

LayerWeaver: Professional Technical Specifications (v16.0)

~ Definition of Internal Algorithms, Naming Conventions, Material Properties, and Core Data Structures ~

Last Updated: 2026/04/20

This document is a comprehensive technical specification for the LayerWeaver (Krita Plugin). Its purpose is to enable engineers to appropriately control and extend the system by referring to this document, and to allow creators to achieve free drawing rigging by understanding its internal logic.

1. System Conceptology: Aesthetics of Independence and Integration

While the Weaver series achieves ultimate automation through a "data baton relay" across the entire series, it is designed with a philosophy of "independence" that allows each tool to be deployed professionally on its own.

1.1 LayerWeaver (Standalone Operation): Efficient Generation of Rigging Assets

Directly generates G3 characters for Cartoon Animator 5 (CTA5) from a "single illustration" drawn by a creator.

- **Immediacy:** Bone placement and rigging are completed just by dragging and dropping the exported PSD into CTA5. You can shift to animation production immediately.
 - *Note: The G3 template standard is an existing industry standard designed by Reallusion. LayerWeaver serves the role of automatically organizing and optimizing data to conform to that standard.*
- **Versatility:** Supports not only human models but also high-difficulty quadruped models.

1.2 MotionWeaver (Standalone Operation): Logical Conversion of Motion

Adapts BVH files from TDPT, Poser, or the internet to any character structure.

- **Shape Conversion:** Freely execute trimming and projection into "High-angle" or "Low-angle" shapes matched to the camera's eye level.
- **Retargeting:** Performs BVH conversion (retargeting) between characters with different structures, maximizing the reusability of assets.

1.3 Integrated Operation: Auto-Rigging with the Weaver Ecosystem

- **Krita (LayerWeaver):** Designs "Islands" and "Tags" to build logical PSDs.
 - **MotionWeaver (Hub):** Converts 3D motion (BVH) onto the coordinate system of the 2.5D canvas (including Z-depth).
 - **RigWeaver (Moho Script):** Reads PSD naming rules and projection data to complete auto-rigging.
-

2. Core File Physical Specifications (The Lawbook)

The versatility of Weaver lies in its "configuration-based" design that completely separates the program (immutable) and data (mutable).

2.1 islands_*.csv (Definition of Physical Identification and Mapping)

Dictionary data used to identify which part (Island ID) an unnamed "Island" extracted on Krita should be treated as.

- **Mapping Logic (Identification Algorithm):**
 - **Two-Point Distance Filter (Strong):** If **two markers** are described in the list, islands existing within a search radius centered on those two points are identified with priority. This works most powerfully by specifying the "endpoints (joints)" of a part.
 - **Inclusion Scoring (Soft):** Determines whether a marker is inside an island (including a 5px buffer) and scores the match rate.
 - **L/R Auto-Determination:** Automatically sorts left and right islands based on the midline connecting the waist (Hip/Pelvis) and head (Head/Neck).
- **Wisdom from the Field:** Setting two primary joint markers placed to "sandwich" a part provides the highest accuracy. If there are sufficient boundary markers for identification, there is no need to describe all internal joints; rather, narrowing them down leads to reduced calculation load and prevention of misjudgment.

2.2 rigging_map_*.csv (Definition of Logical Wiring and Connections)

A blueprint describing which combination of "pixels (Island)" and "joints (Marker)" to use to build the final "layered image with bones."

- **Connectivity (Logic of Connection):** Describing common joint markers between different rows (layers) means defining "**Sharing of markers = Physical connection (Synchronization of pivots)**" on the system.
 - **Example:** By including **Shoulder**, **Elbow** in the **UpperArm** row and **Elbow**, **Wrist** in the **LowerArm** row, both layers are joined with **Elbow** as a shared joint.
- **Data Structure:** **Final Layer Name**, **Reference Island Name**, **Assigned Joint Range**
- **Layer Ordering:** The final stacking order (Z-order) of the layers in the exported PSD is determined by **the order of rows in rigging_map_*.csv (and the confirmed table)**, NOT by the layer order of markers in Krita or the order in **islands.csv**.
 - **Higher rows:** Placed in the **foreground (front)** in the PSD.
 - **Lower rows:** Placed in the **background (back)** in the PSD.
 - **Design Guideline:** By describing parts that should be in front at the top of the list according to the character's perspective, intuitive depth relationships can be constructed. Normally, the default order (from **rigging_map.csv**) works fine, as characters are typically drawn in a T-pose.
- **Description Example of Multi-Bone Chains:** **Tail1**, **Island_Tail**, **j5**, **j6**, **j7**, **j8**, **j9** By listing multiple joints in this way, you can embed a multi-stage joint (bone chain) into a single island.

2.3 templates.json (Master Catalog and Location Definition)

A "central index" that links file paths for the entire Weaver series and physical default values for each profile.

- **Parsing Physical Definitions:**

- **islands_csv / rigging_map_csv**: Defines the location of analysis dictionary files used for each profile.
 - **image_folder / image_folder_head**: Definition of save destination folder names for each exported part.
 - **Analysis Default Values (system_params)**:
 - Defines default thresholds (such as **def_island_thresh**) used for island separation and marker determination. These can be overridden by **config.csv** within the project folder.
-

3. Logical Management and Auto-Construction

The "algorithmic mechanism" of how the Weaver series constructs an intelligent bone character from a set of two-dimensional pixels drawn in Krita.

3.1 Skeleton Topology and Connection Definition

Defines the "completed skeleton structure" assumed by the system, regardless of the presence of physical markers (joints) on the character.

- **target_hierarchy (Skeleton Topology)**:
 - An absolute blueprint defining parent-child relationships of a character. RigWeaver refers to this topology to build the bone hierarchy in Moho.
- **virtual_bones (Virtual Bones)**:
 - Defines logical parents at positions where physical markers do not exist (e.g., thoracic vertebrae or neck connections). Functions as a connector to accurately "translate" 3D motion into the grammar of a 2D rig.
- **flexi_binds (Supple Binding)**:
 - An instruction for "a single image layer to follow the movement of multiple bones." Applied to parts that require organic flexibility, such as tails and capes.

3.2 Absolute Constraints of Drawing Order (ordered_sub_face_groups)

Definition of "depth" when LayerWeaver (Krita) automatically constructs layers.

- The system creates groups in advance in the order listed here (e.g., **Ear** → **Jaw** → **Eye**). This ensures the character is always assembled with the correct front-to-back relationship without the user having to set the order individually.

3.3 Layer Routing and Auto-Tagging

Functions as a "distribution center" that automatically identifies and stores parts based on keywords contained in layer names.

- **face_bone_keywords**: Automatically stores layers containing **eye**, **mouth**, **brow**, etc., into the "Face Parts Group."
- **export_policy**: Controls behavior during export, such as whether to aggregate bones for face parts into a specific target group (such as **Center**).

3.4 Consistency Protection (mismatch_indicators)

"Inspection (validation) logic" executed during analysis.

- If a marker that should not be in the current profile (e.g., a **torso** marker in a quadruped model) is detected, a warning is issued to the user. This prevents serious profile configuration errors beforehand.

3.5 Integration of Semi-Transparent Objects and Joint Detection Specifications

A feature is officially supported to register semi-transparent decorations like scarfs or veils to specific parts by painting the area with markers and detecting them as joints (Hole).

- **Invalidation of Alpha Threshold:** While the **alpha_threshold** defined in **templates.json** is applied to segmenting components for islands (**islands**), **the alpha threshold is fixed to 0** when detecting joints (**joints**). This ensures that even if the artwork under the marker is extremely thin or semi-transparent, it is reliably extracted as a joint area as long as a pixel exists (alpha value > 0).
- **Integration into Base Parts:** By specifying the extracted semi-transparent region (**j** number) in the **Joints** column of a base part such as the body (**Torso**) in the UI correspondence table (or **rigging_map.csv**), the semi-transparent pixel data is merged with the parent part's pixels during the final construction and exported.

4. Configuration Rules via Naming (Tagging Logic)

Symbols included in layer names are "direct behavioral instructions" to the rigging engine.

4.1 "@" Tag: Spatial Snapping and Variation Expansion

A magnet-like function to free illustrators from the trouble of "accurate coordinate alignment."

- **Basic Format:** **Child Part Name @ Anchor Name for Snapping**
 - **Example:** **OpenEye @ Head, Hand_A @ Wrist**
- **Naming Group Layers (Recommended):** While it's possible to attach **@** to individual layers one by one, it's most efficient to gather multiple variations (blinking, mouth shapes, etc.) into a single **Group Layer** and attach **@SnappingDestination** to that group name.
 - **Effect:** Snapping settings are applied collectively to all layers in the group, and they are organized as "Switch Layers" or "Variation Groups" at the output destination.
- **Impact on Rigging (Layer Binding):** Groups (or layers) specified with **@** are **"Layer Bound (fixed collectively)"** to the specified bone by the subsequent rigging engine (RigWeaver, etc.). This creates a robust rig where parts do not fall apart even during intense movement.

4.2 ">" Tag: Dynamic Hierarchy Construction Instruction

Builds complex bone folder hierarchies (parent-child relationships) within the exported PSD just through naming in Krita.

- **Basic Format:** **Parent Bone Name > Child Group Name > Child Bone Name**
 - **Example:** **Hip > Scarf_Root(0) > Scarf_Mid(0) > Scarf_Tip(0)**
- **Effect:** Just by naming this way, parent-child relationships of bones are physically linked during export. This saves the trouble of describing parent-child relationships row by row in a CSV file and allows for immediate implementation of unique skeletons (bone chains) for accessories, etc.
- **Meaning of (0) and Inter-Tool Specifications:** This is a parameter related to bone strength (Influence) in Cartoon Animator 5 (CTA5).

- **CTA5:** By writing `(0)`, the influence range of that bone is set to zero, allowing it to function as a pure "rotation axis (pivot)." This prevents unnatural distortion of the mesh when bending a joint.
- **Moho (RigWeaver):** This notation is ignored, and global physical constants defined in `templates.json` take priority.
- **Recommendation:** When creating a "universal PSD rig" compatible with both tools, it is safe to attach `(0)` to joint bones.

5. Profile System and Standard Templates

The Weaver series defines characters through combinations of "skeleton structure" and "part material."

5.1 2x2 Matrix of Structure × Mass

To maximize production efficiency, four standard templates (.krz files) are provided.

Structure \ Material	Tube (Organic)	Robot (Rigid)
Human Type (Human)	Human_Tube.krz	Human_Robot.krz
Quadruped (Cat)	Cat_Tube.krz	Cat_Robot.krz

5.2 Roles of Each Profile

- **Human V2:** Standard bipedal characters. Directly linked to the hierarchical structure of BVH, providing the highest reproducibility for TDPT and pose data.
- **Quadruped (Cat):** Four-legged animals. Incorporates the concept of independent centers of gravity for front and back (Spine1/2) and supports complex running animations.
- **Custom:** A free slot for building unique creatures (wings, multi-legged, etc.). Can be added by creating a new key in `templates.json`.

5.3 Local Override (Priority Search Path)

LayerWeaver searches the "folder of the open PSD" with the highest priority.

- By placing `templates.json`, etc., in the same folder as the PSD, you can apply "special rules" dedicated to that character without contaminating the settings on the main system side. This allows for safe switching between different bone configurations for each project.

6. Material Design: Tube (Organic) and Robot (Rigid)

A primary advantage of Weaver is the ability to set a "Material" for a single image matched to the bone movement.

6.1 Tube Expression (Organic / Flexi-Bind)

Rigging where a single drawn part is supplied bent with multiple bones, such as skin or a tail.

- **Technical Logic:** Register target layers in the `flexi_binds` list in `templates.json`.

- **Correlation with Naming Tags:** When a bone chain (Parent>Child) is defined using the ">" tag, it is usually ideal to apply this tube expression. Defining a chain of bones allows the image to curve smoothly along that chain.
- **Behavior:** The image receives the movement of specified multiple bones as an "average" (Flexi-Bind method). This prevents joints from bending unnaturally and creates smooth rubbery curves.
- **Recommended for:** Arms, legs, tails, capes, long hair.

6.2 Robot Expression (Rigid / Region-Bind)

Rigging for rigid parts that must not deform themselves, such as armor or mechs.

- **Technical Logic:** Assign only a **single bone** without listing the name in `flexi_binds`.
- **Correlation with Naming Tags:** Parts snapped to a parent bone using the "@" tag usually have this robot characteristic. It has the highest affinity with "Layer Binding," which fixes the position relative to the pivot (rotation axis).
- **Behavior:** Synchronizes 100% as a rigid body with the assigned bone, allowing no deformation within the part. Only the rotation of the bone is transmitted to the image, reproducing accurate joint movement.
- **Recommended for:** Swords, shields, helmets, robot armor.

6.3 Logical Specifications of Automatic Weights and Slide Behavior in the Neck Bone (Neck)

To ensure the character's neck rotates (bends) correctly, it is necessary to design the relationship between the "length of the neck (seam allowance)" and the bone influence range (weights) correctly at the drawing stage.

- **Automatic Weights and Overlap Redrawing:** In animation tools such as CTA5, for the neck image to bend smoothly according to the movement of the neck bone (Neck), the neck image pixels must sufficiently overlap with the movable influence range (weight range) of the neck bone. If the neck part is too short, sufficient weights will not be calculated, and the image will not bend even when the bone is moved (causing the head to detach from the body). To prevent this, it is recommended to draw the neck part long enough so that it "fully extends behind the head (deeply overlapping with the head part)."
- **Transition to Slide (Translation) Behavior:** If the overlap length of the neck is extremely short and the movable influence range (weight) of the bone is not applied, the neck part will not bend and will simply perform a parallel shift (slide) following the movement of the head. This is an effective behavior for intentional designs (e.g., robots or mechanical movable characters) where you want to slide the entire head part without bending the neck. Whether to bend the neck (tube expression) or slide it (behavior close to robot expression) can be controlled by the length of the neck in the drawing.

7. Special Automation Logic and Design Policy

7.1 Lip-Sync Auto-Mapping (Mouth Logic)

Automatically performs phoneme mapping based on the number of Islands detected within a group assigned the @Mouth tag.

- **5 Phonemes (Japanese Standard):** Draw five mouth shapes in the order of Japanese vowels "A, I, U, E, O." LayerWeaver assigns the following phonemes in the order of detection (top to bottom, left to right):

1. **A (Open):** MouthOpen / OpenLip
 2. **I (EE):** SmileOpen / EE / BMP, etc.
 3. **U (U):** PuckerLip / U / WOO
 4. **E (Ah_I):** Ah_I / LNDTh
 5. **O (Oh):** Oh
- **15 Phonemes (CTA5 Full Spec):** When supporting advanced lip-sync, you need to draw in the following 15-stage order: 1:SmileOpen, 2:PuckerLip, 3:OpenLip, 4:ShowTeeth, 5:MouthOpen, 6:(Reserved), 7:Ah_I, 8:Oh, 9:EE, 10:U, 11:WOO, 12:FV, 13:LNDTh, 14:CDSKZ, 15:BMP

7.2 "Boundaries" of Head and Facial Rigging

The design philosophy of the Weaver series is **"to provide the foundation (structure) that maximizes the capabilities of specialized tools."**

- **Weaver's Coverage:** Sorting eyes, mouths, noses, hair, etc., into correct folders (Face/HeadV2, etc.), synchronizing pivot positions, and placing retargetable bones.
- **What Weaver Does Not Do:** Turning blinking into smart bones, complex mesh deformation, morph settings, etc.
- **Reason:** These precise facial controls can be finished with **overwhelmingly high precision and low man-hours** by directly using CTA5's facial editor or Moho's smart bone function, rather than attempting to automate them within Weaver.

7.3 Maximum Area Node Selection (get_art_layer)

Selection rule in the root hierarchy.

- **Logic:** Regardless of visibility attributes, the layer (or group) with the largest area is automatically selected as the "Art (Main Subject)" for analysis. To switch the main subject, you must delete other unnecessary huge layers or place them outside the group (in parallel).

8. State Machine of Analysis (Diagnostics, Analysis, Final Finish)

The "colors" and "stages" of the analysis button define "craftsman intervention points" to reach the highest quality in the shortest time.

8.1 Stage 1: Generation and Extraction (Sculpting) and "Diagnostics"

First click (Blue → Yellow). Islands and Joints (Holes) are separated at the pixel level, and diagnostic layers are generated.

- **Diagnostics and Retry:** If the separation of parts in the generated layers is unnatural, delete them and reposition/re-execute the markers. This is the "theoretical royal road."
- **Practical Shortcut (Direct Editing of Islands):** On the other hand, simply processing only the generated **islands** layer with an eraser or brush can result in a sufficiently practical cut.
 - **Spirit of "It's Good Enough":** Skipping compensation by **hole** may leave slight boundary lines (streaks), but this can be accepted as a "provisional solution" prioritizing practical operation, and you can proceed to the next step.

8.2 Stage 2: Analysis and Mapping (Mapping)

Second click on the yellow button. Scans modified layers and determines logical linking with bones (assignment of i-numbers). This is the phase where the system's logical intelligence is constructed.

- **Limits of Auto-Determination and T-Pose:** The system's linking logic is based on a "symmetrical T-pose" as a rule. In special profiles such as quadruped models (side view), distinguishing between right and left limbs is technically extremely difficult, and it is "normal" for guesses to be wrong.
- **Principle that "Tables are to be Corrected":** The correspondence table displayed on the screen is merely a "guess" draft by the program. For unnatural linking or unassigned parts, the **user must manually rewrite to complete the integration**. This flow of "proposal by program + final confirmation by user" supports Weaver's flexibility to correspond to diverse model structures.

8.3 Final Step: Post-Construction Final Inspection and Finishing (Final Polish)

Final finishing performed on a new PSD document after executing the button **[2] 2. Build & Show Final PSD.**

- **Operational Simulation:** Rotate parts with **Ctrl+T** on the exported PSD to visually identify deficiencies in seam allowances (overlaps) or "streaks" left in the first stage.
- **Cleanup and Correction:** By redrawing or erasing gaps that appear when rotating or conspicuous streaks on the spot, the final visual quality is adjusted. Weaver is designed on the premise that "calculation results of the program" are adjusted with the "craftsman's brush" to complete the deformed expression.

Chapter 9: Major Parameter Cross-Reference for templates.json (LayerWeaver Only)

LayerWeaver performs automatic sorting by referring to the following items in `templates.json` when analyzing and reconstructing PSDs on Krita.

Parameter Name	Engineering Role	Example Setting
<code>image_folder</code>	Body Image Folder: Root group name for storing character body parts.	"RL_ImageV2"
<code>image_folder_head</code>	Head Image Folder: Dedicated group name for storing facial and head parts (Face/Ear, etc.).	"RL_ImageHeadV2"
<code>face_group_name</code>	Facial Core Node: Identifier for the "Face" group containing dynamic parts such as mouth and eyes.	"Face"
<code>direct_face_groups</code>	Angle Definition: Subgroups placed directly under the face group defining directions (Front/Side/Back, etc.).	["Front", "Side", "Back"]
<code>ordered_sub_face_groups</code>	Part Overlap Order: List of "expected order" when scanning and detecting layers such as phonemes and eyes.	["BackHair", "Ear", "Face", ..., "FrontHair"]

LayerWeaver: Professional Technical Specifications - Putting all logic to help production.