

## Re: Socket write behaviour is inconsistent?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-11/msg01288.html>

---

- *From:* c jard <mcw8@xxxxxxxxxxx>
  - *Date:* Fri, 09 Nov 2007 17:07:10 -0800
- 

The concept is that the following approaches ought to be equivalent:

- Send 1 byte, Send many bytes, Send 1 byte
- copy 1 byte to buffer, copy many bytes to buffer, copy one byte to buffer, send entire buffer
- make a list of 3 ArraySegment<byte>

Of course they should be equivalent. But obviously they are not. So obviously there is something else going on.

Your opinion of their suggested equivalence is appreciated, thank you.

By this time, there are now two strong pieces of evidence that you have not posted enough code to answer your question definitively:

And my assertion that I have, still stands. I could post the entire program, but the only variation you will see is in the lines given. As they are the only variance, I am seeking opinions or explanations of any apparent or known difference.

- 1) See above. The mere fact that your short description describes something that `_should_` work but doesn't means there's something missing from the short description.

I disagree. I'm not here to debate the semantics of this. I have

## Re: Socket write behaviour is inconsistent?

posted 3 sections of code, two that work and one that does not, with the assertion that their behaviour in a goal oriented sense, ought to be equivalent. The behaviour is not equivalent, because the results observed differ.

2) Has anyone provided you with a definitive answer based on the information you've provided so far?

There are several possible scenarios beyond your answer to your own rhetoric. In summary, most people reading the question either do not know the answer, or cannot be bothered to explain it. I usually demand a level of precision from others when they ask a question that I know the answer to, but feel that they have not sufficiently helped themselves through their own clarity of thought in producing an understandable description of the problem. If you feel this to be the case here (to wit; you know the answer but you're taking the same stance that I take in being bloody minded about the specification of the question, for my own good), do let me know, wont you? :)

people are just ignoring your question for the heck of it

While I dont mean to sound big headed, it's rare that I ask a question, not because I know everything, but usually because I can easily take a correct guess or research something I dont know. Occasionally, when I find the topic particularly hard to research or exceptionally unusual, I'll defer to other sources such as people more knowledgeable than myself. As these situations are invariably few and far between, and those questions often go unanswered, I'm left with one of two conclusions; I'm crap at asking questions, or noone knows/ cares to give the answer.

You can assert that you've provided enough information 'til the cows come home, but as long as a person trying to answer the question tells you that you haven't provided enough information, you haven't.

Hmm.. Well, we can run with that, and I'll answer your probes for more info as far as I am able. There are limitations to what I can discover here, which I dont doubt may lead us to an impasse; I wont be able to provide information you assert to be essential to answering the question. Ergo, the question shall have no answer.

## Re: Socket write behaviour is inconsistent?

There is no exception. The client device is a relatively dumb terminal – a credit card terminal to be precise. When I interact with it in the first way, it drops the connection and contacts the server again, each time spitting out a piece of paper saying "End of Transmission from host"

Then why did you write "the client throws an error"?

I'll concede that choosing the word "throws" was poor; in .NET terms it would have led you to believe the client app was something under source code control

find: throws  
replace: reports

I have assumed in my reply that when you write "client" you are talking about your own code and when you write "host" you are talking about the remote endpoint for your connection.

I wrote the host, the client is a dumb terminal running software developed by Sagem

If those are not the correct labels, you should clarify that, and you should be VERY specific about what errors and other behaviors happen. Your original post is very vague on those matters, and as with the lack of actual code, it makes it very hard to answer the question without better details.

Considering the bi-directionality of TCP communication, I'm not sure that the specifics on the client and server really matter, but I did point out the desired dialog flow at the start of the post.. Here is what goes on:

The terminal collects transaction information from user interaction and makes a dialup connection over GPRS, to O2. O2 route the connection to BT's Cardway X25 PAD. From there the connection is routed via some proprietary protocol we know (nor care) anything about terminating in a device attached to one of the ISDN lines entering the building. The device has a connection to the LAN and forges a connection to the host software I wrote, on a specific port. I read the request, calculate some things and response with a few bytes.

When the response is sent using the first chunk of code, the terminal

Re: Socket write behaviour is inconsistent?

Re: Socket write behaviour is inconsistent?

gives up, announces that we sent it an EOT (we didnt) and redials.  
When corresponding in the latter two ways, the terminal is satisfied with the response it receives, drops the connection and we dont hear from it again

The terminating 0x03 is mandatory, as far as the spec is concerned.

No doubt that as far as the spec is concerned everything should work fine as it is now.

I dont really understand this sentence. It appears to be missing some punctuation.

But that's not really all that relevant, is it?

The spec is relevant; the terminal is asserted to be compliant with the spec, and I really dont think a couple of people chewing over a puzzle on Usenet can question every major bank in the UK at this stage.

Again, I'm not really clear on who is "server" and who is "client" here. When I write "server" in the above quote, I'm talking about the remote endpoint of your connection, because you appeared to write "client" to describe your own code. But if that's not right, you should clarify.

Hm. OK, few definitions:

Client: the submitter of a Request, a credit card terminal with a software written by a third party

Server/Host: the respondant to the Request, the issuer of the Response, a piece of .NET code written by myself

Your original post implies that the remote endpoint is closing the connection. Is that what is happening? If not, please explain in detail what is actually happening.

The spec states that the recipient of the last message in a dialogue

Re: Socket write behaviour is inconsistent?

## Re: Socket write behaviour is inconsistent?

should close the connection. In all cases as far as I am concerned, I send a response. I only find out if it is satisfactory or not when the client re-presents itself (or not)

If the client goes away and is not heard from again ,for this request, then it was satisfied

When sending the response piecemeal, the client returns. When buffering the whole response in a byte[] or List and then sending it, the client does not return. The diagnostic codes printed by the client when it retries, indicate that it received an EOT from the server host. EOT is defined as 0x04, not that it matters; No such byte is ever sent by the host and EOT is taken to mean that the transmission was ended.

If you still feel that your question is an appropriate .NET question, you need to provide more details as I've described.

You touched on the source of my puzzlement in that I have to work around an issue. Your opinion that the three methods ought be equivalent was most useful; i concur and I have no explanation for why I should have to "work round" in this way, or even why such a "work round" works at all. If all three methods are functionally equivalent, why does one fail?

.