

Re: Communicating with DCOM server using another user account

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.ole/2007-04/msg00082.html>

- *From:* "Brian Muth" <bmuth@xxxxxxxx>
 - *Date:* Mon, 23 Apr 2007 20:26:43 -0700
-

"nicolasr" <nicolasrREMOVETHISSPAMBLOCKER@xxxxxxxx> wrote in message news:eZ7ycGfhHHA.4924@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

You may be right, the entire design should probably be questioned. Currently the situation is like this:

The server is registered for autostart and runs as "Interactive user", all clients shall use this same instance.

There are two types of clients, say EndUser and Controller. EndUsers are allowed anonymous read access to all the servers "data". But write access is denied.

Controllers should have full write access to all "data" and must therefore authenticate themselves. For this a special account is created on each machine a Controller will run on. The Controller should impersonate using this account and the server calls CoQueryClientBlanket to identify its a Controller. (Of course the server machine will have an identical account for this to work).

So up to here everything would probably work with CoSetProxyBlanket as you suggested.

Unfortunately one of the Controllers must run on the same machine as the server. It will also get autostarted.

If I were free to implement the servers interfaces myself I could maybe write methods that optionally accept a password that the Controllers could use. But the interfaces are fixed by the OPC specification which says that security should be implemented via Windows security APIs.

I really appreciate any comment.

Probably the easiest way is to create a launcher app that uses CreateProcessAsUserW as I earlier posted...

Re: Communicating with DCOM server using another user account

Ah, this should have been the very first post.

This is trivially solved if you use a COM+ application. Rather than creating a COM executable, create a COM DLL instead and using Component Services, add it to the COM+ application. The COM+ application should run as a specific account.

Component Services allow you to define roles over which clients have access to the COM DLL. Obviously you would be defining the EndUser and the Controller role. Only users belong to one of those two roles would be permitted access.

From within the COM+ DLL, the implementation would call CoGetCallContext to

acquire the ISecurityCallContext interface, and then call ISecurityCallContext::IsCallerInRole() to determine which role the client is in and to behave accordingly.

Slip-slap... you're done.

Brian

.