Innhold

[Login and character creation 2](#_Toc323757490)

[Gameplay (lot) 3](#_Toc323757491)

# Login and character creation

Client connects to the LoginServer. It sends a LoginRequest packet,which results in a InitLoginNotify packet if the accountname and password was OK, or a LoginFailed packet if the password or account-name was incorrect.

If the authentication succeeded, the rest of the packets are encrypted with a random key generated using the client’s password. The client sends a CharacterInfoRequest packet to the LoginServer, which simply contains a timestamp.

The LoginServer checks the client’s account’s characters in its database. If the timestamp received by the client was newer or older than the timestamp for a character (should be the same for each character), information about all characters is relayed to the client in the form of a response-CharacterInfoRequest packet. If the timestamp sent by the client matched the timestamp for a character, or the client’s account had created 0 characters, the response-CharacterInfoRequest packet simply contains a 0 to indicate 0 characters.

When the client creates a character, it sends a CharacterCreate packet to the LoginServer, which contains information about the newly created character, in addition to the client’s account-name. This information is stored in the LoginServer’s database. This results in a CityServerList packet being sent to the client, containing information about all CityServers that are currently online (meaning that they are connected to the LoginServer).

When the client has selected a city to move the character into, it disconnects from the LoginServer and connects to the corresponding CityServer. Once connected, it sends a CharacterCreate packet to the CityServer. The account-name in this packet is unencrypted. This is always the first packet received by the CityServer, and when received, the server fetches the client’s encryption-key from the LoginServer using the account-name in the CharacterCreate packet[[1]](#footnote-1). It then uses this key to decrypt all packets received by the client.

If the information in the CharacterCreate packet didn’t exist for the client’s account, the character is stored in the CityServer’s database.

# Gameplay (lot)

For each currently active lot in a city, the city server runs a virtual machine. Each object in a house is run on a virtual thread in this virtual machine. The server runs the simulation at a set number of ticks per second.

For each tick, the simulation is advanced one step, and the virtual machine runs one line of code for each object on a lot (sims are also objects). Every 5 ticks, the server compiles a SimulationState packet, containing all the changes for all the objects on a lot for the last 5 ticks. This packet is then relayed to all clients that are present on a lot.

When a client joins a lot, it will receive a SimulationState packet that contains all the changes for all the objects on a lot since the last 10 ticks, unless the client is the first one to join a lot. The client then ticks its simulation 10 ticks so that it is up to date with the server.

When a lot owner places an object, the owner’s client sends a CreateSimulationObject packet to the server.

When a lot owner deletes an object, the owner’s client sends a DeleteSimulationObject packet to the server. The server then relays that packet to all the clients on a lot, so that it is removed from their simulation.

1. The CityServer sends a KeyRequest packet to the LoginServer, which results in a KeyRequest-response packet containing the encryption-key. The LoginServer updates the the city of a created character based on the CityServer that sends the KeyRequest packet. [↑](#footnote-ref-1)