

# LayerWeaver：プロフェッショナル技術仕様書 (v16.0)

---

～ 内部アルゴリズム、命名規則、物性設定、および基幹データ構造の定義 ～

最終更新：2026/04/20

本ドキュメントは、LayerWeaver（Krita Plugin）の全技術仕様をまとめた技術仕様書である。エンジニアが本紙を参照することでシステムを適切に制御・拡張し、クリエイターがその内部論理を理解して自由な作画リギングを実現することを目的とする。

---

## 1. システム・コンセプトロジック：独立独歩と統合の美学

Weaverシリーズは、シリーズ全体での「データのバトンリレー」による究極の自動化を実現する一方で、各ツールが単体でもプロフェッショナルな実戦投入を可能にする「独立独歩」の設計思想を持つ。

### 1.1 LayerWeaver（単体運用）：効率的なリギングアセットの生成

クリエイターの描いた「一枚の絵」から、Cartoon Animator 5 (CTA5) 用のG3キャラクターを直接生成する。

- **即時性:** 書き出されたPSDをCTA5にドラッグ＆ドロップするだけで、ボーン配置とリギングが完了。即座にアニメーション制作へ移行できる。
  - 注：G3テンプレート規格はReallusion社が設計した既存の業界標準規格であり、LayerWeaverはその規格に適合するようデータを自動整理・最適化する役割を担う。
- **汎用性:** 人体モデル（Human）のみならず、難易度の高い四足歩行モデル（Quadruped）にも対応。

### 1.2 MotionWeaver（単体運用）：モーションの論理的な変換

TDPT、Poser、あるいはインターネット上のBVHファイルを、あらゆるキャラクター構造へ適合させる。

- **形状変換:** カメラ目線に合わせた「俯瞰（High-angle）」や「煽り（Low-angle）」形状へのトリミング・投影を自在に実行。
- **リターゲット:** 構造の異なるキャラクター間でのBVH変換（レターゲットイング）を行い、資産の再利用性を極限まで高める。

### 1.3 統合運用：Weaverエコシステムによる自動リギング

- **Krita (LayerWeaver):** 「画素（Island）」と「命名（Tag）」を設計し、論理的なPSDを構築する。
  - **MotionWeaver (Hub):** 3Dモーション（BVH）を2.5Dキャンバスの座標系（Z値含む）へと変換する。
  - **RigWeaver (Moho Script):** PSDの命名ルールと投影データを読み込み、自動リギングを完遂する。
- 

## 2. 基幹ファイル物理仕様（The Lawbook）

Weaverの汎用性は、プログラム（不変）とデータ（可変）を完全に分離した「設定ベース」の設計にあります。

### 2.1 islands\_\*.csv（物理特定・マッピングの定義）

Krita上で抽出された無名の「島 (Island)」を、どの部位 (Island ID) として扱うかを特定するための辞書データである。

- **特定アルゴリズム (Mapping Logic) :**
  - **二点間距離フィルタ (Strong):** リストに2つのマーカーが記述されている場合、その2点を中心とした検索半径内に存在する島を優先的に特定する。これはパーツの「端点 (関節)」を指定することで最も強力に機能する。
  - **包含スコアリング (Soft):** マーカーが島の内側 (5pxのバッファを含む) にあるかを判定し、一致率をスコア化する。
  - **L/R自動判定:** 腰 (Hip/Pelvis) と頭 (Head/Neck) を結ぶ正中線に基づき、左右の島を自動的に選別する。
- **実戦の知恵:** パーツを「挟み込む」ように配置された2つの主要な関節マーカーを設定するのが最も精度が高い。特定に十分な境界マーカーがあれば、内側の全関節を記述する必要はなく、あえて絞ることによって計算負荷の軽減と誤判定の防止に繋がる。

## 2.2 rigging\_map\_\*.csv (論理配線と接続の定義)

「どの画素 (Island)」と「どの関節 (Marker)」を組み合わせ、最終的な「ボーン付きレイヤー」を構築するかを記した設計図である。

- **接続の論理 (Connectivity) :** 異なる行 (レイヤー) 間で共通の関節マーカーを記述することは、システム上で\*\*「マーカーの共有=物理的な接続 (ピボットの同期)」\*\*を定義することを意味する。
  - **例:** UpperArm の行に Shoulder, Elbow を含め、LowerArm の行に Elbow, Wrist を含めることで、両レイヤーは Elbow を共有の関節として結合される。
- **データ構造:** 最終レイヤー名, 参照島名, 担当ジョイント範囲
- **レイヤー重なり順 (Layer Ordering) :** 出力されるPSDのレイヤーの重なり順 (Z順) は、Krita上のマーカーのレイヤー順や islands.csv の順序ではなく、\*\*rigging\_map\_\*.csv の行の記述順 (および確定時のテーブル上の並び順) \*\*に直結する。
  - **上の行ほど:** PSD内では「手前 (前面)」に配置される。
  - **下の行ほど:** PSD内では「奥 (背面)」に配置される。
  - **設計の指針:** キャラクターのポーズ (向き) に合わせ、手前にあるべき部位をリストの上部に記述することで、直感的な前後関係の構築が可能となる。通常はTポーズを前提としたデフォルトの並び順 (rigging\_map.csv) のままで問題ない。
- **多連装ボーン (チェーン) の記述例:** Tail1, Island\_Tail, j5, j6, j7, j8, j9 このように複数の関節を列挙することで、一本の島に多段関節 (ボーンチェーン) を仕込むことができる。

## 2.3 templates.json (マスターカタログ・所在定義)

Weaverシリーズ全体のファイルパス、およびプロファイルごとの物理規定値を紐付ける「中央インデックス」である。

- **物理定義のパス:**
  - **islands\_csv / rigging\_map\_csv:** プロファイルごとに使用する解析辞書ファイルの所在を定義する。
  - **image\_folder / image\_folder\_head:** 出力される各パーツの保存先フォルダ名の定義。
- **解析規定値 (system\_params) :**
  - 島の分離やマーカー判定に使用するデフォルトの閾値 (def\_island\_thresh 等) を定義する。これらはプロジェクトフォルダ内の config.csv によって上書き (オーバーライド) が可能で

ある。

### 3. 論理制御と自動構築 (Logical Management)

Weaverシリーズが、Kritaで描かれた二次元の画素集合を、いかにして知的なボーンキャラクターへと構築するかの「アルゴリズムの仕組み」である。

#### 3.1 骨格トポロジーと接続の定義

キャラクターの物理的なマーカー（関節）の有無に関わらず、システムが想定する「骨格の完成形」を定義する。

- **target\_hierarchy (骨格トポロジー):**
  - キャクターの親子関係を定義した絶対的な設計図。RigWeaverはこのトポロジーを参照してMohoのボーン階層を構築する。
- **virtual\_bones (仮想ボーン):**
  - 物理的なマーカーが存在しない位置（例: 胸椎や首の連結部）に論理的な親を定義する。3Dモーションを2Dリグの文法へ正確に「翻訳」するための連結器として機能する。
- **flexi\_binds (しなやかな結合):**
  - 「一つの画像レイヤーが、複数のボーンの動きに追従する」ための命令。尻尾やマントなど、有機的なしなりが必要な部位に適用される。

#### 3.2 描画順の絶対制約 (ordered\_sub\_face\_groups)

LayerWeaver (Krita) がレイヤーを自動構築する際の「奥行きの定義」である。

- システムは、このリストに記載された順（例: **Ear** → **Jaw** → **Eye**）にグループをあらかじめ作成する。これにより、ユーザーが個別に順序を設定することなく、常に正しい前後関係でキャラクターが組み立てられる。

#### 3.3 レイヤー・ルーティングと自動タグ付け

レイヤー名に含まれるキーワードに基づき、システムが自動的に部位を判別して格納する「配送センター」の役割を担う。

- **face\_bone\_keywords:** **eye**, **mouth**, **brow** 等が含まれるレイヤーを自動的に「顔パーツ・グループ」へ格納。
- **export\_policy:** 顔パーツ等のボーンを特定のターゲットグループ（**Center** 等）に集約するかどうか等の、出力時の挙動を制御。

#### 3.4 整合性保護 (mismatch\_indicators)

解析中に実行される「検品（バリデーション）ロジック」である。

- 現在のプロファイルには含まれないはずのマーカー（例: 四足モデルなのに **torso** マーカーがある）を検知した場合、ユーザーに警告を出す。これにより、重大なプロファイル設定ミス未然に防止する。

#### 3.5 半透明オブジェクトの統合と関節検出仕様

スカーフやベールなどの半透明な装飾品を特定のパーツに登録するために、マーカーで領域を塗りつぶし、関節（Hole）として検出させる機能が仕様としてサポートされている。

- **アルファしきい値の無効化:** 島（**islands**）のセグメンテーション（部位分割）検出においては **templates.json** で指定された **alpha\_threshold** が適用されるが、関節（**joints**）の検出においては **アルファしきい値が 0 に固定される**。これにより、マーカーの下に位置するアートワークが極めて薄い半透明であっても、絵が存在する（アルファ値 > 0）限り確実にジョイント領域として抽出される。
- **ベースパーツへの統合:** 抽出された半透明領域（**j** 番号）を、UI上の対応表（または **rigging\_map.csv**）で胴体（**Torso** 等）などのベースパーツの **Joints** 列に指定することで、最終構築時にその半透明の画素データが親パーツの画素にマージされて出力される。

## 4. 命名による設定ルール（Tagging Logic）

レイヤー名に含まれる記号は、リギングエンジンに対する「直接的な動作命令」である。

### 4.1 「@」 タグ：空間吸着とバリエーション展開

イラストレーターを「正確な座標合わせ」の手間から解放するための、磁石のような機能である。

- **基本的な書き方:** 子パーツ名 @ 吸着先アンカー名
  - 例: **OpenEye @ Head, Hand\_A @ Wrist**
- **グループレイヤーへの命名（推奨）:** 個別のレイヤーにひとつずつ @ を付けることも可能だが、複数のバリエーション（瞬き、口の形など）を一つの **グループレイヤー** にまとめ、そのグループ名に **@吸着先** を付与するのが最も効率的である。
  - **効果:** グループ内の全レイヤーに一括で吸着設定が適用され、出力先では「スイッチレイヤー」や「バリエーション群」として整理される。
- **リギングへの影響（Layer Binding）:** @ で指定されたグループ（またはレイヤー）は、後続のリギングエンジン（RigWeaver等）によって指定のボーンへ\*\*「レイヤーバインド（一括固定）」\*\*される。これにより、激しい動きでもパーツがバラけない強固なリグが完成する。

### 4.2 「>」 タグ：動的な階層構築命令

Krita上の命名だけで、出力後のPSD内に複雑なボーンフォルダ階層（親子関係）を構築する。

- **基本的な書き方:** 親ボーン名 > 子グループ名 > 子ボーン名
  - 例: **Hip > Scarf\_Root(0) > Scarf\_Mid(0) > Scarf\_Tip(0)**
- **効果:** このように命名するだけで、出力時にボーンの親子関係が物理的に連結される。CSVファイルに一行ずつ親子関係を記述する手間を省き、アクセサリなどに独自の骨格（ボーンチェーン）を即座に実装できる。
- **(0)の意味とツール間仕様:** これはCartoon Animator 5 (CTA5) のボーン強度（Influence）に関わるパラメータである。
  - **CTA5:** (0) と記述することで、そのボーンの影響範囲をゼロに設定し、純粋な「回転軸（ピボット）」として機能させる。これにより関節を曲げた際のメッシュの不自然な歪みを抑止できる。
  - **Moho (RigWeaver):** この表記は無視され、**templates.json** で定義されたグローバルな物理定数が優先される。
  - **推奨:** 両ツールに対応した「汎用PSDリグ」を作成する場合は、関節ボーンには (0) を付与しておくのが安全である。

## 5. プロファイル・システムと標準テンプレート

Weaverシリーズは、「骨格の構造」と「パーツの質感」の組み合わせにより、キャラクターを定義する。

### 5.1 構造 × 質量の 2x2 マトリックス

制作効率を最大化するため、以下の4つの標準テンプレート（.krzファイル）が提供される。

構造 \ 質感	チューブ (Organic)	ロボット (Rigid)
人間型 (Human)	Human_Tube.krz	Human_Robot.krz
四足歩行 (Cat)	Cat_Tube.krz	Cat_Robot.krz

### 5.2 各プロファイルの役割

- **Human V2**: 標準的な二足歩行キャラクター。BVHの階層構造に直結しており、TDPTやポーズデータの再現性が最も高い。
- **Quadruped (Cat)**: 四足歩行動物。前後で独立した重心（Spine1/2）の概念を持ち、複雑な走行アニメーションに対応する。
- **Custom**: 独自のクリーチャー（翼、多足など）を構築するための自由枠。templates.json に新たなキーを作成することで追加可能。

### 5.3 ローカル・オーバーライド（優先検索パス）

LayerWeaverは「開いているPSDのフォルダ」を最優先で検索する。

- PSDと同じフォルダに templates.json 等を置くことで、本体側の設定を汚さずに、そのキャラクター専用の「特別ルール」を適用できる。これにより、作品ごとに異なるボーン構成を安全に切り替えられる。

---

## 6. 物性設計：チューブ（有機）とロボット（剛体）

Weaverの主な利点は、一枚の画像にボーンの動きに合わせた「質感（Material）」を設定できる点にある。

### 6.1 チューブ表現（Organic / Flexi-Bind）

皮膚や尻尾など、一つの描画パーツを複数のボーンでしなやかに曲げるリギング。

- **技術論理**: templates.json の flexi\_binds リストに対象レイヤーを登録する。
- **命名タグとの相関**: 「>」タグを使用してボーンチェーン（親>子）を定義した場合、通常このチューブ表現を適用するのが理想的である。骨の鎖が定義されることで、画像がその鎖に沿って滑らかに湾曲する。
- **挙動**: 指定された複数のボーンの動きを画像が「平均」して受け取る（Rubberhose法）。により、関節部が不自然に折れ曲がらず、ゴムのように滑らかな曲線を描く。
- **推奨**: 腕、脚、尻尾、マント、長い髪。

### 6.2 ロボット表現（Rigid / Region-Bind）

鎧やメカなど、それ自体が変形してはならない硬質なパーツのリギング。

- **技術論理:** `flexi_binds` に名前を記載せず、**単一のボーンのみ**を割り当てる。
- **命名タグとの相関:** 「@」タグを使用して親ボーンに吸着させたパーツは、通常このロボット特性を持つ。ピボット（回転軸）に対して位置を固定する「レイヤーバインド」との親和性が最も高い。
- **挙動:** 割り当てられたボーンに剛体として100%同期し、パーツ内の変形を一切許さない。ボーンの回転のみが画像に伝わり、正確な関節可動を再現する。
- **推奨:** 剣、盾、兜、ロボットの装甲。

### 6.3 首ボーン（Neck）における自動ウェイトとスライド挙動の論理

キャラクターの首を正常に回転（曲げる）させるためには、作画段階における「首の長さ（のりしろ）」とボーンの影響範囲（ウェイト）の関係を正しく設計する必要がある。

- **自動ウェイトとオーバーラップ描き足し:** CTA5等のアニメーションツールにおいて、首ボーン（Neck）の動きに合わせて首の画像を滑らかに曲げるためには、首の画像ピクセルが首ボーンの可動影響範囲（ウェイト範囲）に十分に重なっている必要がある。首のパーツが短すぎると、十分なウェイトが計算されず、ボーンを動かしても画像が曲がらなくなる（頭が体から浮いてしまう原因となる）。これを防ぐため、首パーツは「頭の後ろまでしっかりと描き足す（頭部と深く重なる長さにする）」設計が推奨される。
- **スライド（平行移動）挙動への遷移:** 首の描き足し長さが極端に短く、ボーンの可動影響範囲（ウェイト）が適用されなかった場合、首パーツは曲がらずに頭の動きに追従して平行移動（スライド）するだけの挙動になる。これは「首を曲げずに頭部全体をスライドさせたい」という意図的なデザイン（例：ロボット、機械的な可動キャラ）において有効な動作である。首の曲げ（チューブ表現）か、平行移動（ロボット表現に近い挙動）かは、作画における首の長さによってコントロールできる。

## 7. 特殊自動化ロジックと設計ポリシー

### 7.1 リップシンク自動マッピング（Mouth Logic）

`@Mouth` タグが付与されたグループ内の島（Island）の数に基づき、自動的に音素マッピングを実行する。

- **5音素（日本語標準）:** 日本語の母音「あ・い・う・え・お」の順で5つの口の形を描く。LayerWeaverは検出順（上→下、左→右）に以下の音素を割り当てる。
  1. **あ (Open):** `MouthOpen` / `OpenLip`
  2. **い (EE):** `SmileOpen` / `EE` / `BMP` 等
  3. **う (U):** `PuckerLip` / `U` / `WOO`
  4. **え (Ah\_I):** `Ah_I` / `LNDTh`
  5. **お (Oh):** `Oh`
- **15音素（CTA5フルスペック）:** 高度なリップシンクに対応する場合、以下の15段階の順序で描く必要がある。1:`SmileOpen`, 2:`PuckerLip`, 3:`OpenLip`, 4:`ShowTeeth`, 5:`MouthOpen`, 6:(予備), 7:`Ah_I`, 8:`Oh`, 9:`EE`, 10:`U`, 11:`WOO`, 12:`FV`, 13:`LNDTh`, 14:`CDSKZ`, 15:`BMP`

### 7.2 頭部・表情リギングの「境界線」

Weaverシリーズの設計思想は、\*\*「専門ツールの能力を最大化する下地（構造）を提供すること」\*\*にある。

- **Weaverの守備範囲:** 目、口、鼻、髪などを正しいフォルダ（Face/HeadV2など）へ仕分け、ピボット位置を同期し、リターゲット可能なボーンを配置するまで。

- **Weaverがやらないこと**: 目パチのスマートボーン化、複雑なメッシュ変形、モーフ設定など。
- **理由**: これら精緻な表情制御は、Weaver内で自動化を試みるよりも、CTA5のフェイシャルエディタやMohoのスマートボーン機能を直接使ったほうが、**圧倒的に高精度かつ低工数**で仕上がるからである。

### 7.3 面積最大ノード選択 (get\_art\_layer)

ルート階層における選定ルール。

- **ロジック**: 視認属性に関わらず、面積が最大のレイヤー（あるいはグループ）が自動的に解析対象の「Art（主役）」として選択される。主役を切り替えるには、他の不要な巨大レイヤーを削除するか、グループ外（並列）に置かなければならない。

---

## 8. 解析のステートマシン（診断・解析・最終仕上げ）

解析ボタンの「色」と「段階」は、最短時間で最高品質に到達するための「職人の介入ポイント」を定義している。

### 8.1 第1段階：生成・抽出 (Sculpting) と「診断」

初回クリック（青→黄）。島 (Island) と関節 (Hole) が画素レベルで分離され、診断用レイヤーが生成される。

- **診断とリトライ**: 生成されたレイヤーでパーツの分離が不自然であれば、一度それらを削除してマーカーを再配置・再実行する。これが「理論上の正攻法」である。
- **実務的ショートカット (Islands単独編集)**: 一方で、生成された **islands** レイヤーのみを消しゴムやブラシで加工するだけでも、十分実用的なカットが可能である。
  - **「それでいいのだ」の精神**: **hole** による補填をスキップすることで僅かな境界線（筋）が残る場合があるが、これは実働優先の「暫定解」として許容し、次の工程へ進んで良い。
  - **運用の最適化 (高精度分割)**: パーツの切り分けをより精緻に行いたい場合、描画（または修正）用ブラシとして Krita の「**Digital**」→「**Pixel Art**」カテゴリ (Pixel Art Fill 等) の使用を推奨する。これらのブラシはアンチエイリアスのないバイナリな不透明度 (100% Opaque) を提供するため、システム内部の **alpha\_threshold (不透明度閾値)** **ロジック** の恩恵を最大限に受けることができる。これにより、理論上 **1ピクセル単位の極小の隙間** であっても、システムはそれを確実にセグメンテーション（部位分割）の境界として認識することが可能となる。

### 8.2 第2段階：解析・マッピング (Mapping)

黄色ボタンの2回目クリック。修正済みのレイヤーをスキャンし、ボーンとの論理的な紐付け (i番号の付与) を確定させる。システムの論理知能が構築されるフェーズである。

- **自動判定の限界とTポーズ**: システムの紐付けロジックは、原則として「左右対称のTポーズ」を基準としている。四足歩行モデル（横向き）などの特殊なプロファイルでは、右肢・左肢の判別は技術的に極めて困難であり、推測が外れるのが「普通」である。
- **「テーブルは修正するもの」という原則**: 画面に表示される対応表（テーブル）は、あくまでプログラムによる「推測」のドラフトである。不自然な紐付けや未割当のパーツに関しては、**ユーザーが手動で書き換えて統合を完了させる**必要がある。この「プログラムによる提案 + ユーザーによる最終確定」のフローこそが、多様なモデル構造に対応するためのWeaverの柔軟性を支えている。

### 8.3 最終工程：構築後の最終検品と仕上げ (Final Polish)

ボタン **[2] 2. 最終PSDを構築 & 表示** の実行後、新しいPSDドキュメント上で行う最終仕上げ。

- **実動シミュレーション**: 出力されたPSD上でパーツを **Ctrl+T** で回転させ、のりしろ（重なり）の不備や、第1段階で残した「筋」を目視で特定する。
- **仕上げの加筆・修正**: 回転させて隙間が空く箇所や、目立つ筋をその場で描き足し・消去することで、最終的なビジュアル品質を整える。Weaverは「プログラムの計算結果」を「職人の筆」で調整し、デフォルメ表現を完成させることを前提に設計されている。

## 第8章：templates.json 主要パラメータ対照リファレンス（LayerWeaver 専用）

LayerWeaver は Krita 上で PSD を解析・再構成する際、`templates.json` の以下の項目を参照して自動仕分けを行います。

パラメータ名	工学的役割	設定例
<code>image_folder</code>	<b>ボディ画像フォルダ</b> : キャラクターの体パーツを格納するルートグループ名。	"RL_ImageV2"
<code>image_folder_head</code>	<b>ヘッド画像フォルダ</b> : 表情や頭部パーツ（Face/Ear等）を格納する専用グループ名。	"RL_ImageHeadV2"
<code>face_group_name</code>	<b>表情コアノード</b> : 口、目などの動的パーツを内包する「顔」グループの識別子。	"Face"
<code>direct_face_groups</code>	<b>アングル定義</b> : 顔グループ直下に配置される、方向（Front/Side/Back等）を定義するサブグループ群。	["Front", "Side", "Back"]
<code>ordered_sub_face_groups</code>	<b>パーツ重なり順</b> : 音素や目などのレイヤーを走査・検出する際の「期待される並び順」のリスト。	["BackHair", "Ear", "Face", ..., "FrontHair"]

LayerWeaver：プロフェッショナル技術仕様書 - 全ての論理を、制作の助けに。