Problem
We need a way for launcher to communicate with client and server, and vice versa.

Use cases
I highlighted the messages that need to be sent between processes:
client → launcher
server → launcher
launcher → client
launcher → server

1. Starting a local game:
    1. Launcher spawns the server
    2. When server is ready to receive connections, it must notify launcher about it
    3. After receiving the message, launcher spawns a client
2. Refreshing client(s) when settings are changed from launcher's SETTINGS tab. When settings are saved, launcher must notify the client that cvars have changed. As a reaction, client should refresh to reflect the changes - I see 3 scenarios:
    1. Changing the cvar requires stopping the game, and restarting it entirely. Think of cvars related to graphics, or -fs_mod cvar. Some cvars must be even passed as launch arguments, so there is no other way than restarting client.
    2. After changing the cvar, only a /retry is necessary for the client to refresh. These are cvars like cl_player_name, and other configurable via SETTINGS→PLAYER.
    3. Cvar change is reflected right away. These are cvars like snd_volume.

3. Refreshing local server when settings are changed in launcher's LOCAL tab. When settings are saved, launcher must notify the server. Based on the type of cvars that changed, I see 2 scenarios like above:
    1. Changing the cvar requires stopping both client and server, and restarting them.
    2. Cvar change is reflected in server right away.

4. Client could notify the launcher about success/failure when connecting to server. I'm referring to cases when grey background with white text appears – message about wrong password, about wrong checksum of a .smod file, and so on… Not sure what launcher could do in this case
5. Client could notify launcher about file downloads? Idk what's the current behavior of client when a file download is in progress
6. Client could notify launcher when a server redirects/forwards the player to another server? Again, idk what's the current behavior in such scenarios, but I assume client could handle this on its own?

7. Client could have a new entry in ESC menu that says "Settings". When clicked, client could notify launcher – as a reaction, client's window would be minimized, and launcher's window would be brought up, displaying the SETTINGS tab. I think this would improve user's experience.

Concerns

- I can't tell if Case 1 is needed, but current implementation has always felt flaky to me, and a person that tested the launcher told me that he couldn't launch a local game. I think asking people to test current implementation on their machines is the only way to find out
- Despite the communication between launcher and game, most changes to the settings won't be smooth anyway, and they will require a rejoin, or a restart of the game. Cases 2 and 3 make most sense for cvars that don't require a restart, like mouse sensitivity, or sound volume. For other cvars, we need to either inform the player that he needs to restart his client, or launcher could implement client's restarting/rejoining
- Cases 4, 5 and 6 could be covered by the game itself, there's no need for launcher here imo. Besides, switching focus between game's window and launcher's window when handling those scenarios could be pretty bad when it comes to UX…

Requirements

- Implement cases 2, 7, 3, maybe 1

- There are different message formats that can be used. Keep in mind that the communication is bidirectional, so messages sent from launcher to client/server may have a different format than messages sent from client/server to launcher:
    - We can assume that every message sent from launcher to client/server will be passed directly to OpenSoldat's command system, so we can either run a command, or set a cvar. What I'd like to have though is the ability to send multiple cvar settings/commands at once, in one message. We can think of some separator so that it's easy to parse these messages in OpenSoldat, ideally it would probably map to a TStringList.
    - Each message sent from client/server to launcher will at least have an identifier (most likely a string that identifies the command we want to execute on the launcher, such as "SHOW_SETTINGS"). To pass parameters, we could use a separator

What I'm afraid of is that both these approaches will limit the ability to extend the launcher integration in the future. Messages sent from launcher to client/server must be implemented as a command or cvar, and messages sent from game to launcher don't have a clean way of passing parameters.
Conceptually, I feel like JSON-RPC would make more sense, especially for game → launcher messages. At the same time, though, it seems overkill for the use cases that we're currently planning. As an alternative, maybe we could encode the messages as JSON, without following the whole JSON-RPC standard. It would make the IPC communication cleaner, and we could easily pass parameters in the messages.

- Preferably, the server for IPC should be started from launcher. When OpenSoldat's client or server starts, it should connect to this IPC server. If the IPC server was on OpenSoldat's client or server, launcher would have to rely on hacky approaches to figure out when the IPC server is ready to accept connections. Another benefit of having IPC server on launcher is that we only need to start one server, and all clients/servers can connect to it.

- In the game, launcher integration should be toggled with cvars

- Ideally the changes in game related to launcher integration should be self-contained, so it's easy to roll them back if a new GUI gets implemented