**BAIS1C VACE Dance Sync Suite — Node Audit**

**1. source\_video\_loader.py**

**Function**

* Loads a video file, extracts audio (as waveform), BPM (with librosa), frame count, FPS, width, height, duration, and computes a full sync\_meta dictionary.
* Returns a decord VideoReader object, audio dict, metadata, and a UI info string for display.

**Inputs/Outputs**

* **Inputs:** video\_path (string), title (string), debug (bool)
* **Outputs:** video\_obj (decord object), audio\_obj (dict), sync\_meta (dict), ui\_info (string)

**Recommended Fixes**

* **Enforce sync\_meta as the standard meta input/output name** for all nodes.
* **Robust error handling:** Trap exceptions in video/audio reading, and return a meaningful error string via ui\_info.
* **Aspect ratio warning:** Include explicit flags in sync\_meta and UI info when input aspect is nonstandard.
* **Audio channels:** Note in sync\_meta if source audio is stereo (for full transparency).
* **Sample stride (optional):** Add a sample\_stride or max\_frames input for long videos to prevent memory overload.
* **Consistent output:** Ensure that video\_path is always included in sync\_meta for downstream compatibility.

**2. pose\_tensor\_extract.py**

**Function**

* Receives video object and sync\_meta from loader, extracts (dummy, for now) 128-point pose tensors, returns both pose tensor and updated sync\_meta.

**Inputs/Outputs**

* **Inputs:** video\_obj (decord), sync\_meta (dict), title (string), debug (bool)
* **Outputs:** pose\_tensor (torch), sync\_meta (dict)

**Recommended Fixes**

* **Replace dummy pose logic:** Integrate DWPose or other real model.
* **Consistency:** Always use sync\_meta as input/output for metadata.
* **Shape discipline:** Ensure output is always (frames, 128, 2) (strip confidence if needed).
* **Error handling:** If extraction fails, output error info in sync\_meta (e.g., add {"extraction\_success": False, "error": "..."}).

**3. save\_pose\_json.py**

**Function**

* Saves pose tensors and sync\_meta to a JSON file in the dance\_library directory.

**Inputs/Outputs**

* **Inputs:** pose\_tensor (torch), sync\_meta (dict)
* **Outputs:** *(None, only file written; could emit a status string or path)*

**Recommended Fixes**

* **Enforce sync\_meta as input.**
* **Status output:** Return the file path or a success/failure message for UI chaining.
* **Error trapping:** Return or log any file write errors.

**4. music\_control\_net.py**

**Function**

* Takes pose tensors and audio, aligns dance sequence to the music, with optional tempo sync, looping, and video output.

**Inputs/Outputs**

* **Inputs:** audio (dict), pose\_tensor (torch), sync\_meta (dict), plus creative parameters
* **Outputs:** synced\_pose\_tensor (torch), pose\_video (torch), sync\_report (string)

**Recommended Fixes**

* **Enforce sync\_meta as input and output.**
* **Meta propagation:** Downstream meta should always update/reflect new timing/BPM/FPS if adjusted.
* **Shape discipline:** Validate input tensor and adapt if confidence/extra dims.
* **Error output:** Return errors in sync\_report for the UI.

**5. simple\_dance\_poser.py**

**Function**

* Creative dance node — generates basic dance sequences from built-ins or library JSONs, with music reactivity.

**Inputs/Outputs**

* **Inputs:** audio (dict), input\_pose\_tensor (optional), sync\_meta (dict), creative params
* **Outputs:** animated\_poses (torch), dance\_video (torch), creation\_info (string)

**Recommended Fixes**

* **Enforce sync\_meta as input and output.**
* **Shape validation:** Ensure all pose data is consistently shaped (frames, 128, 2).
* **Library input check:** Add error info to creation\_info string if pose load fails or format is wrong.

**6. dance\_poser.py**

**Function**

* Advanced dance poser with music EQ mapping to different limbs, body-part speed modifiers, and rich animation logic.

**Inputs/Outputs**

* **Inputs:** audio (dict), input\_poses (optional), sync\_meta (dict), EQ/speed params
* **Outputs:** pose\_video (torch), pose\_tensor (torch)

**Recommended Fixes**

* **Enforce sync\_meta as input and output.**
* **Pose format discipline:** If pose is 128-point and node expects 91, trim or map as needed, and warn in output if conversion done.
* **Error info:** Return errors in a string output for debugging/UI.

**7. dwpose\_detector.py, wholebody.py, onnxpose.py, onnxdet.py, util.py**

**Function**

* DWPose detection, ONNX inference, and skeleton drawing utilities. Not ComfyUI nodes directly, but used in extraction logic.

**Recommended Fixes**

* **None required** for node interface. Ensure upstream pose extraction calls these when not in dummy mode.

**8. \_\_init\_\_.py**

**Function**

* Handles node registration, dependency validation, and exposes DWPose classes/utilities.

**Recommended Fixes**

* None, unless you want to standardize category or naming further.

**BAIS1C\_SourceVideoLoader.js — Meta Display Widget**

**Function**

* ComfyUI web extension widget that reads the ui\_info string output from the Python node, displays it in a styled HTML panel under the node.
* Highlights the “Aspect:” line in green (WAN/VACE safe) or yellow (non-standard) for quick user inspection.

**Behavior & Handling**

* On node execution, it:
  + Removes any existing panel attached to the node
  + Reads the first string from the ui\_info output (assumed to be the summary string from the loader)
  + Styles the panel (dark background, monospace, pre-wrapped)
  + Applies color to aspect line (#0f0 if “✅”, #ff0 otherwise)
  + Attaches the panel using addDOMWidget so it persists with the node
* **This provides immediate metadata feedback (FPS, duration, aspect, BPM, etc.) inline within the ComfyUI graph, making pipeline QA easier.**

**Recommended Improvements**

* If the loader outputs an error in ui\_info, consider adding a red highlight or alert icon.
* If other nodes downstream emit summary/error strings, extend the JS widget to listen for and display those as well.
* Optionally, make aspect warning clickable for “how to fix” help popups.

**General Recommendations Across Nodes**

1. **Meta Discipline:**
   * Use sync\_meta as the *only* meta input/output name for all nodes.
2. **Consistent Error Reporting:**
   * Any node that can fail should emit errors in its string output (for the widget).
3. **Shape Checks:**
   * Standardize pose tensors to a single shape (128×2 or 91×2, not mixed).
4. **UI Widget Extension:**
   * Let the JS widget handle error and meta string displays from any node, not just the loader.