Subject: Re: About storing references and pointers to callbacks.
Posted by Oblivion on Wed, 05 Jul 2017 21:01:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 05 July 2017 15:02Oblivion wrote on Mon, 26 June 2017 13:16Quote:
Well, I have some reservation about the whole api design, but yes, this case I would say it is ok and has to be documented.

Thank you Mirek.

I hope I'm not taking much of your time, but could you share your thoughts on that?
So that I can revise the code before publishing it.

Well, _right now_ I feel a bit uneasy about the whole asynchronous operations queue. Now, it is likely that the whole concept is forced by the nature of problem, but I would rather liked to have some "single request" tool.

Actually, I think that your original question suggest exatly that long queues that store references to external objects are troublesome...

Also, I think that there always should be Event<void *, int> basic variant for data outputs.

Hello Mirek,

Thanks for the answer.

I understand your uneasiness about the queue. But it allows writing "single request" tools too.
I mean it can be easily programmed to process only one asynchronous or synchronous job at a time.
For example, it all takes adding a single line to the StartGet() method above to make it a "single request" (adding JobQueue::ClearQueue() will clear the queue if there are pending jobs prior to current call). However, as you might have already noticed in the SSH package I published (which is still a work-in-progress), I did not walk that path. I simply provided non-blocking calls and their blocking counterparts. And that's intentional. My experience with the asynchronous sockets in the past years have taught me that performing a single operation with them is trivial but "a set of sequential operations" is at best tedious. Whole point of the queue is to simplify that process for the "user" (developer) and make the code readable by reducing complexity. It proved well up to the task. (E.g. I have several unpublished packages such as a dbus client which uses the JobQueue, and they all work fine with several different scenarios. The queue model simplified things a lot in the asynchronous sockets' realm.).

As for the data outputs using an Event<void*, int>, indeed, that's a valid point too. I should think on it.

Best regards,

Oblivion