

## Re: Remobjects, good in multi-threading environment?

---

*Source:*

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.thirdpartytools.general/2007-12/msg00299.html>

---

- *From:* Arthur Hoornweg <[antispam.hoornweg@xxxxxxxxxxxxxx](mailto:antispam.hoornweg@xxxxxxxxxxxxxx)>
  - *Date:* Fri, 07 Dec 2007 09:32:26 +0100
- 

Danijel Tkalcec wrote:

Using the RealThinClient SDK, you can simply set MultiThreaded to TRUE for your client-side connection component and have the RTC SDK handle all processing in background threads, while you can still make your calls from the main thread, from a timer or from other threads.

Hmm. Is it handled in a "real" background thread, or does it simply create a messagepump? Can I do things like WaitForSingleObject(event, timeout) in my own threads, without it interfering with the communication?

There will be absolutely no need to shut threads down and all connection issues are handled gracefully by the RealThinClient SDK.

You see, each of my threads is designed to be suicidal. Any error results in an exception being raised. And any exception causes a very thorough shutdown of the thread.

My threads are state machines. When the internet connection (DSL modem) breaks down, state is broken in the middle of the communication and it will take some time before the line is functional again.

Any exception in a thread causes the thread to be shutdown completely. Each method cleans up itself thoroughly and re-raises exceptions until the exception propagates into the Execute() method. Then the execute() method cleans up and quits. The destructor then removes the thread from the heap.

Then, after a few minutes timeout, the thread is re-created and tries to re-establish communication. If it fails, the process repeats itself until it succeeds.

So you see, being able to recover from a breakdown is the core part of the design. The only thing I must be certain of is that the objects that

Re: Remobjects, good in multi-threading environment?

I free are really disposed of properly and that they don't leave any handles, hsockets or message queues dangling in the depths of the operating system. I must be especially careful that they don't "fire" any asynchronous events after being disposed of because these would point to methods that no longer exist.

Do you need more proof of the RTC SDK stability?

What challenges my application is a particularly unstable environment (comms between oil rigs and remote offices). With internet connections breaking down and network cables being pulled inadvertently. If I pull the network cable and re-insert it after 5 minutes, will RTC survive that and continue business as usual?

—

Arthur Hoornweg

(In order to reply per e-mail, please just remove the ".net" from my e-mail address. Leave the rest of the address intact including the "antispam" part. I had to take this measure to counteract unsolicited mail.)

.