

# Datenerhaltung

## TextureSync

Version	1.1
Datum	17.04.19
Autor	Lukas Fürderer
Projektmitglieder	Hendrik Schutter, Lukas Fürderer, Robin Willmann, Jannik Seiler

# Inhaltsverzeichnis

1Einleitung.....	4
2Genereller Aufbau.....	4
3Metadaten.....	4
3.1Definitionen.....	4
4Konsistenz.....	6
5Changelog.....	8

# 1 Einleitung

Dieses Dokument beschreibt die Art und Weise, wie Texturen und deren Metadaten auf dem Server im Dateisystem abgelegt werden.

## 2 Genereller Aufbau

Alle Nutzdaten sind im aktuellen Arbeitsverzeichnis des Serverprozesses untergebracht. Auf diese Weise ist der Speicherort einfach konfigurierbar.

Die Texturdateien selbst liegen im Ordner `./textures/` und haben als Namen ihren SHA256-Hashwert ohne Dateiendung. Der Hashwert ist wie üblich hexadezimal angegeben und die enthaltenen Buchstaben sind klein geschrieben.

Alle Metadaten der Texturen sind in einer Datei unter `./collection.json` abgelegt.

## 3 Metadaten

Die Textur-Metadaten sind in der `collections`-Datei so abgelegt, wie sie auch über das Netzwerk übertragen werden.

### 3.1 Definitionen

Im Folgenden sind sind Datentypen für JSON definiert, welche zur Speicherung verwendet werden:

Für *String*, *Number*, *Array* von `<..>` siehe JSON-Standard.

**UUID ::= <String>**

UUID nach Version 4

Beispiele

- `"a78c59fc-4198-421a-8ba4-db232ad7b91e"`
- `"1f010407-130f-432c-8463-6c61fdfb8c14"`
- `"ecb109bb-d9d6-494d-9d5e-b1e44734e20d"`

**Format ::= "png" | "jpeg"**

Dateiformat

## Beispiele

- "png"
- "jpeg"

## **Resolution ::= [<Number>, <Number>]**

Die erste Nummer stellt die Weite in Pixeln dar, die Höhe in Pixeln wird durch die zweite Nummer repräsentiert.

## Beispiele

- [1024, 1024]
- [2048, 512]
- [13, 400]

## **Tag ::= <String>**

Stellt ein Tag dar. Kann Groß- und Kleinbuchstaben beinhalten.

Hinweis: Vergleiche von Tags sind nicht Case-Sensitiv. Die Darstellung in der UI jedoch unter Umständen schon.

## Beispiele

- "Holz"
- "mEtALL"
- "Chesse Cake"

## **Date ::= <String>**

im Format "yyyy-MM-dd", siehe Javadoc unter *java.text.SimpleDateFormat* für mehr Informationen.

## Beispiele

- "2019-03-04"
- "2017-12-21"

## **Hash ::= <String>**

Sha256-Hash von z.B. Texturdaten oder anderen Binärdaten, in Hexadezimal-Darstellung. Kann Groß- oder Kleinbuchstaben enthalten. Dies wird genutzt, um auf diese zu verweisen.

## Beispiele

- "a98f43a976e5b501961635b981022ebaf98321b97055ead4d8d4de55114015e7"
- "02a08f7d697a93937cc5ace273a534c2eb021ae76b7c15ba146d279d57898893"
- "A6A04ADC2E6D580B8E37CE8F4784652BE6D668EC1FB340B971DD8E8A582CE6BC"

- "7bdc65d8550b0A4FBC899550bbda87DAA2E780D618A66a1F7813967ECF6C0831"

```
Texture ::= {
    id: <UUID>,
    name: <String>,
    tags: <Array von <Tag>>,
    format : <Format>,
    resolution: <Resolution>,
    added_on: <Date>,
    texture_hash: <Hash>
}
```

Stellt einen Textur-Eintrag mit Metadaten dar.

```
CollectionFile ::= {
    textures: <Array von <Texture>>
}
```

Die Datei `./collection.json` enthält genau ein Json-Objekt vom Typ `CollectionFile`.

## 4 Konsistenz

Um die Daten bei einem Serverabsturz konsistent zu halten und auch die Konsistenz eines einfachen Datei-Backups möglichst zu gewährleisten, muss die Collection-Datei atomar überschrieben werden. Hierzu erstellt der Server zunächst eine neue Datei unter dem Namen `./collection_new.json` und füllt diese mit allen notwendigen Daten. Als letzter Schritt wird mit einem `rename(2)` Syscall die `./collection.json` atomar durch die neue Datei ersetzt.

Beim Hinzufügen und Löschen von Texturen können die Datei selbst und der zugehörige Metadaten-Eintrag nicht gemeinsam atomar erstellt bzw. gelöscht werden. An dieser Stelle gilt die Grundregel: Eine Datei darf ohne Metadaten-Eintrag existieren, jedoch nicht umgekehrt.

Beim Hinzufügen einer Textur muss also zuerst die Textur-Datei selbst geschrieben werden, danach darf der Server die `collection.json` aktualisieren.

Umgekehrt muss der Server beim Löschen einer Textur zuerst den Eintrag aus der `collection.json` entfernen und diese neu schreiben, danach darf er die Textur-Datei löschen.

Um Datenmüll aufzuräumen muss der Server zusätzlich nach dem Start das Verzeichnis aller Texturen auslesen und alle nicht mehr referenzierten Texturdateien löschen. Umgekehrt löscht er auch Metadaten-Einträge, bei denen die Textur-Datei fehlt. Dieser Fall kann bei Backups im laufenden Betrieb auftreten.

## 5 Changelog

Version	Änderung
1.0	Dokument Datenerhaltung erstellt
1.1	Daten liegen im CWD statt unter /var/texturesync/