1. Übersicht

SCUM ermöglicht es Euch, Quests über JSON-Dateien zu erstellen und anzupassen, die sich in bestimmten Ordnern befinden, je nachdem, ob ihr einen Multiplayer-Server betreiben oder im Sandbox-Modus spielt. Alle Änderungen an den JSON-Dateien erfordern einen Neustart des Servers (oder einen Neustart des Spiels im Sandbox-Modus), damit die Änderungen wirksam werden.

Link zu einem Tool, das über eine einfache grafische Oberfläche den Erstellungsprozess vereinfacht und gleichzeitig sicherstellt, dass die Ausgabe korrekt strukturiert und anhand bekannter Spieldaten validiert ist:
SCUMQuestTool

Diese Dokumentation wurde von Scumworld.de ins Deutsche übersetzt. Fehler bitte dort melden! Die offizielle Doku gibt es hier auf Englisch: Custom Quest Modification

Relevante Befehle

- `#Quests ListAllQuests`
  Kopiert ein JSON-Array, das alle Standardquests in SCUM auf listet, in die Zwischenablage. Ihr könnt dieses Array in eine Datei einfügen (z.B. DisabledQuests.json), um unerwünschte Standardquests zu blockieren.
- `#ExportQuests`
  Erstellt den Ordner „Quests“ (falls er noch nicht existiert) und exportiert questbezogene JSON-Dateien, die eine Momentaufnahme aller aktuellen Quests darstellen.
- `#GetMeshInfo`
  Wird für Interaktionsbedingungen verwendet, um Objektinformationen auf der Karte abzurufen (z.B. für die Platzierung interaktiver Objekte).

2. Datei und Ordnerstruktur

Quests Ordner

Abhängig von Eurer Einrichtung des Spiels befindet sich der Ordner „Quests“ an folgendem Ort:

- Multiplayer Server:
  `<Server>\SCUM\Saved\Config\WindowsServer\Quests`
- Sandbox:
  `%LocalAppData%\SCUM\Saved\Config\WindowsNoEditor\Quests`

Dieser Ordner wird automatisch erstellt, wenn ihr das erste Mal den Befehl `#ExportQuests` ausführt.
Innerhalb des Ordners Quests findet ihr folgende Unterordner:

Blocked
Enthält BlockedQuests.json, mit der bestimmte Quests nach Namen blockiert (deaktiviert) oder alle Standard Quests blockiert werden können.

Override
Wird verwendet, um benutzerdefinierte Quests hinzuzufügen oder zu aktualisieren. Wenn der Server startet (oder das Spiel in der Sandbox), werden die JSON-Dateien in diesem Ordner analysiert und alle neuen oder aktualisierten Quests übernommen.

QuestList
Enthält CustomQuestList.json (Liste aller exportierten benutzerdefinierten Quests) und DefaultQuestList.json (Liste aller Standard Quests).

Die Bearbeitung dieser Dateien hat **keinen** Einfluss auf den Server oder das Spielgeschehen; sie dienen nur als Referenz.

Blockieren von Standard-Quests

Wenn ihr verhindern möchtet, dass bestimmte Standard Quests angezeigt werden, könnt ihr die Datei BlockedQuests.json verwenden, die sich im Ordner Blocked befindet. Fügt dort einfach den Namen der Quest hinzu, die ihr deaktivieren möchtet. Zum Beispiel:

```
{
  "BlockAllDefaultQuests": false,
  "BlockQuestNames": [
      "T1_AR_Fetch_45ACPAmmobox",
      "T1_AR_Fetch_50AEAmmobox",
      "T1_AR_Fetch_9mmAmmobox"
  ]
}
```

Setzt `BlockAllDefaultQuests` auf true, wenn ihr alle Standard Quests deaktivieren möchtet.

3. Erstellen von eigenen Quests

Alle benutzerdefinierten Quests müssen als ein JSON-Objekt pro Datei im Ordner „Override“ abgelegt werden. Jede Datei beschreibt eine einzelne Quest. Wenn der Server (oder das Spiel) neu gestartet wird, wird SCUM diese Quests laden.

Beispiel der Dateistruktur

```
Quests
|
├── Blocked
|    └── BlockedQuests.json
├── Override
|    ├── MyFirstCustomQuest.json
|    └── MySecondCustomQuest.json
└── QuestList
     ├── CustomQuestList.json
     └── DefaultQuestList.json
```

4. Aufbau einer Quest JSON Datei

Jede Quest, die ihr erstellt, wird durch ein Quest-JSON-Objekt mit spezifischen Eigenschaften definiert. Einige Eigenschaften sind obligatorisch (**fett**), während andere optional sind (*kursiv*).

Nachfolgend findet die Struktur, gefolgt von detaillierten Erklärungen:

```
{
  "AssociatedNPC": "Bartender",
  "Tier": 1,
  "Title": "My First Custom Quest",
  "Description": "Help the Bartender with some tasks.",
  "TimeLimitHours": 24,
  "RewardPool": [
    {
      // One or more Reward objects can go here.
    }
  ],
  "Conditions": [
    {
      // One or more Condition objects can go here.
    }
  ]
}
```

Mehr über `RewardPool` (Belohnungen) und `Conditions` (Bedingungen) später.

4.1 Verpflichtende Eigenschaften

- **AssociatedNPC**
  - Typ: `string`
  - Muss einer der folgenden sein: `"Armorer"`, `"Banker"`, `"Barber"`, `"Bartender"`, `"Doctor"`, `"Fisherman"`, `"GeneralGoods"`, oder `"Mechanic"`.
  - Zeigt an, welcher Händler-NPC die Quest anbietet und woher die Belohnungen (einschließlich Laden Rabatt-Belohnungen) stammen.
- **Tier**
  - Typ: `integer` (valid range: `1` to `3`)
  - Gibt den Schwierigkeits- oder Wichtigkeitsgrad der Quest an.
- **Title**
  - Typ: `string`
  - Ein beliebiger Text für den Namen der Quest.
- **Description**
  - Typ: `string`
  - Eine kurze Beschreibung, die beschreibt, was der Spieler tun muss.
- **RewardPool**
  - Typ: `array` von Belohnungsobjekten.
  - Genau eine dieser Belohnungen wird jedes Mal zufällig ausgewählt, wenn eine neue Instanz der Aufgabe erzeugt wird.
- **Conditions**
  - Typ: `array` von Bedingungsobjekten.
  - Diese definieren die Ziele, die erfüllt werden müssen, um die Quest abzuschließen.

4.2 Optionale Eigenschaften

TimeLimitHours

- Typ: `number` (kann eine ganze Zahl oder eine Dezimalzahl sein)
  - Legt das Zeitlimit für den Abschluss der Quest in Stunden fest (z. B. 24, 48,5 usw.). Weglassen, wenn es kein Zeitlimit gibt.

5. Belohnungsobjekte

Jeder Eintrag im `RewardPool` Array ist ein Reward-Objekt. SCUM platziert maximal fünf „Reward-Slots“ in jedem Objekt, die wie folgt gezählt werden:

- **Gruppe Währung (Normal, Gold, Ruhm):**
  - Alle drei zählen zusammen als **1** Reward-Slot - egal, ob ihr nur einen (z. B. nur CurrencyNormal) oder alle drei (CurrencyNormal, CurrencyGold und Fame) einbezieht.
- **Fertigkeiten:**
  - Jeder **Skill** Grant zählt als 1 Reward-Slot pro Skill.
  - Beispiel: Wenn du Erfahrung in Bogenschießen und Kochen gibst, sind das 2 Reward-Slots.
- **Handelsgeschäfte:**
  - Der **erste** TradeDeal innerhalb eines Reward-Objekts zählt als 2 Reward-Slots.
  - **Jeder weitere** TradeDeal im selben Reward-Objekt zählt als 1 Reward-Slot.
  - Der vergebene Gegenstand muss zum Inventar des Händlers gehören.

Das folgende Beispiel zeigt, wie das Zählen von Belohnungen funktioniert:

```
{
  "CurrencyNormal": 100,
  "CurrencyGold": 1,
  "Fame": 10,
  "Skills": [
    {
      "Skill": "Cooking",
      "Experience": 50
    }
  ],
  "TradeDeals": [
    {
      "Item": "Weapon_M9",
      "Price": 50,
      "Amount": 2,
      "AllowExcluded": false,
      "Fame": 10
    }
  ]
}
```

CurrencyNormal (100) + CurrencyGold (1) + Ruhm (10) → 1 Belohnungsplatz

Kochfertigkeit +50 EXP → 1 Reward-Slot

Erstes Handelsgeschäft → 2 Reward-Slots

Bis jetzt gibt es insgesamt 4 Reward-Slots. Ihr könntet eine weitere Fertigkeit oder einen weiteren TradeDeal (oder ein beliebiges Belohnungselement mit nur einem Slot) hinzufügen, bevor ihr das Limit von 5 erreicht.

5.1 Eigenschaften des Belohnungsobjekts

- CurrencyNormal (integer)

  Der Betrag der normalen Währung, den der Spieler erhält.

- CurrencyGold (integer)

  Der Betrag der Goldwährung, den der Spieler erhält.

- Fame (integer)

  Vergebene Ruhmespunkte.

- Skills (array von Skill-Objekten)

  Der Spieler erhält Skill-Erfahrung für jedes Skill-Objekt in diesem Array.

- TradeDeals (array von TradeDeal-Objekten)

  Definiert vergünstigte oder anderweitig veränderte Gegenstandspreise im entsprechenden NPC-Shop.

6. Skill Objekte

Wird innerhalb des `Skills` Arrays in einem Reward-Objekt verwendet.

```
{ "Skill": "Cooking",
  "Experience": 50
}
```

- Skill (string)
  Muss einer der vordefinierten Skill-Namen sein: `"Archery"`, `"Aviation"`, `"Awareness"`, `"Boxing"`, `"Camouflage"`, `"Cooking"`, `"Demolition"`, `"Driving"`, `"Endurance"`, `"Engineering"`, `"Farming"`, `"Handgun"`, `"Medical"`, `"MeleeWeapons"`, `"Motorcycle"`, `"Rifles"`, `"Running"`, `"Sniping"`, `"Stealth"`, `"Survival"`, `"Tactics"`,oder `"Thievery"`.
- Erfahrung (integer)
  Anzahl der Erfahrungspunkte, die für diese Fertigkeit gewährt werden.

7. TradeDeal Objekte

Wird innerhalb des `TradeDeals` Arrays innerhalb eines Reward-Objekts verwendet.

```
{
  "Item": "Weapon_M9",
  "Price": 50,
  "Amount": 2,
  "AllowExcluded": false,
  "Fame": 10
}
```

- **Item** (string)
  Name des Gegenstands, wie ihr ihn im Spiel auf spawnen lassen könnt mittels `#SpawnItem`.

- **Price** (integer, optional)
  Der ermäßigte (oder geänderte) Preis. Lasst ihn weg, um den Standardpreis beizubehalten.

- **Amount** (integer ≥ 1, optional)
  Die verfügbare Menge des rabattierten Gegenstandes.

- **AllowExcluded** (boolean, optional)
  Ermöglicht den Kauf von Gegenständen, die normalerweise nicht vom Händler verkauft werden, wenn er auf `true` gestellt ist

- **Fame** (integer ≥ 0, optional)
  Ruhmpunkte-Voraussetzung für den Kauf des Gegenstands. Weglassen, um die Standard-Ruhmpunkte-Anforderungen zu verwenden.

8. Condition Objects

Das `Conditions` Array in einem Quest-JSON-Objekt besteht aus einem oder mehreren Condition-Objekten. Jede Bedingung kann einer von drei Typen sein: Eliminierung, Abruf oder Interaktion.

8.1 Gemeinsame Bedingungseigenschaften

```
{
  "Type": "Elimination",
  "CanBeAutoCompleted": false,
  "TrackingCaption": "Eliminate 3 puppets",
  "SequenceIndex": 0,
  "LocationsShownOnMap": [
    {
      "Location": { "X": 0.0, "Y": 0.0, "Z": 0.0 },
      "SizeFactor": 1.0
    }
  ]
}
```

- **Type** (string)
  Muss eines der folgenden sein: `"Elimination"`, `"Fetch"`, oder `"Interaction"`.

- **CanBeAutoCompleted** (boolean, default: `false`)
  Wenn `false`, muss der Spieler zum questgebenden NPC zurückkehren, um diesen Schritt abzuschließen. Wenn `true`, wird es automatisch abgeschlossen, sobald die Voraussetzungen erfüllt sind.

- **TrackingCaption** (string, optional)
  Was im Tagebuch oder Questlog des Spielers erscheint, um diesen Schritt zu beschreiben.
- **SequenceIndex** (integer ≥ 0)
  Legt die Reihenfolge fest, in der die Bedingungen aktiv werden.
    - Mindestens eine Bedingung muss `SequenceIndex: 0`.
    - Mehrere Bedingungen können sich denselben Index teilen, und alle müssen erfüllt sein, bevor die Quest zum nächsten Index weitergeht.

- **LocationsShownOnMap** (array of **MapLocation** objects, optional)
  Zeichnet Kreise auf der Karte für die Position(en) dieser Bedingung.

8.1.1 MapLocation Objects

Jedes Objekt in LocationsShownOnMap sieht wie folgt aus:

```
{"Location": { "X": 1000.0, "Y": 2000.0, "Z": 30.0 },
  "SizeFactor": 1.0}
```

Strg+C im Spiel zum Kopieren des Ortes funktioniert auch:

```
{ "Location": "{X=-157607.328 Y=-687586.562 Z=667.976|P=353.113800
Y=101.191971 R=0.000000}",
    "SizeFactor": 1.0}
```

- **Location**
  - Ein Objekt mit `X`, `Y`, `Z` Koordinaten in der Spielwelt.
- **SizeFactor** (number > 0.0)
  - Der Maßstab des gezeichneten Kreises (1,0 ist ~300 m Durchmesser).

8.2 Elimination Conditions

Wird für Tötungsanforderungen verwendet:

```
{
"Type": "Elimination",
  "TargetCharacters": [
    "Puppet",
    "Prisoner" ],
  "Amount": 5,
  "AllowedWeapons": [
    "BP_Weapon_M1911_C",
    "BP_Weapon_M9_C"]}
```

- **TargetCharacters** (array of strings)
  Listet auf, wer oder was eliminiert werden muss. Kann Spawn Namen enthalten, die in `#SpawnZombie` (z.B., `Puppet`) oder `#SpawnAnimal` genutzt werden. Spezielle Platzhalter:
  - `"Puppet"` (passt zu allen Puppets)
  - `"Prisoner"` (passt zu allen Spielcharakteren)
  - `"Razor"`, `"Sentry"`, `"SentryOld"`
  - `"ArmedNPC"` (passt zu allen bewaffneten NPCs)
- **Amount** (number ≥ 1)
  Die Anzahl der benötigten Kills.
- **AllowedWeapons** (array of strings, optional)
  Wenn vorhanden, werden **nur** Kills mit diesen Waffen gezählt. Weglassen, um jede Waffe zuzulassen.

8.3 Fetch Conditions

Dient der Abholung oder Zustellung von Gegenständen:

```
{
  "Type": "Fetch",
  "DisablePurchaseOfRequiredItems": false,
  "PlayerKeepsItems": false,
  "RequiredItems": [
    {
      "AcceptedItems": [ "Apple" ],
      "RequiredNum": 3,
      "RandomAdditionalRequiredNum": 2,
      "MinAcceptedItemUses": 1,
      "MinAcceptedCookLevel": "Raw",
      "MaxAcceptedCookLevel": "Cooked",
      "MinAcceptedCookQuality": "Poor",
      "MinAcceptedItemMass": 100.0,
      "MinAcceptedItemHealth": 50.0,
      "MinAcceptedItemResourceRatio": 20.0,
      "MinAcceptedItemResourceAmount": 50.0
    }
  ]
}
```

- **DisablePurchaseOfRequiredItems** (boolean)
  Wenn `true`, kann niemand Gegenstände kaufen, die in `RequiredItems` gelistet sind, während die Quest aktiv ist - nur relevant, wenn der Server über `BLOCK PURCHASE OF REQUIRED ITEMS` Einstellung auf ON gestellt ist.

- **PlayerKeepsItems** (boolean)
  Wenn `true`, Gegenstände werden nach Abschluss der Quest nicht aus dem Inventar des Spielers entfernt.

RequiredItems (array of **Item** objects)
Jedes **Item** Objekt gibt an, welche und wie viele Items gesammelt werden müssen.

8.3.1 Item Objects

Eigenschaften für jedes erforderliche Element:

- **AcceptedItems** (array of strings)
  Liste der Gegenstand Namen wie sie von `#SpawnItem`.

- **RequiredNum** (integer ≥ 1)
  Wie viele dieser Gegenstände müssen gesammelt werden?

- **RandomAdditionalRequiredNum** (integer ≥ 1, optional)
  Ein zufälliger zusätzlicher Betrag (bis zur angegebenen Zahl), der eventuell benötigt wird.

- **MinAcceptedItemUses** (integer ≥ 0, optional)
  Falls zutreffend, muss der Gegenstand mindestens so viele Verwendungsmöglichkeiten haben.

- **MinAcceptedCookLevel** / **MaxAcceptedCookLevel** (string, optional)
  Erlaubter Garstufenbereich für Lebensmittel. Gültige Stufen: `"Raw"`, `"Undercooked"`, `"Cooked"`, `"Overcooked"`, `"Burned"`.

- **MinAcceptedCookQuality** (string, optional)
  Niedrigste akzeptable Kochqualität: `"Ruined"`, `"Bad"`, `"Poor"`, `"Good"`, `"Excellent"`, `"Perfect"`.

- **MinAcceptedItemMass** (number ≥ 0.0, optional)
  In Gramm ausgedrückt, muss die Masse eines Gegenstands mindestens diesen Wert haben.

- **MinAcceptedItemHealth** (number from 0.0 to 100.0, optional)
  Mindestprozentwert für die Gesundheit des Gegenstands erforderlich.

- **MinAcceptedItemResourceRatio** (number from 0.0 to 100.0, optional)
  Wird für flüssigkeitshaltige Gegenstände (z. B. Wasserflaschen) verwendet. Muss mindestens diesen Prozentsatz der Flüssigkeit enthaltenMindestprozentsatz für den Gesundheitszustand des Artikels erforderlich..

- **MinAcceptedItemResourceAmount** (number ≥ 0.0, optional)
  Wird für flüssigkeitshaltige Gegenstände verwendet. Muss mindestens so viel Gramm haben.

8.4 Interaction Conditions

Dient zur Interaktion mit den auf der Karte platzierten Objekten (z. B. Umlegen von Schaltern, Sammeln von Notizen usw.).

Beschaffung von Standortdaten

Benutze `#Quests GetMeshInfo` im Spiel, während ihr ein Objekt betrachtet, um dessen Mesh-Namen, Instanz, Transformation usw. zu erfassen. Es kopiert ein JSON-Snippet in die Zwischenablage, den ihr direkt in die Quest-Datei einfügen könnt.

```
{ "Type": "Interaction",

  "Locations": [{
      "AnchorMesh": "/Game/World/SomeMap/BP_Switch.BP_Switch_C",
      "Instance": 4,
      "FallbackTransform": "X=123.456 Y=234.567 Z=10.0 Pitch=0
Yaw=0,Roll=0",
      "VisibleMesh":
"/Game/World/SomeMap/BP_SwitchModel.BP_SwitchModel_C" },
    {
      "AnchorMesh": "/Game/World/SomeMap/BP_Door.BP_Door_C"
    }],
  "MinNeeded": 1,
  "MaxNeeded": 2,
  "SpawnOnlyNeeded": true,
  "WorldMarkerShowDistance": 50}
```

- **Locations** (array of **Location** objects)
  Jeder definiert einen Objektplatz oder einen Interaktionspunkt in der Welt.

- **MinNeeded** / **MaxNeeded** (integers)
  Bestimmt nach dem Zufallsprinzip, mit wie vielen der angegebenen Ortsobjekte der Spieler interagieren muss. Wenn `MinNeeded = 1` und `MaxNeeded = 2`,wird das Spiel entscheiden, ob 1 oder 2 Interaktionen erforderlich sind.

- **SpawnOnlyNeeded** (boolean)
  Wenn `true`, wird nur die zufällig bestimmte Anzahl von Objekten erzeugt. Bei false werden alle Objekte erzeugt, aber der Spieler muss nur mit so vielen Objekten interagieren, dass die gewünschte Anzahl erreicht wird.

- **WorldMarkerShowDistance** (integer ≥ 0)
  Die Entfernung (in Metern), in der ein Marker in der Welt sichtbar ist und den Spielern den genauen Standort des Objekts anzeigt.

8.4.1 Location Objects

- **AnchorMesh** (string)
  Identifiziert das Objekt im Spiel auf der Karte.

- **Instance** (integer, optional)
  Wird verwendet, wenn sich mehrere Kopien desselben Netzes in unmittelbarer Nähe befinden.

- **FallbackTransform** (string)
  Positions-/Drehungsdaten werden verwendet, wenn `AnchorMesh` fehl schlägt (z.B., der Sandkastenmodus verwendet immer die `FallbackTransform`).

- **VisibleMesh** (string)
  Das anzuzeigende 3D-Modell.

9. Alles zusammenfügen: Beispiel Quest

Im Folgenden findet ihr ein einfaches, detailliertes Beispiel für eine benutzerdefinierte JSON-Datei. Speichert diese Datei als `MyFirstQuest.json` (oder jeder andere Name) im `Override` Ordner ab.

```
{
  "AssociatedNpc": "GeneralGoods",
  "Tier": 1,
  "Title": "General Goods trader's Special",
  "Description": "Collect apples for the General Goods trader and
get a small reward.",
  "TimeLimitHours": 24.0,
  "RewardPool": [{
      "CurrencyNormal": 100,
      "Fame": 5,
      "Skills": [{
          "Skill": "Cooking",
          "Experience": 20}],
      "TradeDeals": [
        {
          "Item": "Pineapple",
          "Price": 50,
          "Amount": 1,
          "Fame": 0 } ] } ],
  "Conditions": [{
      "TrackingCaption": "Gather apples",
      "SequenceIndex": 0,
      "CanBeAutoCompleted": false,
      "Type": "Fetch",
      "DisablePurchaseOfRequiredItems": false,
      "PlayerKeepsItems": true,
      "RequiredItems": [{
          "AcceptedItems": [
            "Apple_2"],
          "RequiredNum": 3,
          "MinAcceptedItemHealth": 50.0}],
      "LocationsShownOnMap": [{
          "Location": {
            "X": 1000.0,
            "Y": 2000.0,
            "Z": 50.0},
          "SizeFactor": 1.0} ]}]}
```

Erläuterung

1. **AssociatedNPC**
   Der Barkeeper bietet die Quest und die Belohnungen aus dem `TradeDeals` array im Barkeeper Shop an..

2. **Tier**
   Stufe 1 bedeutet eine einfache oder niedrigstufige Aufgabe.

3. **RewardPool**
   Der Spieler erhält 100 normale Währung, 5 Ruhm, 20 Kocherfahrung und kann eine M9 zu einem reduzierten Preis kaufen, sobald er die Quest abgeschlossen hat.

4. **Conditions**

   - **Typ** ist `Fetch`. Der Spieler muss 3 Äpfel mitbringen (`AcceptedItems`: `Apple`) mit mindestens 50% Gesundheit.
   - Der Spieler muss zum Barkeeper zurückkehren (`CanBeAutoCompleted`: false) um die Quest abschließen zu können.

   - `PlayerKeepsItems` ist `true`, damit die Äpfel nach der Fertigstellung der Quest nicht behalten werden können.

Nachdem ihr diese Datei erstellt oder geändert habt, müsst ihr euren Server oder euer Spiel neu starten, damit die Quest aktiv wird.

10. Tipps und bewährte Praktiken

- Bewahrt immer **Sicherungskopien (Backup)** eurer JSON-Dateien für den Fall auf, dass ihr Änderungen rückgängig machen müsst.
- **Validiert euer JSON** mit Online- oder lokalen JSON-Validatoren, um Formatierungsfehler zu vermeiden.
- Verwendet `#Quests GetMeshInfo` für genaue Platzierungsdaten für Bedingungen des Typs Interaktion.
- Testet eure Quest nach Möglichkeit zunächst im Sandbox-Modus, bevor ihr sie zu einer Live-Server-Umgebung hinzufügt.
- Denkt daran, dass ihr die Standard-Quests nicht bearbeiten könnt. Ihr könnt sie nur blockieren oder völlig neue Quests erstellen.

Abschluss

Wenn ihr diese Anleitung befolgt, solltet ihre in der Lage sein, eure eigenen Quests in SCUM erstellen, anpassen und verwalten zu können. Egal, ob ihr die Standardquests blockiert, neue Quests mit einzigartigen Bedingungen hinzufügt oder Spieler mit Währung, Ruhm, Rabatten oder Fertigkeitserfahrung belohnen möchtet, die oben beschriebene JSON-Struktur deckt alles Wesentliche ab.

Wichtig: Startet euren Server oder euer Spiel immer neu, nachdem Änderungen an den JSON-Dateien vorgenommen wurden, damit die benutzerdefinierten Quests korrekt geladen werden.

Viel Spaß beim Erstellen einzigartiger SCUM-Erlebnisse mit euren benutzerdefinierten Quests!