

Writing a windows service with a socket interface.

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2005-02/1114.html>

From: Peter Strøiman (*blah_at_blahblahblah*)

Date: 02/17/05

Date: Thu, 17 Feb 2005 11:04:41 +0100

Hi.

I'm writing a windows service that is accessible via sockets. I have tried to find literature on the subject but I couldn't.

Basically my service, when asked to start, starts up a new thread that just waits for connection. See bottom of this mail.

The StreamParser class is the facade that parses data sent to my service, and sends a response back to the client.

I'm left with a couple of questions.

A: Are there any good resources on writing windows services with socket interfaces anywhere?

B: Is my main code robust. E.g. if some idiot accidentally pulled the network cable out of the server, would `_listener.AcceptTcpClient()` throw an exception. Or perhaps if the IP address of the server was modified while the service was running.

C: When a client connects and sends a request and thereafter gets a response, should I close the socket, or keep the socket open. I'm writing the class that handles the client's communication with the server as well, so I can control these things. If the best option is to leave the connection open, I would implement a sort of connection pool of course.

D: How should I detect if clients are disconnected? My StreamParser implementation simply calls the blocking `Stream.Read()` method, waiting for data to arrive. Can I count on that the stream throws an `IOException` if the client is disconnected? E.g. if the network cable is unplugged (either from the server or client), someone changes the IP address of the server/client, or the client process suddenly terminates. Of course I would implement a "close" command in my communication protocol that would manually close a connection.

Thanks in advance,
Peter Strøiman

microsoft.public.dotnet.framework: Writing a windows service with a socket interface.

```
// service main thread
try
{
    _listener = new TcpListener( _ipEndPoint );
    _listener.Start();
    _state = TcpInterfaceState.Running;
    while( true )
    {
        TcpClient client = _listener.AcceptTcpClient();
        if ( client != null )
        {
            Stream stream = client.GetStream();
            StreamParser parser = new StreamParser( stream );
            Thread thread = new Thread( new ThreadStart( parser.Run ) );
            thread.ApartmentState = ApartmentState.STA;
            thread.Start();
        }
    }
}
catch( Exception e )
{
    // error handling code
}
```