

ジェムリアTRPGダイスボット 統合マニュアル

最終更新日: 2026-01-22 対象バージョン: Current

目次

1. プレイヤーガイド
2. スキル定義と特記処理
3. バフ・ステータス定義
4. アイテム定義
5. 輝化スキル定義

00. プレイヤーガイド (Player Guide)

このマニュアルでは、「ジェムリアTRPGダイスボット」の基本的な遊び方、キャラクターの作成方法、およびバトルの流れについて解説します。

1. アプリへのアクセスと入室

アクセス手順

- 配布されたURL（またはローカル環境の <http://localhost:5000>）にブラウザでアクセスします。
- ログイン画面が表示されます。
 - あなたの名前:** セッションで使用する名前（PL名またはGM名）を入力してください。
 - あなたの属性:** プレイヤー(Player) か GM かを選択します。
 - 「ルームポータルへ進む」をクリックします。

ルーム選択

- ルーム一覧が表示されます。
- 新しいルームを作る場合:** 「新規ルーム作成」欄にルーム名を入力し、「作成」ボタンを押します。
- 既存のルームに入る場合:** 表示されているルームの「入室」ボタンを押します。
- 入室すると、メイン画面（バトルフィールド）が表示されます。

2. 画面の見方とキャラクター準備

メイン画面の構成

- 左カラム:** キャラクターリスト（味方/敵）、タイムライン
- 中央カラム:** アクションエリア（スキルの計算・宣言）
- 右カラム:** ログ、チャット、ルーム機能（セーブ/リセット）

キャラクターの追加

- 左上の「+ キャラクター追加」ボタンを押します。
- キャラクターロード画面が開きます。

- **JSON読み込み**: 作成したキャラクターデータ(JSON)を貼り付けて追加します。
 - **DB読み込み(GMのみ)**: 保存済みのプリセットから読み込みます。
3. 読み込まれたキャラクターは、初期状態では盤面外（未配置）になっている場合があります。GMが座標を設定するか、リストに表示されるのを確認してください。
-

3. バトルの流れ (ラウンド進行)

ラウンド開始 (イニシアチブ決定)

1. ラウンドの最初に、タイムライン上部の「**次R開始**」ボタンを押します。
2. 全キャラクターが「速度(**Speed**)」判定を行い、行動順（タイムライン）が決定されます。

手番の実行

タイムラインの一番上にいるキャラクターが手番プレイヤーとなります。

1. **アクションエリア（中央）**を操作します。
2. **使用者**: 自分のキャラクターが自動選択されます（GMは選択可能）。
3. **対象**: ターゲットを選択します。
4. **スキル**: 使用するスキルを選択します。
5. 「**威力計算**」ボタン: ダイスを振る前に、判定値や予測威力（2d6+5など）を確認します。
6. 「**宣言**」ボタン: 行動を確定し、全体に公開します。チャットログに結果が表示されます。

マッチ処理 (対決)

攻撃スキルを使用した場合、**マッチ（対決）**が発生します。

1. **攻撃側**: スキルを宣言済み。
2. **防御側**: 「対応側」パネルが有効になります。
 - **スキル**: 回避、防御、または反撃スキルを選択します。
 - 「**宣言**」: 防御行動を宣言します。
3. **マッチ開始**: 両者の宣言が揃うと、「**マッチ開始**」ボタンが表示されます（GMまたは攻撃側が押下）。
4. 自動的にダイスロールが行われ、命中・ダメージ計算結果がログに表示されます。HPなどは自動で変動します。

広域攻撃 (Wide Match)

「広域」タグのついたスキルを使用する場合、複数の敵を同時に攻撃できます。

1. スキルを選択すると、自動的に「広域攻撃モード」パネルが表示されます。
 2. 攻撃側が「宣言」を行います。
 3. 防御側全員がそれぞれ防御スキルを宣言します。
 4. 全員の宣言が完了したら、「**広域攻撃 実行**」ボタンを押して一括処理します。
-

4. キャラクター作成の手順

本ツールでは、キャラクターデータをJSON形式で管理しています。専用の「キャラクター作成ツール」を使用してJSONを生成します。

作成手順

1. [GEMDICEBOT_CharCreator.html](#) をブラウザで開きます。
2. **基本設定:** キャラクターネーム、HP、MPなどの基礎ステータスを入力します。
 - **Speed:** 行動順決定に使用するステータスです。
3. **スキル登録:**
 - スキルの名前、分類（物理/魔法）、属性、基礎威力などを入力します。
 - **特記処理:** 高度な効果（バフ付与など）が必要な場合は、JSON形式で記述します（作成ツールの支援機能を利用してください）。
4. **データ出力:**
 - 入力が完了したら「**JSON出力**」ボタンを押します。
 - 表示されたテキストをコピーし、ダイスボットツールの「キャラクター追加」画面に貼り付けます。

データの保存

作成したJSONテキストは、メモ帳などに保存しておくことを強く推奨します（ブラウザのキャッシュが消えると復元できないため）。

スキル (Skills) 定義マニュアル

`skill_data.json` における通常のスキル定義および「特記処理 (Special Effects)」の詳細マニュアルです。

JSON構造

```
"E-00": {
    "スキルID": "E-00",
    "チャットパレット": "2+1d2+1d{物理補正} 【E-00 戻る】",
    "デフォルト名称": "戻る",
    "分類": "物理",
    "距離": "近接",
    "属性": "打撃",
    "取得コスト": "0",
    "基礎威力": "2",
    "ダイス威力": "+1d2",
    "使用時効果": "",
    "特記": "",
    "発動時効果": "",
    "特記処理": "{...}", // JSON文字列として記述
    "tags": ["攻撃"]
}
```

- **特記処理:** JSON形式の文字列として記述する必要があります。エスケープ処理に注意してください。
- **tags:** `"攻撃"`, `"守備"`, `"広域"`, `"即時発動"`, `"宝石の加護スキル"` など。

特記処理 (Special Effects) の構造

特記処理フィールドは、内部的には以下のJSON構造を持ちます。

```
{  
  "tags": ["攻撃"],  
  "power_bonus": [],  
  "cost": [{"type": "MP", "value": 5}],  
  "effects": []  
}
```

1. Cost (コスト定義)

スキルの使用コストを定義します。（システム的には表示用ですが、将来的に自動減算に対応可能です）

```
"cost": [  
  {"type": "MP", "value": 5},  
  {"type": "MP", "value": 5},  
  {"type": "FP", "value": 2},  
  {"type": "HP", "value": 10}  

```

- **type:** MP, FP, TP に加え、 HP も指定可能です。
 - **HPコストの挙動:** 現在HPを消費します。HPが足りない場合は発動できません（戦闘不能にはなりません）。

2. Power Bonus (威力ボーナス)

条件に応じて基礎威力を加算するルールです。

```

"power_bonus": [
  {
    "source": "target",           // 参照先 ("self", "target")
    "param": "戦慄",             // 参照パラメータ
    "operation": "PER_N_BONUS",   // 計算ルール
    "per_N": 5,                  // Nごとに
    "value": 1,                  // +1
    "max_bonus": 2              // 最大+2まで
  }
]

```

- Operations:

- PER_N_BONUS: パラメータ N ごとに value 加算
- FIXED_IF_EXISTS: パラメータが1以上あれば value 加算
- MULTIPLY: パラメータ × value_per_param を加算

3. Effects (発動効果)

スキル命中時や使用時に発動する効果のリストです。

共通フィールド

- timing: 発動タイミング
 - HIT: 命中時
 - WIN: マッチ勝利時
 - LOSE: マッチ敗北時
 - PRE_MATCH: マッチ開始前 (即時発動など)
 - END_ROUND: ラウンド終了時
 - END_MATCH: マッチ終了時
 - UNOPPOSED: 一方攻撃 (相手が防御・回避を持たない、または行動不能時) の攻撃時
- target: 効果対象
 - self: 自分自身
 - target: 選択した対象
 - NEXT_ALLY: タイムライン上で自分の次に行動する味方
 - ALL_ENEMIES: 敵全体 (ターゲット選択不要)
 - ALL_ALLIES: 味方全体 (自分含む)
 - ALL: 敵味方全員
- condition: 発動条件 (任意)

効果タイプ (Type)

APPLY_STATE (状態異常付与) ステータスや状態異常値を加算します。

```
{
  "type": "APPLY_STATE",
  "state_name": "出血",
  "value": 3
}
```

APPLY_BUFF (バフ付与) バフ図鑑のバフ、または動的バフを付与します。

```
{
  "type": "APPLY_BUFF",
  // ID指定: カタログや特殊ロジック(Bu-07等)を利用する場合は必須
  "buff_id": "Bu-01",
  // 名前指定: 動的バフを指定する場合、または既存IDの表示名を上書きする場合に使用
  "buff_name": "Power_Atk5",
  "lasting": 1,           // 持続ラウンド (省略時: 1)
  "delay": 0,             // 発動遅延ラウンド (省略時: 0)
  "flavor": "力が湧く！", // フレーバーテキストの上書き/設定 (任意)
}
```

設定例: 表示名の変更と時限発動 Bu-07 (時限破裂) のロジックを使いつつ、名前を「時限式魔力爆弾」にし、3ターン後に爆発・消滅させる例:

```
{
  "type": "APPLY_BUFF",
  "buff_id": "Bu-07",
  "buff_name": "時限式魔力爆弾",
  "delay": 3,    // 3ラウンド後に発動
  "lasting": 0   // 発動と同時に消滅
}
```

APPLY_STATE_PER_N (動的付与) パラメータに応じて付与量を変動させます。

```
{  
    "type": "APPLY_STATE_PER_N",  
    "state_name": "亀裂",  
    "source": "self",  
    "source_param": "戦慄",  
    "per_N": 2,  
    "value": 1  
}
```

MULTIPLY_STATE (状態異常値の乗算) 対象の状態異常値（スタック数）を指定した倍率で乗算します。結果は標準的な四捨五入（0.5は切り上げ）で整数に丸められます。

```
{  
    "type": "MULTIPLY_STATE",  
    "state_name": "出血",  
    "value": 2.0, // 2倍にする  
    "target": "target"  
}
```

- **value:** 乗算する倍率（浮動小数点数）。
 - 2.0: 2倍（例: 5 -> 10）
 - 0.5: 半分（例: 5 -> 3, 4 -> 2）
 - 0: 消去（0にする）

CUSTOM_EFFECT (特殊効果) プラグインとして実装された特殊な処理を実行します。

```
{  
    "type": "CUSTOM_EFFECT",  
    "value": "破裂爆発"  
}
```

使用可能な CUSTOM_EFFECT 一覧:

値 (Value)	説明	実装
破裂爆発	対象の破裂威力に応じたダメージを与え、破裂を消費する。	BurstEffect
出血氾濫	対象が持つ出血威力だけダメージを与える。	BleedOverflowEffect
戦慄殺到	戦慄威力に応じてMP減少や行動不能を付与する。	FearSurgeEffect
荊棘飛散	荊棘威力に応じて他対象に荊棘を拡散する。	ThornsScatterEffect
亀裂崩壊_DAMAGE	亀裂による追加ダメージ処理（通常は自動計算だが強制発動用）。	FissureEffect
APPLY_SKILL_DAMAGE AGAIN	同じスキルのダメージ処理をもう一度実行する（連撃）。	SimpleEffect

その他の効果タイプ (Advanced Types)

APPLY_STATE_PER_N (定数比例付与) 「自分の【ステータスA】Xにつき、【ステータスB】をY与える」といった効果を実現します。

```
{
  "type": "APPLY_STATE_PER_N",
  "state_name": "亀裂",
  "source": "self",
  "source_param": "戦慄",
  "per_N": 2,
  "value": 1,
  "max_value": 5
}
```

MULTIPLY_STATE (状態異常値の乗算) 対象の状態異常値（スタック数）を指定した倍率で乗算します。

```
{  
    "type": "MULTIPLY_STATE",  
    "state_name": "出血",  
    "value": 2.0,  
    "target": "target"  
}
```

ランダムターゲット選定 (Random Target) `effects` 内のフィールドとして記述することで、効果対象をランダムに決定します。

- `target_select`: "RANDOM"
- `target_filter`: "ENEMY", "ALLY", "ALL"
- `target_count`: 選択数

```
{  
    "type": "APPLY_STATE",  
    "target_select": "RANDOM",  
    "target_filter": "ENEMY",  
    "target_count": 2,  
    "state_name": "出血",  
    "value": 3  
}
```

フレーバーテキスト (Flavor Text) バフ付与などのログに演出テキストを追加します。

```
{  
    "type": "APPLY_BUFF",  
    "buff_name": "勇気の印",  
    "flavor": "「負ける気がしない！」心に勇気が湧いてくる."  
}
```

5. 条件付き効果 (Conditional Effects)

特定のスレッシュホールド（閾値）や状態異常の有無に応じて、効果を発動するかどうかを判定します。すべての効果タイプ（`APPLY_STATE`, `APPLY_BUFF`, `DAMAGE_BONUS` 等）に対し、`condition` プロパティを追加することで制御できます。

定義例: 「出血」している相手に追加FP

```
{  
    "type": "APPLY_STATE",  
    "state_name": "FP",  
    "value": 1,  
    "target": "self",  
    "condition": {  
        "source": "target",  
        "param": "出血",  
        "operator": "GTE",  
        "value": 1  
    }  
}
```

パラメータ詳細

- **source:** `self` (自分), `target` (相手)
- **param:** 参照するステータス名 (`HP`, `MP`, `戦慄`, `出血` など)
- **operator:**
 - `GTE`: 以上 (\geq)
 - `LTE`: 以下 (\leq)
 - `GT`: より大きい ($>$)
 - `LT`: より小さい ($<$)
 - `EQUALS`: 等しい (\equiv)
 - `CONTAINS`: (タグなどが)含まれる
- **value:** 比較する値

6. カウンター・スタックシステム

特定の「スタック（蓄積値）」をリソースとして消費・参照するメカニクスです。独自のステータス（State）を定義し、それを参照・操作することで実現します。

実装パターン

1. **スタックの付与:** `APPLY_STATE` で独自のステータス名を指定（例: "魔力", "チャージ"）。
2. **スタックによる強化:** `power_bonus` や `APPLY_STATE_PER_N` でそのステータスを参照。

定義例: 「魔力」スタックに応じて攻撃力アップ

```
"特記処理": {  
    "power_bonus": [  
        {  
            "source": "self",  
            "param": "魔力",  
            "operation": "MULTIPLY",  
            "value_per_param": 2.0  
        }  
    ]  
}
```

- 上記例では「魔力」1につき威力+2されます。

定義例: 「魔力」を3つ消費して強力な効果

```
"特記処理": {  
    "cost": [  
        {  
            "type": "魔力",  
            "value": 3  
        }  
    ],  
    "effects": [...]  
}
```

バフ・ステータス (Buff & Status) 総合マニュアル

本システムにおけるバフ・デバフ、および状態異常は、**スプレッドシート定義**、**動的定義 (Naming Patterns)**、**静的定義(Code)** の3つのレイヤーで管理されています。

1. データの優先順位

バフ名が参照された際、システムは以下の順序で定義を検索します。

1. 静的定義 (Static):

- Pythonコード(`manager/buff_catalog.py`)に直接記述された複雑なバフ。
- 例: `背水の陣` など。
- 最も優先度が高い。

2. スプレッドシート定義 (Spreadsheet):

- Google Spreadsheet で管理されているバフ図鑑。
- 一般的なバフの追加・編集はここで行います。
- キャッシュファイル: `buff_catalog_cache.json`

3. 動的定義 (Dynamic Patterns):

- バフ名が特定のパターン (例: `AttackUp_Atk10`) に一致する場合、自動的に効果を生成します。
- 定義ファイルが存在しなくても機能します。

2. スプレッドシート定義 (Spreadsheet)

基本となるバフ定義方法です。

必須カラム構成

カラム名	説明	備考
バフID	システム管理用ID	例: Bu-001
バフ名称	ゲーム内での表示名	一意である必要があります
JSON定義	効果内容を記述するJSON	後述の構成またはPlugin定義を記述
持続ラウンド	バフの持続時間	デフォルト: 1
バフ説明	マウスオーバー時の説明文	
フレーバーテキスト	演出用テキスト	

JSON定義の書き方

JSON定義 カラムには、以下の形式でJSONオブジェクトを記述します。

A. ステータス補正型 (Simple Stat Mod)

最も基本的な形式です。

```
{
  "type": "stat_mod",
  "stat": "基礎威力",
  "value": 5
}
```

- **type**: "stat_mod" (固定)
- **stat**: 対象パラメータ (**基礎威力**, **物理補正**, **魔法補正**, **ダイス威力**)
- **value**: 変動値 (マイナスでデバフ)

B. プラグイン型 (Plugin Effect)

特殊な挙動をするバフです。実装済みのプラグイン名を指定します。

```
{
  "type": "plugin",
  "name": "provoke",
  "category": "debuff"
}
```

利用可能な主要プラグイン一覧 ([plugins/buffs/](#)):

プラグイン名 (<code>name</code>)	効果概要	パラメータ
<code>provoke</code>	挑発: 敵のターゲットを自身に固定	なし
<code>confusion</code>	混乱: 被ダメージ倍加 + 行動阻害	<code>damage_multiplier</code> (float), <code>restore_mp_on_end</code> (bool)
<code>immobilize</code>	行動不能: アクション不可	なし
<code>dodge_lock</code>	再回避ロック: 回避のみ可能、攻撃不可	なし
<code>timebomb_burst</code>	時限破裂: 終了時に破裂を与える	なし
<code>burst_no_consume</code>	破裂消費無効: 破裂消費なしで発動可	なし

C. イベントフック型 (Event Hooks)

特定のゲームイベント発生時に発動する特殊効果です。これらは `effects` 配列ではなく、ルートレベルのプロパティとして定義します。

```
{  
    "on_death": [  
        { "type": "APPLY_STATE", "target": "ALL_ENEMIES", "state_name": "呪  
い", "value": 1 }  
    ],  
    "battle_start_effect": [  
        { "type": "APPLY_STATE", "target": "self", "state_name": "FP",  
        "value": 1 }  
    ]  
}
```

- **on_death**: HPが0以下になった瞬間に発動します。
- **battle_start_effect**: 戦闘開始時（リセット時）およびキャラクター配置時に発動します。

3. 動的定義 (Dynamic Patterns)

特定の命名規則に従うバフは、定義ファイルを作成しなくても自動的に効果が発動します。

命名規則一覧

[任意の接頭辞] + [キーワード] + [数値] の形式で記述します。例: **AttackBoost_Atk10** (名
称: **AttackBoost_Atk10**, 効果: 攻撃威力+10)

キーワード	効果	例	意味
_Atk	攻撃タグを持つスキルの威力UP	Power_Atk10	攻撃威力+10
_AtkDown	攻撃タグを持つスキルの威力DOWN	Weak_AtkDown5	攻撃威力-5
_Def	守備タグを持つスキルの威力UP	Shield_Def10	守備威力+10
_DefDown	守備タグを持つスキルの威力DOWN	Break_DefDown5	守備威力-5
_Phys	物理補正UP (ダイスロール修正)	Str_Phys2	物理補正+2
_PhysDown	物理補正DOWN	Weak_PhysDown2	物理補正-2
_Mag	魔法補正UP	Int_Mag2	魔法補正+2
_MagDown	魔法補正DOWN	Dull_MagDown2	魔法補正-2
_DaIn	被ダメージ倍率増加 (%)	Paper_DaIn20	被ダメ1.2倍 (20%増)
_DaCut	被ダメージカット率 (%)	Hard_DaCut20	被ダメ0.8倍 (20%減)
_Crack	亀裂付与量UP (消費しない)	Earth_Crack2	亀裂付与+2
_CrackOnce	亀裂付与量UP (1回消費)	Quake_CrackOnce3	次の亀裂付与+3 (使用後消滅)
_BleedReact	被弾時、自身に出血付与	Blood_BleedReact2	被弾するたび出血+2

4. バフプラグイン詳細 (Buff Plugin Details)

特定の `buff_id` を指定することで発動する、スクリプト制御された高度な特殊効果の詳細です。

4-1. 破裂威力減少無効 (Burst Conservation)

- ID: `Bu-06`

- **効果:** このバフを持つキャラクターが【破裂爆発】を受ける際、本来消費される【破裂】の値が減少しなくなります。
- **用途:** 「次の一撃だけ破裂を消費したくない（破裂2倍ダメージ用）」などのコンボに使用します。

4-2. 時限式破裂爆発 (Timed Burst)

- **ID:** Bu-07
- **効果:** バフの delay カウントが0になった瞬間、自分自身に対して【破裂爆発】が発動します。
- **動作:** 蓄積されている【破裂】値を参照してダメージを与えます。
- **設定:** APPLY_BUFF で delay (発動までのラウンド) と lasting: 0 (発動後即消滅) を設定して使用します。

4-3. 龜裂崩壊連鎖 (Dual Collapse / Burst Chain)

- **概要:** 「亀裂崩壊」発生時に、さらに別の効果（破裂爆発など）を誘発させることができます。
- **実装方法:** fissure.py プラグインの設定により、亀裂消費時に triggered_effect を指定して連鎖させることができます。（※要上級者向け設定）

4-4. 出血維持 (Bleed Maintenance)

- **ID:** Bu-08
- **効果:** このバフを持つキャラクターは、ラウンド終了時に【出血】の値が半減しません（現在の値を維持します）。
- **用途:** 出血特化のビルトや、長期的なDoT（継続ダメージ）戦略に使用します。

5. 特殊なバフ定義 (Advanced)

条件付きバフ

`stat_mod` よりも複雑な条件（HP10以下のときのみ等）を設定したい場合、JSON内の `effect` を拡張できますが、現在はスプレッドシート上のJSON定義よりも、

`manager/buff_catalog.py` の `STATIC_BUFFS` に直接Pythonコードとして記述することができます。

```
# manager/buff_catalog.py の例
"背水の陣": {
    "power_bonus": [
        "condition": { "source": "self", "param": "HP", "operator": "LTE", "value": 10 },
        "operation": "FIXED",
        "value": 5
    ]
}
```

フレーバーテキストと表示名のカスタマイズ

スキル等の `APPLY_BUFF` アクションにおいて、定義元のバフ設定を上書きすることができます。

A. フレーバーテキストの追加・上書き

`flavor` キーを指定することで、動的バフや既存バフに演出テキストを追加できます。

```
{
    "type": "APPLY_BUFF",
    "buff_name": "Power_Atk5",
    "flavor": "全身に力がみなぎる！"
}
```

B. 表示名とロジックの分離 (Name Overriding)

`buff_id` と `buff_name` を同時に指定することで、「効果は特定のIDのものを使いつつ、表示名だけ変える」ことが可能です。

```
{  
    "type": "APPLY_BUFF",  
    "buff_id": "Bu-07",  
    "buff_name": "時限式魔力爆弾",  
    "delay": 3,  
    "lasting": 0  
}
```

この例では、システム上は **Bu-07** (時限破裂爆発) として処理されますが、ログや画面には「時限式魔力爆弾」と表示されます。

時限発動バフの設定 (Timebomb Configuration)

「3ラウンド後に爆発する」といった時限式のバフ（例: **Bu-07** 時限破裂爆発）を設定する場合、**delay** と **lasting** の使い分けが重要です。

- **delay**: 発動までの待機ラウンド数。この値が 0 になった瞬間に効果（ダメージ等）が発動します。
- **lasting**: バフアイコンの表示期間。

推奨設定: 爆発と同時にバフを消滅させたい場合は、**lasting: 0** に設定してください。

```
// 3ラウンド後に爆発し、直後に消滅する設定  
{  
    "type": "APPLY_BUFF",  
    "buff_id": "Bu-07",  
    "delay": 3,          // カウントダウン  
    "lasting": 0        // 完了後即消滅  
}
```

アイテム (Items) 定義マニュアル

消費アイテムや、使用することで効果を発揮するアイテムの定義です。

ファイル構造

- ファイル名: `items_cache.json` (キャッシュとして生成)
- 形式: JSON Object (Key: Item ID)

フィールド定義

フィールド	型	必須	説明	備考
<code>id</code>	string	<input checked="" type="checkbox"/>	アイテムID	例: "I-00"
<code>name</code>	string	<input checked="" type="checkbox"/>	アイテム名	
<code>description</code>	string	<input checked="" type="checkbox"/>	効果説明文	UI表示用
<code>flavor</code>	string		フレーバーテキスト	UI表示用
<code>consumable</code>	boolean	<input checked="" type="checkbox"/>	消費フラグ	<code>true</code> なら使用後に個数が減る
<code>usable</code>	boolean	<input checked="" type="checkbox"/>	使用可能フラグ	<code>true</code> ならアイテム欄から使用可能
<code>round_limit</code>	integer		ラウンド使用制限	1Rあたりの使用回数 (-1は無制限)
<code>effect</code>	object	<input checked="" type="checkbox"/>	効果定義	<code>heal</code> または <code>buff</code> タイプ

Effect Object の種類

1. 回復 (Heal)

HPやMPを回復します。

```
"effect": {  
    "type": "heal",  
    "target": "single",  
    "hp": 15  
}
```

- `type`: "heal"
- `target`: 現在は "single" (単体) のみ対応
- `hp`: HP回復量 (任意)
- `mp`: MP回復量 (任意)

2. バフ付与 (Buff)

バフ図鑑に定義されたバフを付与します。

```
"effect": {  
    "type": "buff",  
    "target": "single",  
    "buff_id": "Bu-00"  
}
```

- `type`: "buff"
- `target`: "single"
- `buff_id`: `buff_catalog_cache.json` で定義されているバフIDを指定

定義例

HP回復ポーション

```
"I-00": {
    "id": "I-00",
    "name": "ギルド印の安物ポーション",
    "description": "味方1人のHPを15回復する。",
    "flavor": "パン一切れくらいの値段で売っている粗悪な回復薬。",
    "consumable": true,
    "usable": true,
    "round_limit": -1,
    "effect": {
        "type": "heal",
        "target": "single",
        "hp": 15
    }
}
```

バフ付与アイテム

```
"I-01": {
    "id": "I-01",
    "name": "鋭敏の魔力薬",
    "description": "味方1人を選び、1ラウンドの間スキルの基礎威力を+1する。",
    "consumable": true,
    "usable": true,
    "round_limit": 1,
    "effect": {
        "type": "buff",
        "target": "single",
        "buff_id": "Bu-00"
    }
}
```

輝化スキル (Radiance Skills) 定義マニュアル

輝化スキルはキャラクターのステータス (HP, MP) を永続的に強化するパッシブスキルです。

ファイル構造

- ファイル名: `radiance_skills_cache.json` (キャッシュとして生成)
- 形式: JSON Object (Key: Skill ID)

フィールド定義

フィールド	型	必須	説明	備考
<code>id</code>	string	<input checked="" type="checkbox"/>	スキルID	例: "S-00"
<code>name</code>	string	<input checked="" type="checkbox"/>	スキル名	
<code>cost</code>	integer	<input checked="" type="checkbox"/>	習得コスト	現在はシステム的には未使用(1固定)
<code>description</code>	string	<input checked="" type="checkbox"/>	効果説明文	UI表示用
<code>flavor</code>	string		フレーバーテキスト	UI表示用
<code>effect</code>	object	<input checked="" type="checkbox"/>	効果定義	<code>STAT_BONUS</code> タイプ
<code>duration</code>	integer		持続時間	パッシブのため通常 <code>-1</code> (無限)

Effect Object (Stat Bonus)

ステータスを恒久的に増加させる効果です。

```
"effect": {  
    "type": "STAT_BONUS",  
    "stat": "MP",  
    "value": 1  
}
```

- **type**: 固定値 "STAT_BONUS"
- **stat**: 対象ステータス ("HP" または "MP")
- **value**: 増加させる値 (数値)

定義例

```
"S-00": {  
    "id": "S-00",  
    "name": "魔力の解放",  
    "cost": 1,  
    "description": "MPの上限を1上げる。",  
    "flavor": "幾度も魔力を費やす経験を積み、精神の限界を遠ざけることに成功した証。",  
    "effect": {  
        "type": "STAT_BONUS",  
        "stat": "MP",  
        "value": 1  
    },  
    "duration": -1  
}
```