Subject: Re: HTTPS?
Posted by mirek on Fri, 01 Dec 2006 19:43:44 GMT
View Forum Message <> Reply to Message

rylek wrote on Fri, 01 December 2006 14:29Hello,

the 'problem' with Socket implementing pick semantics is honestly more about getting Mirek to like it than a problem with the socket class as such. From the code point of view I believe the current implementation with a separate interface wrapper object and its internal socket handler object is most logical as the sockets use a completely different access interface than SSL sockets. Mixing the implementation of the two together would work well if the Open / OpenSSL methods did only some initialization of setup stuff, which they don't. The only result would be that practically every access method would have to begin with an if() distinguishing the two interfaces and wanting the code buildable without SSL would require tens of #if-#endif pairs to mask off all these SSL variants.

No, that is not why I am suggesting. In fact, the only change that I suggest is to transform current functions

```
bool ServerSocket(Socket& socket, int port, bool nodelay = true, int listen_count = 5, bool
is_blocking = true);
bool ClientSocket(Socket& socket, const char *host, int port, bool nodelay = true, dword *my_addr
= NULL, int timeout = DEFAULT_CONNECT_TIMEOUT, bool is_blocking = true);
bool SSLServerSocket(Socket& socket, SSLContext& ssl_context, int port, bool nodelay = true, int
listen_count = 5, bool is_blocking = true);
bool SSLClientSocket(Socket& socket, SSLContext& ssl_context, const char *host, int port, bool
nodelay = true, dword *my_addr = NULL, int timeout = DEFAULT_CONNECT_TIMEOUT, bool
is_blocking = true);
```

into methods, which would eliminate the need for exposing internal Data in Socket contructor interface. (and also would activate Assist++ hints

In fact, for backwards compatibility, I would even retain both them and that constructor...

Implementation could be as trivial as:

```
bool Socket::OpenClient(const char *host, int port, bool nodelay = true, dword *my_addr = NULL,
int timeout = DEFAULT_CONNECT_TIMEOUT, bool is_blocking = true) {
   ClientSocket(*this, host, port, nodelay, my_addr, timeout, is_blocking);
}
```

Mirek