Suno AI Music Generation JSON Prompt Structure – Technical Reference

Overview and JSON Prompt Format

Suno AI's music generation system (especially in v4.5 and later) allows advanced JSON-formatted prompts to precisely control how a song is generated. In practice, Suno's model ultimately consumes a text prompt with bracketed tags, but structuring your instructions in JSON can help organize song sections, mixing parameters, and voice settings for better results. Each key in the JSON corresponds to either a global parameter (affecting the entire song) or a specific section of the song (like intro, verse, chorus, etc.). By using JSON as a "meta-prompt," creators can clearly specify musical details – from arrangement to vocal style – in a machine-readable way that reduces ambiguity.

> Note: Suno's interface doesn't literally take raw JSON text as input; rather, this format is a structured way to build the prompt. The JSON is ultimately converted to Suno's bracket-tag syntax for the AI to interpret (e.g. "[Verse 1] Lyrics..."). This reference treats the JSON schema as a conceptual blueprint for prompt-building.

Below we break down the core fields and controls in Suno's JSON prompt format, followed by examples and tips for using GPT to generate optimized prompts.

Core Global Fields in a Suno Prompt JSON

At the top level of the JSON, several core fields define the overall generation mode and high-level settings for the track:

action – Specifies the operation mode. For a fresh composition, the action might be "create" (often default if omitted). For re-using or transforming an existing track, use "rebuild" to remix or re-arrange an existing song with new settings. (There are also separate Suno features like Extend, but those are triggered differently – e.g. via an extend API – rather than via this JSON prompt.) For example, {"action": "rebuild", ...} tells Suno to take an existing composition's blueprint and remake it with new parameters.

preserve_structure – Boolean. When rebuilding a track, this controls whether to keep the original song's structure (section order and approximate lengths). If true, the AI will adhere to the existing arrangement of verses, choruses, bridge, etc., in the output. This is useful when you want a new style but the same song layout. If false, the structure can be altered by the AI.

preserve_vocals – Boolean. Applicable in a rebuild/remix scenario; if set to true, Suno will attempt to preserve the original vocals – meaning the same vocal melody and lyrics are retained. This is akin to an instrumental cover or rearrangement: the song's "vocals" (lyrics and tune) remain, but everything else can change. If false, the AI is free to generate new vocal lines or lyrics in the new style. (In practice, even with true the vocals are re-sung by the model, but the intent is to keep the character and lyrics of the original performance.)

enhance_instruments – Boolean. When true, the AI will enrich or complexify the instrumental arrangement during a rebuild. This might add extra ornamentation, improve instrument timbres, or generally make the new instrumentation "bigger" or higher-fidelity. If false, the instruments will be a straight interpretation of the prompt without this extra enhancement step.

instrument_quality – String. Controls the overall fidelity/complexity of instrument sounds. For example, "high" can be used to request high-quality, detailed instrument rendering. Lower settings (e.g., "standard") might yield a simpler or less layered instrumentation, potentially using fewer model resources. In practice, most users opt for high quality unless token limits or speed are concerns.

mixing – String. Sets a general mixing style for how the track's elements are balanced. One known value is "modern", which aims for a clean, contemporary pop mix (crisp highs, punchy but controlled bass, etc.). Other possible values (inferred from context) could be things like "vintage" (for a retro, analog mix feel) or "lofi" for a more muted, unpolished mix – although the exact allowed values aren't officially documented. The mixing parameter influences levels, panning, and overall EQ to match the style era or aesthetic.

mastering – String. Sets the final mastering profile or overall sound coloration of the track. This field has options such as "studio", "warm", and "epic" (as well as possibly "bright"). "studio" would aim for a neutral, polished master as if done in a professional studio (balanced frequency response, standard loudness). "warm" might apply analog warmth – emphasizing bass/mids, gentle compression for a softer, richer tone (akin to tube or vinyl warmth). "epic" likely pushes the master for maximum cinematic impact – e.g. louder volume, enhanced bass and treble, and expanded spatial width for a "larger-than-life" sound. (In fact, using the [epic] modifier in prompts tends to result in "expanded stereo width" and more grandiose dynamics.) Mastering styles help tailor the final output to genres: for instance, a lo-fi acoustic ballad might use a warm master, whereas a cinematic score or EDM anthem might use an epic master for heightened intensity.

Other fields: Suno's JSON may include additional global fields like "seed" (to set a random seed for deterministic output) or "tempo" if tempo control is exposed. However, these are less documented. The primary fields above cover the most impactful high-level controls.

Example: The snippet below illustrates these core fields in context of a remix prompt:

```
{
"action": "rebuild",
"preserve_structure": true,
"preserve_vocals": false,
"enhance_instruments": true,
"instrument_quality": "high",
"mixing": "modern",
"mastering": "studio",
...
}
```

In this example, we're telling Suno to rebuild an existing track, keeping its structure but not retaining the original vocals. We opt to enhance the instruments with high quality, and we want a modern mix and studio-style mastering. After these global fields, the JSON would list the song sections (detailed next).

Defining Song Sections and Structure

Inside the JSON, the song's content is typically organized by sections as key-value pairs. A common practice is to use an outer key like "song" whose value is an object containing all sections (as in some user examples), or simply to list sections at the top level. Each section (intro, verse, chorus, bridge, etc.) is a JSON object that can contain keys for that section's instrumentation, lyrics, and sometimes style or vocals. For example:

```
"verse_1": {
"vocals": "[Male Vocals]",
"instrumentation": "[Acoustic Guitar, Soft Strings]",
"lyrics": "[Verse 1]\nYour lyric lines here..."
}
```

Each section object often includes:

A section tag in brackets as part of lyrics (e.g. "[Verse 1]", "[Chorus]") to explicitly mark it for the AI.

A "lyrics" field: the lyrical content (or a description if the section is instrumental). If the section has no vocals, the lyrics field can still carry an instruction (for example, "[Solo: guitar solo]" or "[Bridge: instrumental breakdown]") to guide the AI's output for that section.

An "instrumentation" field: describing the instruments or accompaniment style in that section. Often given as a bracketed tag like "[instrumental]" for sections with no vocals, or more specific descriptions like "[Gentle Piano]" or "[Harmonic Support]". These tags act like directives to the model about the arrangement. For instance, an instrumentation tag of "[Harmonic Support]" might indicate backing chords or harmonies should be prominent.

A "vocals" field: (optional) specifying the vocal style in that section. This can be used to indicate different singers or group vocals. For example, "[Male Vocals]", "[Female Vocals]", or "[Group Vocals]" are common tags. Using these, you can orchestrate duets or choral sections.

Some sections might also include a "style" field if a particular sub-style is needed locally. For instance, one could tag a bridge with a different feel: "style": "[Soft Ballad]" in the bridge while the rest of the song is pop. In practice, however, many prompts set style primarily via instrumentation and the global tags.

Example: In the advanced JSON example by Jack R., each section was clearly defined with its attributes: the first verse had a Melodic Pop style and male vocals, the chorus used group vocals and harmonic support instrumentation, and the bridge switched to female vocals with a soft ballad piano style. This level of explicit structure makes it more likely the AI will follow the intended arrangement and not skip sections. (In fact, using these section labels can help mitigate issues where Suno would sometimes skip or truncate sections when given ambiguous input.)

To preserve space and clarity (keeping prompts under 1000 characters), it's wise to keep section descriptions concise. Use the bracketed tags (e.g. "[Chorus]") rather than verbose sentences. The JSON structure inherently avoids repetition by isolating each part. If a chorus repeats exactly, you can either list it again as a section (like "chorus_repeated" in the JSON) or simply instruct the AI to repeat it by tagging it appropriately. In Jack's example, they included a second chorus section identical to the first – this is sometimes needed since Suno won't automatically repeat a chorus unless prompted.

Tip: The section keys in JSON can be named flexibly (e.g., "verse1", "verse_1", or even using bracket names as keys). What matters is that the "lyrics" field inside contains the actual bracketed section label for the AI. For instance, you might have "Verse 1": { "lyrics": "[Verse 1] ...", ... } or use a simplified key and include the label in lyrics. Consistency and clarity are the goal.

Fine-Grained Persona Control (Voice Customization)

One of Suno's most powerful features is controlling the vocal persona and delivery. In JSON prompts, this is done via a set of fine-grained parameters usually grouped under the voice or persona settings. These parameters shape how the AI "singer" sounds and performs:

weight – Voice character weight. This controls how strongly the voice leans into a distinct persona or caricature. A lower weight (e.g. 0.5) yields a more neutral or subtle voice, whereas 1.0 makes it full-on character – really leaning into a stylized voice or accent. Essentially, this can dial up the "actor" aspect of the vocals. For example, a cartoonish or heavily accented persona might require weight 1.0, whereas a gentle, plain vocal might use 0.3–0.5 so it's less exaggerated.

tone_shift – Vocal pitch register shift. A **0 value means no shift (natural pitch)**. Negative values make the voice deeper (shifting the pitch down), while positive values make it higher. This is useful for changing the apparent gender or age of the singer or matching a desired key. For instance, tone_shift: -2.0 could turn a voice significantly deeper (maybe an octave down if -1.0 is a semitone – the exact scaling isn't documented, but conceptually negative is deeper). Use small increments to fine-tune; even ±0.5 can noticeably change the vocal tone.

grain – Timbre "grit" or raspiness. Ranges from **0.0 (clean voice) to 1.0 (extremely gravelly/raspy)**. A high grain would emulate a rough, textured voice (think rock or blues singers with vocal fry), while 0 gives a smooth, clear tone. For example, to simulate a smoky jazz club singer, you might set grain around 0.3–0.5 to add some huskiness.

clarity – Diction vs airiness. 1.0 means very clear, crisp enunciation. Lower values introduce a more airy, mumbled or breathy quality (as if the singer is not pronouncing every word sharply). This can make a voice sound intimate or dreamy at lower clarity, versus precise and up-front at high clarity. A value like 0.8 might be a pop vocal with a soft edge, whereas 1.0 is almost too clear (useful for intelligibility, but might sound less emotional if maxed out).

vibrato – Amount of vocal vibrato. 0.0 = none/flat (sustained notes have no oscillation), 1.0 = very dramatic vibrato. Mid-range values produce a natural slight vibrato. For powerful ballads, a vibrato around 0.5–0.7 can sound passionate, whereas fast pop songs might keep this near 0 for a straighter tone. In version 4.5, Suno specifically improved vibrato handling – enabling more natural quivers on held notes. So this parameter can really change the emotional feel of sustained vocals.

intensity – Performance energy and emotional intensity. This influences how forcefully or emotively the vocals are delivered. Higher values (toward 1.0) produce more powerful, emotive singing – more belt and dynamics. Lower values make the performance restrained or soft. For example, a tender lullaby might use intensity 0.3 (gentle, delicate singing), whereas a rock anthem chorus could use 0.9 (full-throttle, belting out notes). This can be adjusted per section – e.g. verses might be lower intensity and choruses higher to follow a typical song dynamic curve.

style_hint – Preset vocal style or persona hint. This is a text label that nudges the AI toward a certain vocal style. For instance, "female_dark_pop" might correspond to a dark-toned female pop vocalist persona. Other examples could be labels like "male_rap_deep" or "female_country_twang" – essentially combining gender and genre descriptors. These hints likely map to internal voice model settings. They are not broadly documented, but the community has discovered some. Using style_hint in combination with the numeric controls above helps achieve a specific singer archetype. (If style_hint is unspecified, the AI will choose a default voice matching the genre/mood, which is sometimes hit-or-miss.)

These persona parameters can usually be placed either at the top level of the JSON (to apply to the main vocalist throughout), or under specific sections if different singers are used. For example, in a duet, you might apply one set of parameters to Verse 1 (for a male voice) and another to Verse 2 (for a female voice). More commonly, one vocalist is consistent, so a single global persona configuration suffices.

Practical example: Suppose we want a female singer with a slightly raspy, emotive voice akin to a dark pop ballad. We could set:

```
{
"style_hint": "female_dark_pop",
"weight": 0.9,
"tone_shift": 0.0,
"grain": 0.2,
"clarity": 0.9,
"vibrato": 0.4,
"intensity": 0.8
}
```

This might yield a rich, characterful female vocal—almost full character weight, mostly clear but with a touch of rasp, some vibrato and strong emotional intensity. According to community findings, weight at 0.9–1.0 would give **“full-on character”**, and grain 0.2 adds slight gravel. Vibrato 0.4 gives noticeable quiver without overpowering. These settings together help the AI emulate, say, a “female dark pop” singer such as a Billie Eilish or Adele style (though one cannot name artists explicitly in prompts, the style hint serves that role).

Vocal Handling: Duets, Backing Vocals, and FX

Beyond single-voice control, Suno prompts can handle multiple voices (duets, call-and-response, harmonies) and certain vocal effects through formatting tricks:

Duet and Multiple Vocalists: The JSON approach for a duet is to explicitly allocate different sections or lines to different vocal tags. For example, you might have:

```
"verse_1": { "vocals": "[Male Vocals]", "lyrics": "[Verse 1]\n...male
singer lines..." },
"verse_2": { "vocals": "[Female Vocals]", "lyrics": "[Verse 2]\n...female
singer lines..." },
"chorus": { "vocals": "[Duet]", "lyrics": "[Chorus]\n...lines sung
together..." }
```

In practice, Suno doesn't have a magic "Duet" button, but setting one verse to male and another to female will make the AI switch voices. In the chorus, you could try a tag like "[Group Vocals]" or even explicitly instruct a duet by overlapping lyrics (though the latter is tricky). Another technique is to use backing vocals: Suno supports writing backup lines in curly braces {} within lyrics. Words in braces are interpreted as harmony or background vocals that accompany the

main vocal. For instance:

[Chorus]

We stand as one, unbroken and free {we echo: "unbroken and free"}

This might cause the phrase "unbroken and free" to be echoed by a second voice in the background. Essentially, braces designate a second singer or choir doing a background part. Using this carefully can create the effect of two voices at once (a true simultaneous duet or harmony). Keep in mind Suno may limit how many distinct voices it can generate clearly at once (often one lead and one backup is reliable; more than two voices can become muddled).

Character Preservation: When doing a remix or multi-section song, you often want the same voice character to persist unless intentionally changed. To preserve vocal character across the song, ensure the persona parameters (style_hint, grain, etc.) remain consistent for all sections that the same "singer" covers. If using the rebuild action on an existing track and wanting the same vocal vibe, you might set preserve_vocals: true (to keep the original melody/lyrics) and also replicate any persona settings the original had. In practice, since you might not know the exact original persona, you simply choose one that fits the desired outcome. The Covers

and Personas feature of v4.5 aims to retain a sense of the original singer when switching genres, but complete timbral preservation is hard – the model will generate a new voice that fits the new style. So, “character preservation” here usually refers to keeping the vocalist consistent within the new track. Do this by not changing style_hint or voice parameters mid-song unless you deliberately want a different singer in a section.

FX Processing on Vocals: There isn’t a direct JSON field like "effect": "autotune" or such, but you can influence vocal effects through prompt text. For example, to get a telephone or lo-fi filtered voice, you might include in the style or instrumentation something like "[Lo-Fi Vocal Filter]" or describe it in lyrics as a stage direction (e.g. "[bridge: vocals sound distant, radio static]"). Suno may or may not always follow these hints, but it sometimes does. There are documented tricks: using asterisks around words can emphasize or distort them (as an “effect”) according to some user tips. Also, writing things like “(vocoder)” or “[Autotune]” in the prompt might coax the model into applying a robotic tone. These are not guaranteed, but creative use of brackets and prompt phrasing can result in effects. For instance, a prompt snippet "[Pre-Chorus: whispered vocals]" would likely yield a hushed delivery for that part.

Important: Because Suno’s output is ultimately an audio generation, you cannot apply arbitrary post-processing via the prompt beyond what the model has been trained to mimic. If you need very specific effects (e.g. heavy autotune, specific reverb settings), you might have to add those in post-production on the exported audio. Suno’s prompt can get you partway (e.g. “echoey vocals” in the style prompt might increase reverb on vocals), but precise control (reverb decay time, etc.) is beyond the prompt’s granularity.

Genre and Instrumentation Transformation

A common use-case is to have Suno change the genre or instrumentation of a song – for example, turning a pop song into a dark orchestral piece or a “glitch pop” remix. The JSON prompt supports this through descriptive tags and instrument lists:

Genre Tags: In the prompt (often in a section’s style or in a top-level "style" field if it existed), you can specify genre cues. For instance, including "[Dark Orchestral]" or "[Glitch Pop]" in a style description will steer the model’s overall arrangement. Even without a dedicated field, these can be placed in the first section (e.g. intro or verse) to establish style. Orchestral prompts typically trigger strings, brass, choir, etc., especially if combined with the [epic] modifier (as discussed, [epic] yields “lush orchestration – strings, horns, choirs” and powerful percussion). Glitch pop, on the other hand, is an electronic style – one might prompt with “glitchy synth arpeggios, bit-crushed effects, quirky pop beat” as part of the instrumentation or an intro description.

Instrumentation Lists: The JSON format allows you to explicitly list instruments per section via the "instrumentation" field. To transform an existing song's instrumentation, simply describe the new instruments. For example, if the original was rock (guitars and drums) and you want dark orchestral, you might set:

"instrumentation": "[Strings, haunting choir, deep brass, orchestral percussion]"

on the relevant sections. The AI will attempt to swap guitar riffs for string motifs, etc. Conversely, for a glitch pop feel, you might use:

"instrumentation": "[Electronic beat, glitch effects, synthesizer arpeggios]"

Now, if you're using action: rebuild, ensure preserve_structure: true so that the song's composition (melodies and section lengths) remain, but the instruments get replaced. The Suno v4.5 "Covers" feature essentially does this – it *"allows users to switch genres on existing tracks – e.g., turning a rock song into a house remix"*. Under the hood, that's a rebuild with new instrumentation/genre tags.

Tempo and Key: While not directly set by JSON fields in any known documentation, you can imply tempo changes by genre choice or by adding tags like "[Slow Ballad]" vs "[Fast 120 BPM]". The model does pay attention to words like "slow, mid-tempo, fast" in the prompt. For key, you might slip in something like "[in A minor]" although it's hit-or-miss if the AI respects exact keys. Usually, describing mood and instruments is more effective than naming a key signature.

Style Consistency: If you want a uniform style across the whole song (e.g. all orchestral), you can just describe it once in an intro meta tag (like [Dark Orchestral Score]) and the AI may carry it through. But to be safe, you can include instrumentation hints in each section's JSON so that the style doesn't "drift". This repetition does count against the character limit, so use short phrases like "[orchestral]" or even just instrument names.

Exclusions: JSON doesn't have a special field for "no guitars" or such, but you can use the negative or exclusion tags in text. For example, some users include an [X] section or a field like "exclude": "[no vocals]" to ensure an instrumental, but a clearer way is to just specify instrumental and not provide lyrics (which you're likely already doing in JSON for an instrumental section). To avoid a particular instrument, you'd need to phrase it positively (emphasize other instruments) because Suno might ignore a "do not include X" note. However, in the text prompt format, sometimes writing something like [NO GUITAR] can work as a hint.

Dynamic & Chorus-Specific Controls (Layering, Intensity, Stereo Width)

Great songs have dynamics – verses might be sparse, and choruses loud and wide. With Suno JSON prompts, you can encourage this via:

Layering and Thickness: To make a chorus bigger, you can layer vocals and instruments in the prompt. As mentioned, using "[Group Vocals]" for a chorus section can yield a choir or multi-singer effect. This effectively layers voices singing in unison or harmony. You can also explicitly state instrumentation layers: e.g. "[Multiple Guitars, Synth Pads]" to suggest a thicker arrangement. If previously a verse had one guitar, the chorus having "instrumentation": "[Layered guitars, bass, drums, backing choir]" will signal a fuller sound. In testing, Suno often responds to “layered” or “stacked” as keywords by making the sound more dense.

Section Intensity: We discussed the intensity parameter under persona control. If you want, you can vary intensity by section. Although there’s no built-in mechanism to say “intensity = 0.5 in verse, 0.9 in chorus” within a single prompt easily, you can achieve it by splitting the vocalist into two persona entries (one for verse, one for chorus) with slightly different intensity values, or more simply, by describing the chorus as powerful in text. For example, put in chorus lyrics or style: "[Chorus: powerful, energetic delivery]". The model will likely sing out more. The global intensity tends to set a baseline, but how the model interprets each section can still be influenced by descriptive words like “softly” vs “with full force” in the section tags.

Stereo Width and Mixing: There isn’t a direct numeric stereo_width field exposed, but certain tags implicitly affect stereo imaging. Notably, the [epic] style tends to produce a wider stereo field (strings and choirs panned out, etc.). You can also prompt explicitly with words like “wide stereo” or “immersive” in the style description. For instance: "[wide stereo synths]" might encourage the model to spread the synths across left-right. In the mixing presets, "modern" mixing likely already has a decent stereo spread. If you wanted a very narrow mono-like mix, you might say "mixing": "vintage mono" (if such an option exists) or add [mono feel] in the style — but this is speculative. In general, epic/cinematic = wide, lo-fi vintage = narrower.

Chorus Impact Tips: Repeating a chorus is one way to emphasize it; Suno sometimes doesn’t know to repeat the final chorus unless told. In JSON you might literally copy the chorus lyrics again in an "outro" or "chorus_2". Also consider using the mastering “epic” setting just for the chorus if that were possible – but since mastering applies to the whole track, you can’t do it per section. Instead, you can trick it by making the chorus instrumentation epic (big drums, etc.) and perhaps adding an [epic] tag in the chorus lyrics line (some have tried nested tags like [Chorus: epic]). According to Jack Whittaker’s guide, adding the word “[epic]” in a section’s prompt text will infuse that section with cinematic scale. Use such modifiers sparingly to avoid overloading the AI with conflicting instructions.

Mixing and Mastering Controls

As covered, the mixing and mastering fields provide high-level control over the sound of the final mix:

Mixing Styles: We know "modern" is one mixing style. This likely results in a clean, digitally polished balance (comparable to commercial 2020s pop production). If other styles exist, they might include terms like "vintage" (which could, for example, lower bass, soften highs, add slight analog noise or distortion, and pan things more centrally as older recordings were) or "live" (maybe simulating a live mix with more reverb and room feel). Without official documentation, you may experiment by setting the field and hearing the differences. If unspecified, Suno presumably defaults to a neutral mix. It's worth noting that earlier versions of Suno had complaints of narrow mixes – by v4.5 they claim "more balanced mixes". Thus, modern might now mean a nicely wide, balanced stereo mix.

Mastering Styles:

Studio: Balanced EQ, standard loudness. Use this for most cases or when unsure – it's likely the default aimed at sounding good on all systems.

Warm: Adds warmth (boost low-mid frequencies, gentle compression). Good for jazz, lo-fi, acoustic, or anything you want to feel cozy or analog. If you prompt something like "warm analog mastering" in text, that might be similar, but the field ensures it. Warm might also slightly reduce harsh treble for a smoother sound.

Epic: Likely maximizes loudness and width, and might hype the bass/treble a bit (a common mastering for trailers or big cinematic music). Use this for orchestral, cinematic, or EDM festival anthems. It can make the track sound larger-than-life, but might introduce subtle distortion or pumping if overdone – as a very high loudness master would.

Other possibilities: Some AI mastering tools use terms like "open", "bright", "bass-heavy". It's conceivable Suno might have a "bright" mastering or "balanced" (besides studio). Check release notes or community findings for any new values.

Using these fields in JSON is straightforward – just set the string. For example:

```
{
"mixing": "modern",
"mastering": "warm"
}
```

would aim for a modern mix with a warm master (perhaps yielding a contemporary sound with an analog vibe). If you wanted the opposite – say an old-school narrow mix but with epic loudness – you could try mixing: "vintage", mastering: "epic" (hypothetically).

Do note that no AI mastering is perfect. Users have observed that Suno's mastering isn't always great by human standards – sometimes too quiet (peaking at -6 dB), other times distorted. So these controls are best-effort. If highest quality is needed, one might export the unmastered audio (if possible) and use a dedicated mastering tool externally. But within Suno, these give you some handle on the output tone.

Keeping JSON Prompts Compact (Under 1000 Characters)

Suno's prompt length limit (around 1000 characters for the text input) means that overly elaborate JSON could be truncated or ignored. Here are strategies to optimize:

Brevity in Descriptions: Use short tags instead of prose. For example, prefer "[Verse 1]" over "[Verse 1:]" (the colon is optional in the tag). Or use a single descriptive word where possible: "[epic]" alone can replace a longer phrase like "grand cinematic style" because the model knows [epic] as a keyword. Similarly, instrument lists can be concise: "[drums, bass, guitar]" instead of "[drum beat, bass guitar, rhythm guitar]" – the shorter will usually suffice.

Avoid Redundancy: If multiple sections share the same style or instrumentation, you don't necessarily need to repeat the full list every time. You might include a global style note in the intro (e.g. "[Glitch Pop song]" at the very start of the lyrics) and then for verses simply put "[Verse]" without restating "glitchy synth" every time. The AI retains context from earlier sections. In JSON, you could have a special key for overall style, or even a "description" field (like in Jack's example, they had a description line outside the song structure). That description doesn't directly become lyrics; it's more for human understanding, but you could embed an overall style there and hope the model uses it. Generally, it's best to put style cues in the actual sections that play, but not to the point of repetition.

Use Abbreviations and Meta Tags: The community often finds that specific meta tags in square brackets are interpreted by Suno even if they aren't fully documented. We've seen tags like [instrumental], [bridge], [intro], etc., all work. Stick to these known tags. If you invent tags, keep them simple (one or two words). Every

character counts, so drop filler words like “the”, “a”, etc., in section descriptors. Compare: "[Bridge: A minimal instrumental section with just piano]" vs "[Bridge: minimal piano only]". The latter conveys the idea in fewer characters.

Lyric Length: If you are providing full custom lyrics, those naturally consume the bulk of the prompt. There’s not much way around that except to be concise in lyric writing. However, if you are not tied to specific lyrics, you could let Suno generate them by feeding just a thematic prompt (but in JSON mode we usually are more explicit). One trick: you can shorten repeated lines by using an ellipsis or a bracketed directive like [x2] to indicate repetition. It’s not guaranteed the AI will follow [x2] (it might literally sing “x2”), but some have used it. For example:

"lyrics": "[Chorus]\nWe will fly high, touch the sky\nNever say goodbye [x2]"

The AI might repeat the line twice. This saves you writing it twice. If that fails, you have to write it out.

No Superfluous JSON Markup: Make sure your JSON syntax itself is tight. Remove any keys that aren’t needed. For instance, if every section has the same vocals throughout, you might decide not to specify "vocals" on every section (just assume the global persona covers it). Or if a section has no special instrumentation (just follow previous), you could omit that field (though leaving it blank might cause default behavior, which could be fine). Be careful: leaving things implicit can sometimes lead to the AI doing unwanted things (like adding a random instrument). So only omit if you’re okay with default behavior.

Check Length Early: A neat trick is to actually count characters of your JSON prompt before submitting. (This is something a GPT-based assistant could help with in the future.) Ensure it’s well under the limit; if near 1000, trim as needed. Sometimes swapping a word with a shorter synonym helps (“melancholy” (9 chars) vs “sad” (3 chars), etc.).

By following these, you can fit a surprisingly detailed prompt in the limit. For example, a full song JSON with two verses, two choruses, and a bridge – each with short descriptions – can come in around 700–900 characters if done efficiently, which is acceptable.

Example JSON Prompt Snippets

To illustrate, here are two examples of effective, compact JSON prompts for different musical goals. These demonstrate many of the concepts above in practice:

Example 1: Acoustic Ballad Conversion

Scenario: We have an upbeat pop song and we want to rebuild it as a mellow acoustic ballad. We'll keep the vocal melody (preserve vocals) but change everything else to be acoustic, warm, and emotional.


```
{
"action": "rebuild",
"preserve_structure": true,
"preserve_vocals": true,
"enhance_instruments": false,
"instrument_quality": "high",
"mixing": "warm",
"mastering": "warm",
"style_hint": "male_soft_ballad",
"weight": 0.8, "tone_shift": -0.5,
"grain": 0.0, "clarity": 1.0,
"vibrato": 0.1, "intensity": 0.5,
"intro": {
"instrumentation": "[instrumental acoustic guitar]",
"lyrics": "[Intro]"
},
"verse_1": {
"vocals": "[Male Vocals]",
"instrumentation": "[Acoustic Guitar]",
"lyrics": "[Verse 1]\nIn these silent hours, I recall your name..."
},
"chorus": {
"vocals": "[Male Vocals]",
"instrumentation": "[Acoustic Guitar, Soft Piano, Strings]",
"lyrics": "[Chorus]\nI'll be with you, through the rise and the fall..."
},
"bridge": {
"instrumentation": "[Solo Piano]",
"lyrics": "[Bridge: instrumental break]"
},
"chorus_final": {
"vocals": "[Male Vocals]",
"instrumentation": "[Acoustic Guitar, Soft Piano, Strings]",
"lyrics": "[Chorus]\nI'll be with you, through the rise and the fall...
(x2)"
}
}
```


Why it works: We set preserve_vocals: true to keep the original lyrics/melody, but we changed the style via instrumentation. The persona is male_soft_ballad with moderate weight and lower tone (tone_shift -0.5 to sing a bit deeper) to give a gentle male voice. We chose a warm mix and warm mastering for an intimate analog feel. Instruments are acoustic guitar as the backbone, with soft piano and strings in the chorus to lift emotion. We explicitly made the bridge an instrumental piano solo to reinforce the ballad vibe. The final chorus repeats

(notated with “(x2)” as a hint to repeat the line – this may or may not be perfectly recognized, but often works). This JSON comes out quite compact but specifies everything important. The AI should produce a slow tempo (implicitly from “acoustic ballad” style) and a heartfelt delivery.

Example 2: Epic Cinematic Remix

Scenario: Take a short simple melody and explode it into an epic orchestral trailer-style track. No need to keep original vocals – we’ll create choir vocals instead – and make it grand.

```
{
"action": "rebuild",
"preserve_structure": true,
"preserve_vocals": false,
"enhance_instruments": true,
"instrument_quality": "high",
"mixing": "modern",
"mastering": "epic",
"style_hint": "female_epic_soprano",
"weight": 1.0, "tone_shift": 0.0,
"grain": 0.1, "clarity": 0.9,
"vibrato": 0.7, "intensity": 0.95,
"intro": {
"instrumentation": "[epic orchestral build-up]",
"lyrics": "[Intro: instrumental]"
},
"verse_1": {
"vocals": "[Female Vocals]",
"instrumentation": "[Strings, Drums, Brass]",
"lyrics": "[Verse]\n(choir)  In the darkness we carry the flame..."
},
"chorus": {
"vocals": "[Female Vocals]",
"instrumentation": "[Thunderous Percussion, Full Orchestra, Choir]",
"lyrics": "[Chorus]\nRise up, we rise up, into the storm we rise!..."
},
"outro": {
"instrumentation": "[big climax outro]",
"lyrics": "[Outro: instrumental]"
}
}
```

Why it works: We chose mastering: "epic" for maximum cinematic impact. We did not preserve vocals – instead, new lyrics are provided with a choir-like style (note I put “(choir)” in the verse lyrics to suggest a choral delivery). The persona is a female epic soprano: weight 1.0 (full character, very dramatic), vibrato 0.7 for that operatic sustain, intensity 0.95 so she belts out nearly at maximum power. Instrumentation lists include strings, brass, big drums, and a full orchestra in chorus, aligning with what we expect from an epic soundtrack (as the [epic] tag normally induces those elements). The mix is modern (should be fine for cinematic too) and

instrument quality high. We also used short descriptive tags (“big climax outro”) to ensure the end is grand. This prompt is approaching the length limit but still within reason, and it explicitly tells the AI to deliver a loud, wide, emotional orchestral remix of the song.

These examples demonstrate balancing detail with brevity. Notice how we leveraged bracketed terms ([Thunderous Percussion], [epic orchestral build-up]) – these compactly convey a lot to the AI.

Using GPT to Generate Optimized Suno Prompts

Given the complexity of constructing these JSON prompts, it’s natural to enlist an AI like GPT to help generate or refine them. Here are recommendations for training or prompting a GPT model to output optimized Suno prompt JSONs for various musical goals:

Provide GPT with the Schema and Examples: First, define the JSON structure for the model. You might give a few examples (like the ones above) of plain English requests and their JSON prompt outputs. For instance, train with a pair: Input: “Convert my rock song into a slow acoustic ballad with female vocals.” Output: (a JSON similar to Example 1, with appropriate adjustments: female vocals, etc.). GPT will learn the pattern of using action: rebuild, toggling preserve_vocals, changing instrumentation to acoustic elements, lowering intensity, choosing warm mastering, etc., when the user asks for an acoustic ballad. By explicitly showing GPT the mapping from request → JSON, you help it understand the logic (this is essentially a few-shot learning approach).

Define Allowed Values and Ranges: In training data or the prompt, enumerate what fields and values are expected. For example, tell GPT: “The output should be a JSON with possible fields: action (create/rebuild), preserve_structure (true/false), preserve_vocals, enhance_instruments, instrument_quality (high/standard), mixing (modern/warm/vintage/etc.), mastering (studio/warm/epic), style_hint (various presets), weight (0–1), tone_shift (–N to N), grain (0–1), clarity (0–1), vibrato (0–1), intensity (0–1), and then sections like intro/verse/chorus with instrumentation, vocals, lyrics.” By giving this schema (like a function signature or JSON schema), GPT is less likely to hallucinate unknown fields. In practice, you can even use OpenAI’s function calling capabilities to enforce this JSON format.

Emphasize Conciseness: When instructing GPT, include guidelines about brevity: e.g., “The JSON should be under 1000 characters. Use short bracket tags and avoid unnecessary words in lyrics or descriptions.” GPT can then try to minimize wording, maybe by using common tags. One can even fine-tune GPT on examples of “long prompt” vs “optimized prompt” to let it see how to compress text.

Musical Context for GPT: Ensure the GPT model has or is given some musical context knowledge – e.g., definitions of genres, instruments, etc. Models like GPT-4 usually know these, but if you have a specialized set of style_hints or internal terms, document those in the training prompt. For example, if “female_dark_pop” is a style_hint you want it to use for a certain request (say the user says “a Billie Eilish style”), you need to have taught GPT that Billie Eilish → style_hint: female_dark_pop, weight ~0.9, grain ~0.2, etc. You could assemble a mapping of famous artists to prompt styles (similar to the one Travis Nicholson provided in his prompt guide, mapping artists to style descriptions). Feeding that to GPT helps it choose appropriate instruments and mood words without naming the artist (since Suno disallows explicit artist names).

Quality Checks: Use GPT’s capabilities to validate JSON output. For instance, after GPT produces a JSON, you could have a second stage where GPT acts as a validator, ensuring the JSON is parseable and within length. This could be done with a simple rule or by using a parser in a script. The build5nines guide suggests prompting the model explicitly: *“Respond with valid JSON only”*. Combining that with a defined schema (via either prompt engineering or OpenAI function parameters) will drastically reduce formatting errors.

Goal-specific Tuning: If you have distinct use-cases (ballad conversion vs cinematic remix vs genre swap), you might fine-tune separate small models or at least include trigger words in the user prompt that GPT picks up on to adjust strategy. For example, if user says “make it cinematic,” GPT should know to set mastering to epic, use orchestral instrumentation, maybe include an [epic] tag, etc. Achieve this by training on the word “cinematic” being present in input and the corresponding JSON including those elements. Similarly, “acoustic” in input → JSON uses acoustic instruments, lower intensity, etc.

Preserve User’s Intent: If a user provides partial info (like “I want an 8-minute ambient track with no vocals, very atmospheric”), GPT should fill the blanks in JSON intelligently: action: create (since it’s a new track), instrumental: true (or preserve_vocals false since no vocals), include sections that reflect ambient structure (maybe longer intro/outro), mixing: modern or perhaps “warm”, mastering: studio or warm, style_hint: maybe none because it’s instrumental, or a specific ambient preset if known, and instrumentation like [synth pads, drones, reverb]. Training GPT with varied examples of inputs – some with explicit details, some requiring inference – will make it robust.

Leverage GPT for Lyric Generation vs. Structure: In some cases, you might want GPT to also generate or tweak lyrics. GPT-4 is quite good at writing lyrics in a specified style. You could feed it the original lyrics and say “adapt these lyrics to a slower tempo and more poetic language for a ballad”, then insert into JSON. Or, if the user hasn’t written lyrics, GPT can create original ones under the "lyrics" fields, matching the theme provided. Just be cautious of length. A strategy: ask GPT for a short chorus lyric if needed, ensure it fits the mood, then populate the JSON.

In summary, training a GPT model to output Suno JSON prompts involves teaching it the format, the vocabulary of styles/instruments, and the cause-effect of prompt changes. With a few examples (as done in community guides) and clear instructions, GPT can reliably produce valid JSON that's tuned for the desired musical goal. This can greatly speed up prompt creation: you describe the transformation you want, and GPT drafts a well-structured prompt that you can then feed into Suno.

By combining the above approaches, creators can exert unprecedented control over AI-generated music. The JSON prompt structure in Suno AI acts like a detailed score or production sheet – you specify what you want, and the generative models (Chirp for music, Bark for vocals) interpret it to create a full composition. As Suno continues to evolve (v4.5's improvements in vocals and prompt fidelity show the trajectory), mastering this prompt format will allow you to push the boundaries of AI music generation. Whether you're turning a punk song into a piano ballad or blending genres in an epic mashup, a well-crafted JSON prompt is key. Use the fine-grained controls – action flags, persona tweaks, mixing/mastering styles – to guide the AI, and consider enlisting GPT as your co-producer to draft these prompts. With practice, you'll be able to reliably coax Suno into producing the exact sound you imagine, all while staying within that 1000-character canvas. Happy prompting!

Examples

Vocal Handling

```
"vocal_handling": {
  "processing": "none",
  "preserve_character": true,
  "duet": {
    "enabled": true,
    "voice_type": "natural_clean",
    "style": "alternating_and_harmony",
    "effects": "none"
  }
}
```

Dynamics

```
"dynamics": {
  "verse": "controlled",
  "chorus": {
    "intensity": "high",
    "volume": "increased",
    "transition": "dramatic",
    "layering": "thicker",
    "stereo_width": "wide"
  }
}
```

Chorus Treatment (Optional)

```
"chorus_treatment": {
  "instrument_boost": true,
  "drums": "heavier_and_punchy",
  "bass": "deeper",
  "synths": "bright_and_layered"
}
```

FX and Tone

```
"effects": {
  "reverb": "natural",
  "compression": "light",
  "distortion": "none"
},
"tone": {
  "brightness": "balanced",
  "warmth": "moderate",
  "space": "wide"
}
```

Example Prompt

```

  "action": "rebuild",
  "preserve_structure": true,
  "preserve_vocals": true,
  "enhance_instruments": true,
  "instrument_quality": "high",
  "mixing": "modern",
  "mastering": "studio",
  "persona": {
    "weight": 0.9,
    "intensity": 1.0,
    "grain": 0.0,
    "clarity": 1.0,
    "tone_shift": 0.0,
    "vibrato": 0.3,
    "style_hint": "haunting_female_ballad"
  }
}
```