

Using Distributed COM with Firewalls

By Michael Nelson

Introduction

This article is intended to provide the reader with information on how to configure the Distributed Component Object Model (DCOM) to work through a firewall. It assumes that the reader is familiar with the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), DCOM, and understands the basics of firewall technology. Please note that the port restriction procedures outlined below only work on Microsoft® Windows NT®; Microsoft Windows® 95 does not include this functionality at this time (don't worry, it isn't necessary in most cases). Also note that much of the discussion in this article is also relevant for remote procedure call (RPC) applications which use dynamic ports. This makes sense considering that the features mentioned below are really implemented in the RPC subsystem that DCOM leverages to communicate across the network.

Unlike most Internet applications which have fixed TCP and/or UDP ports, DCOM dynamically assigns—at run time—one TCP port and one UDP port to each executable process serving DCOM objects on a computer. That is, even if a process is hosting 2,000 clients and 50,000 objects, any client wishing to communicate with objects owned by it will always connect to the same TCP or UDP port. Clients discover the port associated with a particular object by connecting to and using the services provided by DCOM's Service Control Manager (SCM). The SCM always operates at a fixed network port on every computer; in the Internet case this is always port 135 for both TCP and UDP. The SCM offers several RPC-based (not DCOM/ORPC-based) services which handle operations like: "create a new COM class object for me and tell me what TCP and UDP port it is on" or "I have an interface pointer, tell me where I need to go to actually use it", and so forth. A much more technical explanation of this process and of the DCOM wire protocol in general is documented in the Distributed Component Object Model Protocol—DCOM/1.0.

DCOM's dynamic port allocation feature offers great flexibility in that programmers and administrators alike are free from the burden of having to configure (or hard code) applications to specific ports, free from resolving conflicts between multiple applications attempting to use the same port(s), and so on. Unfortunately, because DCOM (by default) is free to use any port between 1024 and 65535 when it dynamically selects a port for an application, it is rather "firewall unfriendly" out of the box. Configuring your firewall to leave such a wide range of ports open would present a serious security hole. Microsoft's developers realized this and have implemented a feature that allows you to restrict the range of ports that DCOM will use to assign to applications.

You should be aware that callbacks in DCOM are *not* handled on the same connection that is used for client/server method calls. When a server makes a callback to a client, it creates a new connection to the client and sends method calls over that separate channel. In other words, DCOM treats callbacks just like any other client/server method call, except that your "client" is really a server and the "server"

Using Distributed COM with Firewalls

By Michael Nelson

is really a client. In some circumstances (this is *very* rare), you may need to configure port restrictions on your clients if your firewall restricts what ports machines on the inside can connect to on the outside.

Also note that if you want to use callbacks through a firewall, you must use TCP. The reason for this is that when the server makes a call to the client, the source port will not be within the range below and thus when the client sends a reply to the server's source port, it will not be able to penetrate the firewall. This is not a problem with TCP because most firewalls keep track of TCP connections and permit bidirectional traffic on connections, regardless of the source port, as long as they are opened from a machine on the inside.

One last thing before I continue, the client *must* be able to reach the server by its actual IP address. You cannot use DCOM through firewalls that do address translation (i.e. where a client connects to virtual address 198.252.145.1 that the firewall maps transparently to the server's actual address of, say, 192.100.81.101). This is because DCOM stores raw IP addresses in the interface marshaling packets and if the client cannot connect to the address specified in the packet, it won't work.

Configuring DCOM to Use TCP Only

Here is a rundown of the transport protocols that DCOM will use depending on the client/server platform:

Platform	Protocol
Microsoft Windows NT 4.0 <-> Windows NT 4.0	UDP
Microsoft Windows 2000 <-> Any	TCP (*)
Microsoft Windows 95 or Windows 98 <-> Any	TCP
DCOM for UNIX <-> Any	TCP

* When a Windows NT 4.0 client attempts to connect to a DCOM server running anything but Windows NT 4.0, there will be a 30-45 second delay while it attempts to connect via the UDP protocol. Windows 95 and Windows 98, Windows 2000, and DCOM for UNIX (including Microsoft's DCOM for UNIX product and Software AG's EntireX product) always use the TCP protocol.

The Windows NT 4.0 implementation of DCOM uses UDP wherever it can because benchmarks show that it can outperform TCP in certain scenarios. Unfortunately, it is very difficult to make UDP applications work through firewalls without exposing your network to unnecessary risks. Before continuing, be sure you make TCP the default protocol on all Windows NT servers by moving the

Using Distributed COM with Firewalls

By Michael Nelson

"NCACN_IP_TCP" value to the top of the list in the *DCOM Protocols* named value of the **HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc** registry key using regedt32.exe. To eliminate a 30-45 second delay when connecting to TCP only servers, the same change should also be made on Windows NT 4.0 clients. Note that this delay only occurs if you haven't connected to the target server in a while (RPC's connection caching features retain knowledge about the server so that multiple connections in a short period of time are resolved instantaneously).

The Windows 2000 implementation of DCOM will likely default to TCP because it is not possible to implement the SSL and SNEGO security protocols over UDP.

Restricting the Range of TCP Ports

There are several registry settings that control the DCOM port restriction functionality. All of the named values listed below are located under the

HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Internet registry key (which you must create).

Remember that you only need to do this on the server machine. Clients will automatically pick up the right port numbers when they connect to the SCM on the server machine.

Note You must use regedt32.exe to configure these settings, regedit.exe does not currently support the REG_MULTI_SZ type required by the *Ports* named value entry. Also, you must reboot your machine any time you make changes to any of the following registry settings in order for them to take effect.

Name	Type	Value	Description
Ports	REG_MULTI_SZ	Specify one port range per line. Example: 3000-4000 5141	One or more port ranges. The options below determine the meaning of this named value.
PortsInternetAvailable	REG_SZ	"Y" (don't include quotes)	Always set this to "Y".
UseInternetPorts	REG_SZ	"Y" or "N" (don't include quotes)	If this value is set to "Y", then the <i>Ports</i> named value indicates which ports should be used for DCOM applications. If this value is set to "N", then the <i>Ports</i> named value indicates which ports should NOT be used for DCOM applications.

Using Distributed COM with Firewalls

By Michael Nelson

Restricting the Internet Accessibility to Specific Applications

An even greater level of security is afforded if you configure your system so that DCOM applications have to explicitly ask to be accessible through the Internet-accessible port range. To set this up:

Change the *UseInternetPorts* named value above from "Y" to "N" so that DCOM object servers (and RPC servers) aren't reachable via the Internet-accessible ports automatically. Reboot your machine.

Insert the following piece of code before **CoInitializeEx()** into every DCOM object server application that should be accessible through the Internet-accessible port range:

```
RPC_POLICY rp;
rp.Length = sizeof (RPC_POLICY);
rp.EndpointFlags = RPC_C_USE_INTERNET_PORT;
rp.NICFlags = RPC_C_BIND_TO_ALL_NICS;
hr = RpcServerUseAllProtseqsEx (RPC_C_PROTSEQ_MAX_REQS_DEFAULT, NULL, &rp);
Add "rpcrt4.lib" to the link command line in your makefile.
```

Configuring Your Firewall

The firewall between your server and the Internet should be configured as follows:

Deny all incoming traffic from the Internet to your server.

Permit incoming traffic from all clients to TCP port 135 (and UDP port 135, if necessary) on your server.

Permit incoming traffic from all clients to the TCP ports (and UDP ports, if necessary) on your server in the *Ports* range(s) specified above.

If you are using callbacks, permit incoming traffic on all ports where the TCP connection was initiated by your server.

About the Author

Michael Nelson is an undergraduate student at the University of Maryland and is the owner of Iapetus Software, a software consulting and development firm based in Rockville, Maryland. You can contact him at mikenel@iapetus.com. Comments or questions specifically about this article are welcomed and may be sent to dcomfw@iapetus.com.