Sending a 4 byte message header from C# client to a Java Server

Asked 16 years, 3 months ago Modified 16 years, 3 months ago

Viewed 7k times

7

I am trying to write a C# client to a server that is written in Java. The server expects a 4 byte (DataInputStread readInt() in Java) message header followed by the actual message.

I am absolutely new to C#, how can I send this message header over to the Java Server? I tried it several ways (mostly trial and error without getting too deep into the C# language), and nothing worked. The Java side ended up with the incorrect (very large) message length.

c# java sockets

Share

Improve this question

Follow

edited Sep 19, 2008 at 14:27

aib

46.8k ● 10 ● 74 ● 79

asked Sep 18, 2008 at 12:57

tellme

6 Answers Sorted by: Highest score (default)

11

It is, as other posters have pointed out, down to endianness.

The Java DataInputStream expects the data to be big-endian (network byte order). Judging from the Mono documentation (for equivalents like BinaryWriter), C# tends toward being little-endian (the default for Win32/x86).

So, when you use the standard class library to change the 32bit int '1' to bytes, they produce different results:

```
//byte hex values
Java: 00 00 00 01
  C#: 01 00 00 00
```

You can alter the way you write ints in C#:

```
private static void WriteInt(Stream stream, int n) {
    for(int i=3; i>=0; i--)
    {
        int shift = i * 8; //bits to shift
        byte b = (byte) (n >> shift);
        stream.WriteByte(b);
    }
}
```

EDIT:

A safer way of doing this would be:

```
private static void WriteToNetwork(System.IO.BinaryWri
    n = System.Net.IPAddress.HostToNetworkOrder(n);
    stream.Write(n);
}
```

Share Improve this answer

Follow

edited Sep 18, 2008 at 15:24

answered Sep 18, 2008 at 14:22

McDowell

109k ● 31 ● 206 ● 270

2

It's simple, but have you checked endianness? It could easily be a mismatch between the endianness you have sent the data in and the endianness you are recieving in.

Share Improve this answer

Follow

answered Sep 18, 2008 at 13:01

workmad3

25.6k ● 4 ● 37 ● 56

2

As everyone here has already pointed out, the issue is most likely caused by the C# application sending ints in little-endian order whereas the Java app expects them in network order (big-endian). However, instead of explicitly rearranging bytes in the C# app, the correct way is to rely on built-in functions for converting from host to network order (htons and the likes) -- this way your code will

continue working just fine even when run on a big-endian machine.

In general, when troubleshooting such issues, I find it useful to record the correct traffic (e.g., Java to Java in your case) using tools like netcat or wireshark, and then compare it to the incorrect traffic to see where it's going wrong. As an added benefit, you can also use netcat to inject the captured/prerecorded requests into the server or inject captured/prerecorded responses into the client. Not to mention that you can also modify the requests/responses in a file and test the results before commencing with fixing the code.

Share Improve this answer Follow

edited Sep 18, 2008 at 15:08

answered Sep 18, 2008 at 14:32

Alexander

9,380 ● 2 ● 28 ● 23

1

If you are going to be exchanging a lot of data, I would recommend implementing (or finding) a Stream-wrapper that can write and read ints in network-order. But if you really only need to write the length do something like this:

```
using(Socket socket = ...){
  NetworkStream ns = new NetworkStream(socket);
  ns.WriteByte((size>>24) & 0xFF);
  ns.WriteByte((size>>16) & 0xFF);
  ns.WriteByte((size>>8)  & 0xFF);
  ns.WriteByte( size      & 0xFF);
```

```
  // write the actual message
}
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 13:12

Rasmus Faber

49.6k ● 25 ● 148 ● 193

0

I dont know C# but you just need to do the equivalent of this:

```
out.write((len >>> 24) & 0xFF);
out.write((len >>> 16) & 0xFF);
out.write((len >>>  8) & 0xFF);
out.write((len >>>  0) & 0xFF);
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 13:09

Craig Day

2,535 ● 1 ● 24 ● 26

0

The Sysetm.Net.IPAddress class has two static helper methods: HostToNetworkOrder() and NetworkToHostOrder() that do the conversion for you. You can use it with a BinaryWriter over the stream to write the proper value:

```
using (Socket socket = new Socket())
using (NetworkStream stream = new NetworkStream(socket
using (BinaryWriter writer = new BinaryWriter(stream))
{
    int myValue = 42;
```

```
    writer.Write(IPAddress.HostToNetworkOrder(myValue)
}
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 14:48

ageektrapped

14.5k ● 7 ● 60 ● 74