A technical look at SOP

This document tries to explain why some of the standard operating procedures have been established, and why some other - sometimes obvious - methods are NOT used. Please note that some of these things are more educated guesswork than downright fact; we do not always know 100% what FDev are doing behind the scenes, but some of this we have verified with them, and other things are established through trial and error methods.

Direction of friend requests

The SOP specifies that the client should be the one to initiate friend requests. We specifically call for them to **not** accept friend requests sent to them for the rescue. This SOP is established not because of a technical reason, but for security. When the client is the person sending the friend request, we can prevent griefers adding the client to their friends list and intercepting the rescue that way. It also reduces the time rats have to spend in menus, giving them more time to jump, rather than handle administrativia.

The bottom line: Rats do not send friend requests. Clients do. **Exception:** On XBox, mutual friend requests are required.

Direction of Wing requests and communications

SOP establishes that the client should be the person sending wing requests to the rats. This is done to improve our chances of successful instancing with the client.

When Elite: Dangerous creates a wing, a Peer-to-Peer session is initialized between the client and the rat's computers. Depending on whether both parties have voice communications enabled or not, that communication may be either simply through the HTTP based text chat communications system, or a direct P2P UDP connection for transmitting and receiving voice data. If a direct P2P connection needs to be established, the involved parties set up that connection using a set of parameters communicated through the matchmaking server. The matchmaking server sends information to the originating party, containing the other party's IP address, the port on which communication should be attempted on, a handshake signature, and some information about the connectivity of the computer they are trying to connect to. Using that information, the originating party attempts to establish a connection to the other party.

The importance of the client being the originating party lies here; with very little work, rats can ensure that they are "open" for direct connections - meaning that they have a port mapping on their network's router that allows traffic from the internet to reach their E:D client correctly. (See What is NAT and why is it a problem?)

The bottom line: The client initiates wing requests. Rats must clear pre-winged status and relog if they have previously been winged to position near the client.

Non-CR Tactical Face Plants

Tactical Face Plants (The act of intentionally slamming into an Exclusion Zone to reach a client) are required for Code Red rescues where the client has lost the ability to power their ship while too close to a star or planet. We provide the optional TFP training specifically for these incidents. TFP training is not standard drill material because in 98% of cases, people who drop out inside Exclusion Zones still have fuel enough to perform a Supercruise Hop and get themselves out of the situation. Thus, TFPs are not SOP for non-CR rescues.

The reasons for this can be summarized simply; it's less risky. For both the client, the rat, and for the outcome of the case. Faceplanting, be it tactical or not, is a risk to the player's ship, both client and rats. It is also a gamble, since you might very well end up well out of reach of the client, even with a carefully planned TFP. When the client **can** get out of the Exclusion Zone, we get them out of the Exclusion Zone first. Then we fuel them.

The bottom line: Tactical Face Plants are for Code Red rescues. It should not be attempted simply as an 'expedient' way of getting to the client. Non-CR TFPs are at the discretion of the dispatcher, when the actual situation requires it. (Client unable to understand instructions to SCHop, or otherwise unable to do so, for instance.)

What is NAT and why is it a problem?

This is necessary because of the wide-spread usage of NAT (Network Address Translation) on the internet. Although E:D has a number of methods to connect through NAT, the easiest and most assured way for it to happen successfully is if at least one party is openly connectable.

There are also a number of pitfalls that may happen even with NAT punchthrough.

- **Correctly establishing the connection details.** When E:D starts up, it probes the NAT configuration of the host and their router, by sending both packets of UDP data to an EDServer, and a string of configuration details established by the client (Such as port mapping, if negotiated through uPnP). This *usually* works quite well, but symmetric NAT can cause problems. Symmetric NAT routers establish a separate port mapping for each outgoing connection from the client, meaning that the router might accept incoming packets from IP 1.2.3.4 on port 23333, but expects packets from IP 2.3.4.5 to port 25555. E:D is not able to detect this natively, and thus has to fall back to TURN to punch through such NAT. Manual portmapping evades this problem.
  Not having a correct connection details to send to other clients can make syncing for wings problematic.
- **Double NAT.** A NAT router behind another NAT router. Although it is possible to punch through this configuration with TURN, it severely limits connectivity. This is not something we run into often, though.
- **Misconfigured manual port forwarding**. When E:D is instructed to NOT utilize uPnP to establish a port mapping, and the manual port mapping is not set up, or is incorrectly configured. This will always result in a connection failure, and an attempt to connect using TURN.

Manual drop vs. navlock drop

When dropping on a client, some swear to using manual drops as a 'better' way to ensure instancing. In truth, there is no technical difference between how the attempt to connect between clients is made. Navlock drops just get you dropping faster.

The bottom line: Use navlock drops

Wingless drop

Dropping wingless on the known location of a beacon is a game of chance at best. The matchmaking servers make no effort whatsoever in placing the players on the same playing field without a wing. Dropping on normalspace through targeting a lowwake left by the client is possible, and valid, as that means you are already on the same EDServer.

The bottom line: If you can see the client in Supercruise and can drop on their Low Wake, you can do so. But as this carries inherent risks of confusion, the *preferred* method is being winged with the client. Use this only if the client is having trouble establishing a wing. (I.e, does not understand the instructions to do so, for instance)

Staying in Normalspace

Some rats swear to staying in Normalspace at the last known location of a client's beacon. This is also very much a game of chance. Although E:D will prefer to drop players into an instance already spun up and populated with players, it will make no special effort to do so without a wing. The only benefit would be that you would, in the case where the player DOES end up on the same EDServer as you are, you are ready to fire limpets the moment the client enters the game.

It IS possible to improve the chances of the client spawning into the same Normalspace instance as the rat by ensuring the rat staying in Normalspace is in the same region (EU or US) as the client. Then, the rat needs to find their CURRENT EDServer ID (a four digit code, like 1131), and the client must restart their game several times until their server ID matches before logging in to the game. Even this is not a surefire way to ensure a Normalspace instance though.

The bottom line: Staying in Normalspace is a game of chance. To ensure instancing, the game has to be given a chance to move you to the same EDServer as the client. That means dropping on their beacon while winged.

EDServers and how they work

Although E:D is a serverless game, meaning that the players communicate directly with eachother rather than to a server which then transmits the location and actions of all involved players, the game does have matchmaking servers. There are several of these per region, and players are distributed between them to reduce the load on the server for the traffic that *does* need to go to FDev servers. These are called EDServers, and to successfully instance with another player, *you must be connected to the same EDServer*. The game has a system to facilitate this when players wing up together. Based on latency and server loads, and to a lesser degree region, the game will move players to a common server. Up until 2.2.3, this mechanism was partially broken, but thanks to amongst others help from us Fuel Rats, these bugs have been greatly reduced.

The move between EDServers happens when you change playfields, i.e when you jump from one system to another, or go from Supercruise to Normalspace. This is why SC hops often helps getting us successful instances; the client gets the chance to move from their EDServer to the same EDServer that the rats are on, the same way that rats have a chance to attempt the move when they drop from Supercruise to Normalspace on their beacon.