Grundlagen des Protokolls

Sämtliche Kommunikation zwischen den Servern und den Clients wird über TCP/IP abgewickelt.

Auf welchen Ports die Server Verbindungen akzeptieren, ist den Spezifikationen der Server zu entnehmen.

Format für Nachrichten

Nachrichten sind das elementare Grundelement, auf dem jede weiter spielespezifische Kommunikation aufbaut. Nachrichten werden als String codiertes JSON-Objekt über das Netzwerk übertragen. Dieser String hat folgendes Format:

```
[Länge des JSON-Strings, Basis 10, inklusive abschließendes Newline][JSON-String][Newline als 0x0A]
```

Beispiel:

```
17{"Hallo":"Welt"}\n
```

Der JSON-String `{"Hallo":"Welt"}` hat eine Länge von 16 Bytes, und das Newline-Zeichen wird als 1 Byte übertragen, daher wird die Summe 17 vor den String geschrieben. Das Newline-Zeichen wurde in obiger Repräsentation über sein Escape `\n` dargestellt; auf Netzwerkebene muss das Byte `0x0A` übertragen werden.

JSON-Strings werden grundsätzlich in **UTF-8** codiert. Treten Nicht-ASCII-Zeichen im String auf, so ist jedes Zeichen in so vielen Bytes zu zählen, wie für die Darstellung des Zeichens in UTF-8 benötigt werden. Beispiel:

```
31{"universität":"TU Chemnitz"}\n
```

Obwohl sich im JSON-String (inklusive Newline) nur 30 Zeichen befinden, sind es dennoch 31 Bytes, da das kleine Ä in UTF-8 mit den Zeichen `0xC3` und `0xA4`, also 2 Bytes, codiert ist.

Um die Übertragungsgröße zu reduzieren, ist es empfohlen, unnötigen Whitespace im JSON-String auszulassen, es ist aber nicht unbedingt notwendig. Unabhängig davon dürfen implementierende Programme Nachrichten, dessen Inhalt 64 KiB (= 65536 Byte) überschreitet, ohne weitere Begründung ablehnen.

Für die Beschreibungen im weiteren Teil dieser Seite sowie anderen Seiten dieser Spezifikation wird JSON in lesbarer Struktur und ohne vorangestellte Länge dargestellt, um das Lesen zu vereinfachen.

Warum dieses Format?

- Das Wurzelelement in JSON muss ein Objekt sein.
- Es ist notwendig, zu wissen, wie groß ein JSON-String ist, da wir nicht wissen, wann er endet.

- Eine maximale Übertragungsgröße ist notwendig, um den Speicherverbrauch der implementierenden Programme kontrollieren zu können.
- Ein abschließendes Newline erlaubt das zeilenweise Einlesen des JSON-Strings, da es sich um ansonsten lesbaren (ASCII-)Text handelt. JSON-Strings ohne Whitespace werden so in eine einzelne Zeile eingelesen.

Kommandos und Antworten

Es werden zwei Arten von Nachrichten unterschieden: **Kommandos** werden von Client zu Server gesendet, um Daten abzufragen oder zu verändern. **Antworten** werden von Server zu Client als Reaktion auf ausgewählte Kommandos gesendet.

Beide Arten von Nachrichten übermitteln aber das gleiche Format:

- Jedes JSON-Objekt enthält einen Schlüssel `"command"`, welches das zu übermittelnde Kommando bzw. die zu übermittelnde Antwort beschreibt.
- Kommandos tragen Namen der Form `"[Serverkürzel][eigentlicher Name]"`.
- Antworten tragen Namen der Form `"[Serverkürzel]R[eigentlicher Name]"`.
- Es sind zwei Kürzel definiert: `ms` für den Verwaltungsserver, `gs` für den Spieleserver.
- Namen der Kommandos werden in CamelCase und Englisch, möglichst in der Form "Infinitivverb + Substantiv", geschrieben.
- Müssen zusätzliche Daten zu einem Kommando oder einer Antwort übermittelt werden, so gibt es einen Schlüssel `"content"`, der ein JSON-Objekt enthält, welches mindestens ein Schlüssel-Wertepaar besitzt.
- Ist es nicht notwendig, weitere Daten zu einem Kommando oder zu einer Antwort zu übermitteln, *so darf der Schlüssel* `"content"` ***nicht*** *existieren.*

Beispiel 1: Wir möchten den Verwaltungsserver anweisen, eine Bratpfanne zu erhitzen, bevor wir Eierkuchen zubereiten können (der Server antwortet uns nicht):

```
{
    "command": "msPreheatPan"
}
```

Beispiel 2: Wir möchten den Spieleserver anweisen, 2 Packungen Mehl im Supermarkt einzukaufen (der Server antwortet uns nicht):

```
{
    "command": "gsBuyIngredient",
    "content": {
        "ingredient": "flour",
        "pieces": 2
    }
}
```

Beispiel 3: Wir möchten den Verwaltungsserver fragen, für wie viele Stück Eierkuchen unsere Zutaten ausreichen.

Der Client sendet:

```
{
    "command": "msEstimateCrepeProduction"
}
```

Der Server antwortet:

```
{
    "command": "msREstimateCrepeProduction",
    "content": {
        "pieces": 7
    }
}
```