Two related questions about keeping track of PIDs of IPC clients

devblogs.microsoft.com/oldnewthing/20250801-00/?p=111427

August 1, 2025

A customer was writing a program that was intended to run in the background and was looking for recommendations on how to store its PID so that other programs can find it and start communicating with the background process.

Another customer was writing a provider program that will have multiple clients connect to it. They want the program to know when all of its clients have exited, so that the provider program can exit, too. They were thinking of polling the list of running processes to see if any with names matching those of its clients is running. Is there a better way? Can they get the PIDs of the clients so they can wait for them?

In both cases, the customer is working too hard.

Since they already have an IPC mechanism for the clients to talk to the providers, they can just have the client pass its own PID to the provider as part of the initial connection. Now the provider has the PIDs and can convert them (via `OpenProcess`) into process handles and wait on the handles.

The customers didn't say what IPC mechanism they were using, but there may be features already built into the IPC mechanism they are using that provide the client PID,[1] or even better, that let them know when the client has disconnected, since disconnection can happen even if the client process hasn't yet exited. For example, if they are using a COM server, they can use the final `Release()` from the client to signal that the client has disconnected. If they are using an RPC server, they can use context rundown to detect that the client has disconnected.

My point is that since you are already using IPC, you can use that IPC channel to pass the information you need.

[1] For example, you may be able to use the `RpcServerInqCallAttributes` function to obtain the PID of an RPC client.