on the AdserverGeo database to create tables.
- Unzip GeoIP.zip. This will extract two txt files that can be imported into the two tables using utilities like DTS. GeoLiteCityBlocks.txt will be imported into GeoLiteCityBlocks table. GeoLiteCityLocation.txt will be imported into GeoLiteCityLocation table.
- NOTE: All Geo files are delimited by character comma.
- Alternatively, customers may use our Geo web service in which case they must check with their hosting provider if they permit outbound web service connections. See Appendix B for details on configuring Geo as a web service.

7. Common Issues Faced

For all issues faced during installation, we recommend all customers to submit logs file from the logs directory. Application will capture majority of errors in the log and it will enable us to trouble shoot issues.

However, we are outlining common problems faced during installation and their possible resolutions.

Cannot configure product using configure.aspx	Ensure that directory has write permission. Also users must be full control to the directory.
Cannot connect to database	Check the connection string. Contact your hosting provider for correct server name and credentials.
Logs indicate object not coming from trusted source	Check with hosting provider if they can reduce object level security to low in machine.config. Or contact us for resolution.
Application producing excessive logging.	Ensure that trace is set to only error or warning by opening configure.aspx.
License key upgrade does not activate the features purchased.	Restart IIS For shared hosting customers, call http://www.myadserver.com/configure.aspx and save it again. This will restart IIS automatically.
Transaction log full.	Shared hosting customers may face situations where transaction logs get full when trying to upload large GeoIP databases. Please contact your hosting company to compress log files.

8. Appendix A

Sample HTML File

Open notepad or any editor of choice and paste contents given below. Save it as test.html under the root of your domain (d:\Adserver) and try to access using http://www.myadserver.com/test.html.

```
<HTML>
 <HEAD>
  <TITLE>Test HTML Page</TITLE>
 </HEAD>

 <BODY>
                Hello World !!
 </BODY>
</HTML>
```

Sample ASPX File

Open notepad or any editor of choice and paste contents given below. Save it as test.aspx under the root of your domain (d:\Adserver) and try to access using http://www.myadserver.com/test.aspx.

```
<%@ Page language="c#" AutoEventWireup="true" %>

<html>
<head>
<title>ASP.NET Hello World</title>
</head>
<body bgcolor="#FFFFFF">

<p>Hello World!</p>
<%Response.Write(Server.MapPath("/"));%>

</body>
</html>
```

9. Appendix B

This appendix provides a snap shot of a file in /bin folder of your package called WebF.AdStreaming.dll.config. It comes pre-configured with our standard Geo web service. However, if this web service server needs to be pointed to another server, line in bold will be required to be changed:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup,
System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
            <section name="WebF.AdStreaming.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
        </sectionGroup>
    </configSections>
    <applicationSettings>
        <WebF.AdStreaming.Properties.Settings>
            <setting name="WebF_AdStreaming_GeoTarget_manager" serializeAs="String">
                <value>http://67.96.208.130:8080/geoip/ui/manager.asmx</value>
            </setting>
        </WebF.AdStreaming.Properties.Settings>
    </applicationSettings>
</configuration>
```

10. Appendix C

ADO.NET Connection Pooling at a Glance

Establishing a Connection with a database server is a hefty and high resource consuming process. If any application needs to fire any query against any database server, we need to first establish a connection with server and then execute the query against that database server.

Not sure whether you felt like this or not, when you are writing any stored proc Or a query, the query returns the results with better response time than the response time, when you execute that same query from your any client application. I believe, one of the reasons for such behavior is the overheads involved in getting the desired results from the database server to the client application; and one of such overheads is establishing the Connection between the ADO.

Web applications frequently establish the database connection and close them as soon as they are done. Also notice how most of us write the database driven client applications. Usually, we have a configuration file specific to our application and keep the static information like Connection String in it. That intern means that most of the time we want to connect to same database server, same database, and with same user name and password, for every small and big data.

ADO.NET with IIS uses a technique called Connection Pooling, which is very helpful in applications with such designs. What it does is, on first request to database, it serves the database call. Once it is done and when client application requests for closing the connection, ADO.NET does not destroy the complete connection, rather it creates a connection pool and puts the released connection object in the pool and holds the reference to it. And next time when the request to execute any query/stored proc comes up, it bypasses the hefty process of establishing the connection and just picks up the connection from the connection pool and uses that for this database call. This way, it can return the results faster comparatively.

Let us see Connection Pooling Creation Mechanism in more detail.

Connection Pool Creation

Connection Pool and Connection String goes hands in hands. Every connection pool is associated with distinct connection string and that too, it is specific to the application. What in turn means is – a separate connection pool is maintained for every distinct process, app domain and connection string.

When any database request is made through ADO.NET, ADO.NET searches for the pool associated with exact match for the connection string, in the same app domain and process. If such pool is not found, ADO.NET creates a new one for it, however, if it is found, it tries to fetch the usable connection from that pool. If no usable free connection is found in the pool, new connection is created and added to the pool. This way, new connections keeps on adding to the pool till *Max Pool Size* is reached, after that when ADO.NET gets request for further connection, it waits for *Connection Timeout* time and then errors out.

Now the next question arises is - How any connection is released to pool to be available for such occasions? Once any connection has served and is closed/disposed, the connection goes to the connection pool and becomes usable. At times, connections are not closed/disposed explicitly, these connections do not go to the pool immediately. We can explicitly close the connection by using Close() or Dispose() method of connection object Or by using the "using" statement in C# to instantiate the connection object. It is highly recommended that we close or dispose(don't wait for GC or connection pooler to do it for you) the connection once it has served the purpose.

Connection Pool Deletion / Clearing Connection Pool

Connection Pool is removed as soon as the associated app domain is unloaded. Once the app domain is unloaded, all the connections from the connection pool becomes invalid and are thus removed. Say for example, if you have an ASP.NET application, the connection pool gets created as soon as you hit the database very first time, and connection pool is destroyed as soon as we do iisreset. We'll see it later with example. Note that connection pooling has to do with IIS Web Server and not with the Dev Environment, so do not expect the connection pool to be cleared automatically by closing your visual studio .Net dev environment.

ADO.NET 2.0 introduces two new methods to clear the pool: ClearAllPools and ClearPool. **ClearAllPools** clears the connection pools for a given provider, and **ClearPool** clears the connection pool that is associated with a specific connection. If there are connections in use at the time of the call, they are marked appropriately. When they are closed, they are discarded instead of being returned to the pool.

Refer to the section *"Simple ways to View Connections in the pool created by ADO.NET"* for details of how to determine the status of the pool.

Controlling Connection Pool through Connection String

Connection string plays a vital role in connection pooling. The handshake between ADO.NET and database server happens on the basis of this connection string only. Below is the table with important Connection Pooling specific keywords of the connection strings with their description.

Name	Default	Description
Connection Lifetime	0	When a connection is returned to the pool, its creation time is compared with the current time, and the connection is destroyed if that time span (in seconds) exceeds the value specified by **Connection Lifetime**. A value of zero (0) causes pooled connections to have the maximum connection timeout.
Connection Timeout	15	Maximum Time (in Secs) to wait for a free connection from the tool
Enlist	'true'	When **true**, the pooler automatically enlists the connection in the creation thread's current transaction context. Recognized values are **true**, **false**, **yes**, and **no**. Set Enlist = "false" to ensure that connection is not context specific.
Max Pool Size	100	The maximum number of connections allowed in the pool.
Min Pool Size	0	The minimum number of connections allowed in the pool.
Pooling	'true'	When **true**, the **SQLConnection** object is drawn from the appropriate pool, or if it is required, is created and added to the appropriate pool. Recognized values are **true**, **false**, **yes**, and **no**.
Incr Pool Size	5	Controls the number of connections that are established when all the connections are used.
Decr Pool Size	1	Controls the number of connections that are closed when an excessive amount of established connections are unused.

* Some Table contents are extracted from Microsoft MSDN Library for reference

Other than the above mentioned keywords, one important thing to note here. *If you are using Integrated Security, then the connection pool is created for each user accessing the client system, whereas, when you use user id and password in the connection string, single connection pool is maintained across for the application.* In the later case, each user can use the connections of the pool created and then released to the pool by other users. Thus using user id and password is recommended for better end user performance experience.

Sample Connection String with Pooling related Keywords

The connection string with the Pooling related keywords would look somewhat like this

initial catalog=Northwind; Data Source=localhost; Connection Timeout=30; User Id=MYUSER; Password=PASSWORD; Min Pool Size=20; Max Pool Size=200; Incr Pool Size=10; Decr Pool Size=5;

Simple ways to View Connections in the pool created by ADO.NET

We can keep a watch on the connections in the pool by determining the active connections in the database after closing the client application. This is a database specific stuff, so to see the active connections in the database server we must have to use database specific queries. *This is with the exception that connection pool is perfectly valid and none of the connection in the pool is corrupted.*

For MS SQL Server: Open the Query Analyser and execute the query : EXEC SP_WHO

For Oracle : Open the SQL Plus or any other editor like PL/SQL Developer or TOAD and execute the following query -- SELECT * FROM V$SESSION WHERE PROGRAM IS NOT NULL

All right, let us do it with SQL Server 2000

1. Create a Sample ASP.NET Web Application

2. Open an instance of Query Analyzer and run the EXEC SP_WHO query. Note the loginname column, and look for MACHINENAME\ASPNET. If you have not run any other ASP.NET application, you will get no rows with loginname as "MACHINENAME\ASPNET".

3. On Page load of default startup page, add a method that makes a database call. Say your connection string is "initial catalog=Northwind; Min Pool Size=20;Max Pool Size=500; data source=localhost; Connection Timeout=30; Integrated security=sspi"

4. Run your ASP.NET application

5. Now repeat Step 2 and observe that there are exactly 20 (Min Pool Size) connections in the results. Note that you made the database call only once.

6. Close the web page of your web application and repeat step 2. Observe that even after you closed the instance of the web page connections persists.

7. Now Reset the IIS. You can do that by execute the command "iisreset" on the Run Command.

8. Now Repeat Step 2 and observe that all the 20 connections are gone. This is because your app domain has got unloaded with IIS reset.

Common Issues/Exceptions/Errors with Connection Pooling

1. You receive the exception with the message: "***Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool. This may have occurred because all pooled connections were in use and max pool size was reached***" in your .NET client application.

This occurs when you try using more than Max Pool Size connections. By default, the max pool size is 100. If we try to obtain connection more than max pool size, then ADO.NET waits for *Connection Timeout* for the connection from the pool. If even after that connection is not available, we get the above exception.

Solution(s):

1. Very first step that we should do is – Ensure that every connection that is opened, is Closed explicitly. At times what happens is, we open the connection, performs the desired

database operation, but we do not close the connection explicitly. Internally it cannot be used as available valid connection from pool. The application would have to wait for GC to claim it, until then it is not marked as available from pool. In such case, even though you are not using max pool size number of connection simultaneously, you may get this error. *This is the most probable cause of this issue.*

2. Increase Max Pool Size value to a sufficient Max value. You can do so by including "Max Pool Size = N;" in the connection string, where N is the new Max Pool size.

3. Set the Pooling Off. Well, this indeed is not a good idea as Connection Pooling puts a positive performance effect but it definitely is better that getting any such exceptions.

2. You receive the exception with the message: "***A transport-level error has occurred when sending the request to the server. (provider: Shared Memory Provider, error: 0 - Shared Memory Provider:)***" in your ASP.NET application with MS SQL Server

This occurs when MS SQL Server 2000 encounter some issues and has to refresh all the connections and ADO.NET still expects the connection from the pool. Basically, it occurs when connection pool gets corrupted. What in turn happens is, ADO.NET thinks that the valid connection exists with database server, but actually, due to database server getting restarted it has lost all the connections.

Solution(s) :

1. If you are working with .NET and Oracle using ODP.NET v 9.2.0.4 or above, you can probably try adding "*Validate Connection=true"* in the connection string. Well, in couple of places, I noticed people saying use *"validcon=true"* works for them for prior versions on ODP.NET. See which works for you. With ODP.NET v 9.2.0.4, *"validcon=true"* errors out and "*Validate Connection=true"* works just fine.

2. If you are working with .NET 2.0 and MS SQL Server, You can clear a specific connection pool by using the static (shared in Visual Basic .NET) method SqlConnection.ClearPool or clear all of the connection pools in an appdomain by using the SqlConnection.ClearPools method. Both SqlClient and OracleClient implement this functionality.

3. If you are working with .NET 1.1 and MS SQL Server,

a. In the connection string at the run time append a blank space and try establishing the connection again. What in turn it would do is, a new connection pool would be created and will be used by your application, In the meantime the prior pool will get removed if it's not getting used.

b. Do exception handling, and as soon as you get this error try connection afresh repeatedly in the loop. With time, ADO.NET and database server will automatically get in sync.

Well, I am not totally convinced with either approach, but frankly speaking, I could not get any better workable solution for this so far.

3. **Leaking Connections**

When we do not close/dispose the connection, GC collects them in its own time, such connections are considered as leaked from pooling point of view. There is a strange possibility that we reach max pool size value and at that given moment of time without actually using all of them, having couple of them leaked and waiting for GC to work upon them. This would actually lead to the exception mentioned above, even if we are not using max pool size number of connections.

Solution(s):

1. Ensure that we Close/Dispose the connections once its usage is over.

End of Document