Kommunikationsspezifikation Final Chess

# Allgemeines

Die Nachrichten haben unterschiedliche Größen je nach Bedarf.

Die Codierung ist ISO-8859-1, womit die meisten Europäischen Sprachen abgedeckt sind. Vorteil: Nur 1 Byte!

Um Antworten auf Anfragen besser verarbeiten zu können folgen diese einer Syntax: {ursprung}\_[Success/Failure], somit kann der Parser diese gleich in zwei Kategorien einteilen.

# Server zu Client (Befehle)

* GameStarts
* CurrentPlayer
* GameOver
* GameRestarts
* Leave (Wenn ein anderer Spieler leaved)
* Disconnect (Ein anderer Spieler ist nicht mehr erreichbar)
* Reconnect (Ein anderer Spieler ist wieder erreichbar)
* StillAlive (Server fragt Client ob er noch erreichbar ist)

# Client zu Server (Anfragen)

## Verwaltung

### Login

Client versucht sich auf dem Server einzuloggen, dies ist der allgemeine Login, sprich die Anmeldung beim Server. Das Beitreten eines Spiels erfolgt mit „JoinGame“. Hier könne nauch erweiterte Serverinfos zurückgegeben werden.

Parameter:

1. Client Port
2. Username

* Syntax: login;{PORT}:{USERNAME}

Rückgabewerte:

1. login\_Success;{ClientId}:{MAX\_GAMES} //Erfolg, Die ClientId Hat 6 Zeichen und ist somit max. 1 Million groß.
2. login\_Failure;{ERROR} //Misserfolg, das errorfeld ist Optional.

* Im Erfolgsfall gilt der User als registriert und kann mit seiner ClientId weitere Anfragen versenden!

### CurrentGames

Fordert die aktuell auf dem Server laufenden Spiele an. Eine ClientId ist nicht nötig.

Parameter:

*keine*

* Syntax: currentGames;

Rückgabewerte:

1. currentGames\_Success;({Name}!{PlayerCount}!{Running}!{GameId}:)\*
   1. Name: Maximal 50 Zeichen -> Max. 1 \* 50 = 50 Bytes
   2. PlayerCount 1 Zeichen = 1 Byte ()
   3. Running 1 Zeichen = 1 Byte (t/f) [eigentlich Implizit durch PlayerCount = 2 gegeben!]
   4. GameId 6 Zeichen = 6 Byte und max 1 Million Ids möglich.
2. currentGames\_Failure;{ERROR}

* Um die Paketgröße im Rahmen zu halten kann nein Maximalwert für Spiele definiert werden, als Standard sei 16 (Nachricht etwa 1kByte groß) gewählt.

### JoinGame

Es wird einem Spiel beigetreten, welches durch die GameID repräsentiert wird, der Client weißt sich durch seine ClientId aus. Wenn dass Spiel noch nicht voll ist kann der Client beitreten. Obe das Spiel nun startet wird direkt vom Server verkündet.

Parameter:

1. ClientId
2. GameId

Rückgabewerte

1. joinGame\_Success;{currentPlayer}:{yourPlayer}
2. joinGame\_Failure;{ERROR}

### CreateGame

Erlaubt es ein neues Spiel zu erstellen solange weniger als MAX\_GAMES bereits laufen. Ein erstelltes Spiel endet, wenn alle Spieler sich aus dem Spiel oder dem Server ausgeloggt haben.

TODO: SPEC!

### LeaveGame

Erlaubt einem Client ein Spiel zu verlassen, daraufhin wird das laufende Spiel zurückgesetzt und der zurückgebliebene kann entweder auf einen neuen Spieler warten oder das Spiel ebenso verlassen, woraufhin es zerstört wird, um Platz für neue zu machen!

TODO: SPEC!

## Spiel

### SendMove

Übersendet einen Clientseitigen Zug an den Server.

### GetGameField

Erfragt das aktuelle Spielfeld