Subject: Socket::IsOpen BUG/FIX (?)
Posted by kohait00 on Wed, 20 Jan 2010 06:57:36 GMT

hi there

in socket.h:73
class Socket

bool IsOpen() const { return !!data; }

is maybe wrong, since the Data class provides an IsOpen(), to which it should map, right?

so maybe like this:
bool IsOpen() const { return (data)?(data->IsOpen()):(false); }
or
bool IsOpen() const { return data && data->IsOpen(); }

or do i miss something??

Subject: Re: Socket::IsOpen BUG/FIX (?)
Posted by rylek on Wed, 20 Jan 2010 20:23:44 GMT

Hello!

Conceptually you are right. I think the main reason for this awkwardness is historical evolution of the socket interface. The socket wrapper (class Socket) in practice receives its Data object only when it's open (through the ClientSocket / ServerSocket functions) and Socket::Close clears the internal pointer after closing the socket, so that for practical purposes (until now) it sufficed to check validity of the pointer. However with support for SSL and other extensions it started to be easier to allow the Data object to remain closed for a certain (initial) period. This is mainly to allow internal creation and initialization of the socket implementor object within the ClientSocket / ServerSocket functions, i.e. within "internals" of the socket hackery. I'm not sure what would happen if someone tried to put an unopened Socket::Data object into a Socket class. I'll discuss this with Mirek in greater detail but in any case I'm sure the modification you suggest is completely harmless and might prove itself valuable when dealing with various exceptional socket circumstances.

Regards

Tomas

Subject: Re: Socket::IsOpen BUG/FIX (?)
Posted by kohait00 on Wed, 20 Jan 2010 22:03:35 GMT

View Forum Message <> Reply to Message

thanks, i'll use this meanwhile, just to intercept any occurence of side effects

---

Subject: Re: Socket::IsOpen BUG/FIX (?)
Posted by mirek on Thu, 21 Jan 2010 22:01:50 GMT

View Forum Message <> Reply to Message

I agree this should be harmless and it is logical too -> patch applied.