

Package: sd2R (via r-universe)

June 2, 2026

Type Package

Title Stable Diffusion Image Generation

Version 0.2.1

Description Provides Stable Diffusion image generation using the 'ggmlR' library, with no 'Python' or external API dependencies. Supports text-to-image and image-to-image generation for SD 1.x, SD 2.x, 'SDXL', and Flux. A single sd_generate() function handles the entire pipeline, including sampling and high-resolution output. Features multi-GPU support, a 'Shiny' GUI, and runs on CPU or 'Vulkan' GPU across Linux, macOS, and Windows.

SystemRequirements GNU make, curl or wget (for downloading vocabulary files during installation)

License MIT + file LICENSE

URL <https://github.com/Zabis13/sd2R>

BugReports <https://github.com/Zabis13/sd2R/issues>

Depends R (>= 4.1.0)

Encoding UTF-8

Imports Rcpp (>= 1.0.0), ggmlR (>= 0.5.0)

LinkingTo Rcpp, ggmlR

Suggests testthat (>= 3.0.0), callr, png, shiny, base64enc, plumber, jsonlite

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs make wget

Repository <https://zabis13.r-universe.dev>

Date/Publication 2026-06-02 20:39:41 UTC

RemoteUrl <https://github.com/zabis13/sd2r>

RemoteRef HEAD

RemoteSha f7ccea2dd08d3772fdc1a1681ed38cf63d8dfa1

Contents

LORA_APPLY_MODE	3
PREDICTION	3
PREVIEW	4
RNG_TYPE	4
SAMPLE_METHOD	4
SCHEDULER	5
sd_api_start	5
sd_api_stop	6
sd_app	7
SD_CACHE_MODE	7
sd_cache_params	8
sd_convert	8
sd_ctx	9
sd_decode_latent	12
sd_default_params	13
sd_denoise_step	14
sd_encode_image	15
sd_encode_text	15
sd_generate	16
sd_generate_multi_gpu	18
sd_generate_multiref	20
sd_image_to_array	21
sd_img2img	22
sd_inverse_noise_scale	24
sd_list_models	24
sd_load_image	25
sd_load_model	25
sd_load_pipeline	26
sd_node	26
sd_noise_scale	27
sd_pipeline	27
sd_preview_start	28
sd_preview_stop	29
sd_profile_get	29
sd_profile_start	29
sd_profile_stop	30
sd_profile_summary	30
sd_read_preview	30
sd_register_model	31
sd_remove_model	32
sd_run_pipeline	32
sd_sample	33
sd_sample_stepwise	34
sd_sampler_begin	35
sd_sampler_sigmas	36
sd_save_image	37

sd_save_pipeline	37
sd_scan_models	38
sd_supports_ref_images	38
sd_system_info	39
sd_txt2img	39
sd_txt2img_highres	41
sd_txt2img_tiled	43
SD_TYPE	45
sd_unload_all	45
sd_unload_model	46
sd_upscale_image	46
sd_vulkan_device_count	47

Index	48
--------------	-----------

LORA_APPLY_MODE	<i>LoRA apply modes</i>
-----------------	-------------------------

Description

LoRA apply modes

Usage

LORA_APPLY_MODE

Format

An object of class `list` of length 3.

PREDICTION	<i>Prediction types</i>
------------	-------------------------

Description

Prediction types

Usage

PREDICTION

Format

An object of class `list` of length 6.

PREVIEW	<i>Preview decode modes</i>
---------	-----------------------------

Description

Mode strings accepted by the preview callback (see [sd_preview_start](#)). "proj" is a fast linear projection of the latent (cheap, rough), "tae" uses the tiny autoencoder (needs `taesd_path`), "vae" runs the full VAE (slow, accurate).

Usage

PREVIEW

Format

An object of class `list` of length 4.

RNG_TYPE	<i>RNG types</i>
----------	------------------

Description

RNG types

Usage

RNG_TYPE

Format

An object of class `list` of length 3.

SAMPLE_METHOD	<i>Sampling methods</i>
---------------	-------------------------

Description

Sampling methods

Usage

SAMPLE_METHOD

Format

An object of class `list` of length 12.

SCHEDULER	<i>Schedulers</i>
-----------	-------------------

Description

Schedulers

Usage

SCHEDULER

Format

An object of class `list` of length 10.

<code>sd_api_start</code>	<i>Start sd2R REST API server</i>
---------------------------	-----------------------------------

Description

Launches a plumber-based REST API for image generation. Optionally pre-loads a model at startup.

Usage

```
sd_api_start(
  model_path = NULL,
  model_type = "sd1",
  model_id = NULL,
  vae_decode_only = TRUE,
  host = "0.0.0.0",
  port = 8080L,
  api_key = NULL,
  ...
)
```

Arguments

<code>model_path</code>	Optional path to model file to load at startup
<code>model_type</code>	Model type for the pre-loaded model (default "sd1")
<code>model_id</code>	Identifier for the pre-loaded model (default: basename of <code>model_path</code>)
<code>vae_decode_only</code>	VAE decode only for the pre-loaded model (default TRUE)
<code>host</code>	Host to bind to (default "0.0.0.0")

port	Port to listen on (default 8080)
api_key	Optional API key string. When set, non-localhost requests must include X-API-Key or Authorization: Bearer <key> header. Default NULL (no auth).
...	Additional arguments passed to <code>sd_ctx</code> for the pre-loaded model

Value

Invisibly returns the plumber router object

Examples

```
## Not run:  
# Start with a pre-loaded model  
sd_api_start("model.safetensors", model_type = "flux", port = 8080)  
  
# Start empty, load models via API  
sd_api_start(port = 8080)  
  
# With API key  
sd_api_start("model.safetensors", api_key = "my-secret-key")  
  
## End(Not run)
```

sd_api_stop	<i>Stop sd2R REST API server</i>
-------------	----------------------------------

Description

Stops the running plumber server and unloads all models.

Usage

```
sd_api_stop()
```

Value

No return value, called for side effects.

sd_app	<i>Launch sd2R Shiny GUI</i>
--------	------------------------------

Description

Opens an interactive Shiny application for text-to-image generation. Requires the **shiny** and **base64enc** packages.

Usage

```
sd_app(model_dir = NULL, launch.browser = TRUE, port = NULL, ...)
```

Arguments

model_dir	Path to folder with model files. If provided, the app scans the folder on startup and auto-assigns model roles.
launch.browser	Open in browser (default TRUE)
port	Port number (default NULL = random)
...	Additional arguments passed to runApp

Value

This function does not return; it runs the Shiny app until stopped.

Examples

```
## Not run:
sd_app()
sd_app(model_dir = "/path/to/models")

## End(Not run)
```

SD_CACHE_MODE	<i>Cache modes</i>
---------------	--------------------

Description

Cache modes

Usage

```
SD_CACHE_MODE
```

Format

An object of class `list` of length 6.

sd_cache_params *Create cache configuration for step caching*

Description

Constructs a list of cache parameters for fine-tuning step caching behavior. Pass the result as cache_config to generation functions.

Usage

```
sd_cache_params(
  mode = SD_CACHE_MODE$EASYCACHE,
  threshold = 1,
  start_percent = 0.15,
  end_percent = 0.95
)
```

Arguments

mode	Cache mode integer from SD_CACHE_MODE (default EASYCACHE)
threshold	Reuse threshold (default 1.0). Lower = more aggressive caching
start_percent	Start caching after this fraction of steps (default 0.15)
end_percent	Stop caching after this fraction of steps (default 0.95)

Value

Named list of cache parameters

sd_convert *Convert model to different quantization format*

Description

Convert model to different quantization format

Usage

```
sd_convert(
  input_path,
  output_path,
  output_type = SD_TYPE$F16,
  vae_path = NULL,
  tensor_type_rules = NULL
)
```

Arguments

input_path	Path to input model file
output_path	Path for output model file
output_type	Target quantization type (see SD_TYPE)
vae_path	Optional path to separate VAE model
tensor_type_rules	Optional tensor type rules string

Value

TRUE on success

sd_ctx	<i>Create a Stable Diffusion context</i>
--------	--

Description

Loads a model and creates a context for image generation.

Usage

```
sd_ctx(
  model_path = NULL,
  vae_path = NULL,
  taesd_path = NULL,
  clip_l_path = NULL,
  clip_g_path = NULL,
  t5xxl_path = NULL,
  llm_path = NULL,
  diffusion_model_path = NULL,
  control_net_path = NULL,
  n_threads = 0L,
  wtype = SD_TYPE$COUNT,
  tensor_type_rules = NULL,
  vae_decode_only = TRUE,
  free_params_immediately = FALSE,
  keep_clip_on_cpu = FALSE,
  keep_vae_on_cpu = FALSE,
  enable_mmap = FALSE,
  vae_conv_direct = TRUE,
  diffusion_conv_direct = FALSE,
  diffusion_flash_attn = TRUE,
  rng_type = RNG_TYPE$CUDA,
  prediction = NULL,
  lora_apply_mode = LORA_APPLY_MODE$AUTO,
```

```

    flow_shift = 0,
    model_type = "sd1",
    vram_gb = NULL,
    device_layout = "mono",
    diffusion_gpu = -1L,
    clip_gpu = -1L,
    vae_gpu = -1L,
    verbose = FALSE
)

```

Arguments

model_path	Path to the model file (safetensors, gguf, or checkpoint)
vae_path	Optional path to a separate VAE model
taesd_path	Optional path to TAESD model for preview
clip_l_path	Optional path to CLIP-L model
clip_g_path	Optional path to CLIP-G model
t5xxl_path	Optional path to T5-XXL model
llm_path	Optional path to an LLM text encoder (Qwen3 / Mistral-Small). Required for models that use an LLM conditioner, e.g. FLUX.2 Klein (Qwen3), FLUX.2 (Mistral-Small), Z-Image and Qwen-Image. Loaded into the <code>text_encoders.llm</code> slot.
diffusion_model_path	Optional path to separate diffusion model
control_net_path	Optional path to ControlNet model
n_threads	Number of CPU threads (0 = auto-detect)
wtype	Weight type for quantization (see <code>SD_TYPE</code>)
tensor_type_rules	Optional per-component weight type override, as a comma-separated string of <code>pattern=type</code> rules. Each pattern is a regex matched against tensor names; the first match wins. Use this to load specific model components at a different precision than <code>wtype</code> . Examples: <ul style="list-style-type: none"> "first_stage_model=f16" — load VAE at F16 "first_stage_model=f16,model.diffusion_model=q8_0" — VAE F16, UNet Q8_0 Type names match ggml type names ("f16", "f32", "q8_0", etc.).
vae_decode_only	If TRUE, only load VAE decoder (saves memory)
free_params_immediately	Free model params after first computation. If TRUE, the context can only be used for a single generation — subsequent calls will crash. Set to TRUE only when you need to save memory and will not reuse the context. Default is FALSE.

keep_clip_on_cpu	Keep CLIP model on CPU even when using GPU
keep_vae_on_cpu	Keep VAE on CPU even when using GPU
enable_mmap	Memory-map model weights from disk instead of reading them into a malloc'd buffer (default FALSE). Lowers RAM footprint for large models (e.g. Flux); pages are loaded on demand by the OS and shared across processes. Ignored for zip-archived weights. May slow the first generation slightly as pages fault in.
vae_conv_direct	Use direct Conv2d implementation in VAE (default TRUE). Faster on GPU; skips im2col and uses direct convolution kernels.
diffusion_conv_direct	Use direct Conv2d in diffusion model (default FALSE).
diffusion_flash_attn	Enable flash attention for diffusion model (default TRUE). Set to FALSE if you experience issues with specific GPU drivers or backends.
rng_type	RNG type (see RNG_TYPE)
prediction	Prediction type override (see PREDICTION), NULL = auto
lora_apply_mode	LoRA application mode (see LORA_APPLY_MODE)
flow_shift	Flow shift value for Flux models
model_type	Model architecture hint: "sd1", "sd2", "sdx1", "flux", "flux2", "sd3", or "auto". Used by sd_generate to determine native resolution and tile sizes. With "auto", the type is detected from a sibling config.json then the filename (GGUF-metadata detection is a future hook); detection errors with a hint if it cannot decide. Default "sd1".
vram_gb	Override available VRAM in GB. When set, disables auto-detection and uses this value for strategy routing. Default NULL (auto-detect from Vulkan device).
device_layout	GPU layout preset for multi-GPU systems. One of: "mono" All models on one GPU (default). "split_encoders" Text encoders (CLIP/T5) on GPU 1, diffusion + VAE on GPU 0. "split_vae" Text encoders + VAE on GPU 1, diffusion on GPU 0. Maximizes VRAM for diffusion. "encoders_cpu" Text encoders on CPU, diffusion + VAE on GPU. Saves GPU memory at the cost of slower text encoding. Ignored when diffusion_gpu, clip_gpu, or vae_gpu are explicitly set (>= 0).
diffusion_gpu	Vulkan GPU device index for the diffusion model. Default -1 (use SD_VK_DEVICE env or device 0). Overrides device_layout.
clip_gpu	Vulkan GPU device index for CLIP/T5 text encoders. Default -1 (same device as diffusion). Overrides device_layout.
vae_gpu	Vulkan GPU device index for VAE encoder/decoder. Default -1 (same device as diffusion). Overrides device_layout.
verbose	If TRUE, print model loading progress and sampling steps. Default FALSE.

Value

An external pointer to the SD context (class "sd_ctx") with attributes `model_type`, `vae_decode_only`, `vram_gb`, `vram_total_gb`, and `vram_device`.

Examples

```
## Not run:
ctx <- sd_ctx("model.safetensors")
imgs <- sd_txt2img(ctx, "a cat sitting on a chair")
sd_save_image(imgs[[1]], "cat.png")

## End(Not run)
```

sd_decode_latent	<i>Decode a latent into a pixel image (low-level VAE decode)</i>
------------------	--

Description

Decode a latent into a pixel image (low-level VAE decode)

Usage

```
sd_decode_latent(ctx, latent)
```

Arguments

ctx	SD context
latent	An <code>sd_tensor</code> list (e.g. the output of sd_sample or sd_encode_image).

Value

An `sd_image` list (width, height, channel, data).

See Also

[sd_save_image](#)

sd_default_params	<i>Default generation parameters</i>
-------------------	--------------------------------------

Description

Returns a named list of all per-generation defaults used by [sd_generate](#). Edit the returned list and pass it back via the `params` argument to set a reusable baseline; any explicit argument to `sd_generate()` overrides the matching field.

Usage

```
sd_default_params()
```

Details

This is the R-level analogue of `IRIS_PARAMS_DEFAULT`. It covers generation knobs only; context-construction options (model paths, devices, offload, etc.) belong to [sd_ctx](#).

Value

A named list with fields: `negative_prompt`, `width`, `height`, `strength`, `sample_method`, `sample_steps`, `cfg_scale`, `seed`, `batch_count`, `scheduler`, `clip_skip`, `eta`, `hr_strength`, `vae_mode`, `vae_tile_size`, `vae_tile_overlap`, `cache_mode`, `cache_config`.

See Also

[sd_generate](#)

Examples

```
p <- sd_default_params()
p$sample_steps <- 30
p$cfg_scale <- 4.0
## Not run:
ctx <- sd_ctx("model.safetensors", model_type = "auto")
imgs <- sd_generate(ctx, "a cat", params = p)

## End(Not run)
```

sd_denoise_step *Run a single denoise step (low-level)*

Description

Runs the diffusion model once on `x` at `sigma` and returns the denoised `x_0` estimate. The Euler update of `x` is done by the caller (see [sd_sample_stepwise](#) for the full loop). Must be called between [sd_sampler_begin](#) and [sd_sampler_end](#).

Usage

```
sd_denoise_step(  
    ctx,  
    x,  
    sigma,  
    cond,  
    uncond = list(crossattn = NULL, vector = NULL, concat = NULL),  
    cfg_scale = 7,  
    step = 1L,  
    total_steps = 1L  
)
```

Arguments

<code>ctx</code>	SD context
<code>x</code>	Current latent <code>sd_tensor</code>
<code>sigma</code>	Current sigma (scalar)
<code>cond</code>	Positive conditioning from sd_encode_text
<code>uncond</code>	Negative conditioning; empty (all NULL) disables CFG
<code>cfg_scale</code>	CFG scale (1 disables CFG)
<code>step, total_steps</code>	1-based step index / total, for progress hooks

Value

An `sd_tensor` list — the denoised `x_0` estimate.

See Also

[sd_sample_stepwise](#)

sd_encode_image	<i>Encode an image into a latent (low-level VAE encode)</i>
-----------------	---

Description

Encode an image into a latent (low-level VAE encode)

Usage

```
sd_encode_image(ctx, image)
```

Arguments

ctx	SD context (must be built with <code>vae_decode_only = FALSE</code>)
image	An <code>sd_image</code> list (width, height, channel, data) as produced by sd_load_image .

Value

An `sd_tensor` list (type, ne, data) — the latent.

See Also

[sd_decode_latent](#), [sd_sample](#)

sd_encode_text	<i>Encode a text prompt into conditioning (low-level)</i>
----------------	---

Description

Runs only the text-encoder stage of the pipeline, returning the conditioning tensors (analogue of `SDCondition`). Building block for custom pipelines; most users want [sd_generate](#).

Usage

```
sd_encode_text(ctx, prompt, clip_skip = -1L, width = -1L, height = -1L)
```

Arguments

ctx	SD context from sd_ctx
prompt	Text prompt
clip_skip	CLIP layers to skip (-1 = model default)
width, height	Intended generation size (affects size-conditioning for some models, e.g. SDXL). -1 lets the model decide.

Value

A conditioning list with elements crossattn, vector, concat; each is an sd_tensor list (type, ne, data) or NULL when the model does not produce it.

See Also

[sd_sample](#), [sd_decode_latent](#)

sd_generate	<i>Generate images (unified entry point)</i>
-------------	--

Description

Automatically selects the best generation strategy based on output resolution and available VRAM (set via vram_gb in [sd_ctx](#)). For txt2img, routes between direct generation, tiled sampling (Multi-Diffusion), or hires fix. For img2img (when init_image is provided), routes between direct and tiled img2img.

Usage

```
sd_generate(
    ctx,
    prompt,
    negative_prompt = "",
    width = 512L,
    height = 512L,
    init_image = NULL,
    strength = 0.75,
    sample_method = SAMPLE_METHOD$EULER,
    sample_steps = 20L,
    cfg_scale = 7,
    seed = 42L,
    batch_count = 1L,
    scheduler = SCHEDULER$DISCRETE,
    clip_skip = -1L,
    eta = 0,
    hr_strength = 0.4,
    vae_mode = "auto",
    vae_tile_size = 64L,
    vae_tile_overlap = 0.25,
    cache_mode = c("off", "easy", "ucache"),
    cache_config = NULL,
    params = NULL,
    preview = FALSE,
    preview_path = NULL,
    preview_mode = PREVIEW$PROJ,
    preview_interval = 1L
)
```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt describing desired image
negative_prompt	Negative prompt (default "")
width	Image width in pixels (default 512)
height	Image height in pixels (default 512)
init_image	Optional init image for img2img. If provided, runs img2img instead of txt2img. Requires <code>vae_decode_only = FALSE</code> .
strength	Denosing strength for img2img (default 0.75). Ignored for txt2img.
sample_method	Sampling method (see <code>SAMPLE_METHOD</code>)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see <code>SCHEDULER</code>)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
hr_strength	Denosing strength for highres fix refinement pass (default 0.4). Only used when auto-routing selects highres fix.
vae_mode	VAE processing mode: "normal", "tiled", or "auto" (VRAM-aware: queries free GPU memory and enables tiling only when estimated peak VAE usage exceeds available VRAM minus a 50 MB reserve). Default "auto".
vae_tile_size	Tile size for VAE tiling (default 64)
vae_tile_overlap	Overlap for VAE tiling (default 0.25)
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache), or "ucache" (UCache).
cache_config	Optional fine-tuned cache config from sd_cache_params .
params	Optional baseline list from sd_default_params . Supplies defaults for any generation argument not passed explicitly; explicitly named arguments to <code>sd_generate()</code> always take precedence. NULL (default) keeps the built-in defaults.
preview	If TRUE, write intermediate preview frames during generation to <code>preview_path</code> ; poll with sd_read_preview . Default FALSE (zero cost). See sd_preview_start .
preview_path	File path for the preview PPM. Defaults to a tempfile when <code>preview = TRUE</code> .
preview_mode	Preview decode mode (see <code>PREVIEW</code>); default "proj".
preview_interval	Emit a preview every N steps (default 1).

Details

When `vram_gb` is not set on the context, defaults to direct generation (equivalent to calling [sd_txt2img](#) or [sd_img2img](#) directly).

Value

List of SD images (or single image for highres fix path).

Examples

```
## Not run:
# Simple - auto-routes based on detected VRAM
ctx <- sd_ctx("model.safetensors", model_type = "sd1",
             vae_decode_only = FALSE)
imgs <- sd_generate(ctx, "a cat", width = 2048, height = 2048)

# Manual override - force 4 GB VRAM limit
ctx4 <- sd_ctx("model.safetensors", model_type = "sd1",
              vram_gb = 4, vae_decode_only = FALSE)
imgs <- sd_generate(ctx4, "a cat", width = 2048, height = 2048)

## End(Not run)
```

sd_generate_multi_gpu *Parallel generation across multiple GPUs*

Description

Distributes prompts across available Vulkan GPUs, running one process per GPU via callr. Each process creates its own `sd_ctx` and calls `sd_generate`. Requires the callr package.

Usage

```
sd_generate_multi_gpu(
  model_path = NULL,
  prompts,
  negative_prompt = "",
  devices = NULL,
  seeds = NULL,
  width = 512L,
  height = 512L,
  model_type = "sd1",
  vram_gb = NULL,
  vae_decode_only = TRUE,
  progress = TRUE,
  diffusion_model_path = NULL,
  vae_path = NULL,
  clip_l_path = NULL,
  t5xxl_path = NULL,
  ...
)
```

Arguments

model_path	Path to the model file (single-file models like SD 1.x/2.x/SDXL)
prompts	Character vector of prompts (one image per prompt)
negative_prompt	Negative prompt applied to all images (default "")
devices	Integer vector of Vulkan device indices (0-based). Default NULL auto-detects all available devices.
seeds	Integer vector of seeds, same length as prompts. Default NULL generates random seeds.
width	Image width (default 512)
height	Image height (default 512)
model_type	Model type (default "sd1")
vram_gb	VRAM per GPU for auto-routing (default NULL)
vae_decode_only	VAE decode only (default TRUE)
progress	Print progress messages (default TRUE)
diffusion_model_path	Path to diffusion model (Flux/multi-file models)
vae_path	Path to VAE model
clip_l_path	Path to CLIP-L model
t5xxl_path	Path to T5-XXL model
...	Additional arguments passed to sd_generate

Value

List of SD images, one per prompt, in original order.

Note

Release any existing SD context (`rm(ctx); gc()`) before calling this function. Holding a Vulkan context in the main process while subprocesses try to use the same GPU can produce corrupted (grey) images.

Examples

```
## Not run:
# Single-file model (SD 1.x/2.x/SDXL)
imgs <- sd_generate_multi_gpu(
  "model.safetensors",
  prompts = c("a cat", "a dog", "a bird", "a fish"),
  devices = 0:1
)

# Multi-file model (Flux)
imgs <- sd_generate_multi_gpu(
```

```

diffusion_model_path = "flux1-dev-Q4_K_S.gguf",
vae_path = "ae.safetensors",
clip_l_path = "clip_l.safetensors",
t5xxl_path = "t5-v1_1-xxl-encoder-Q5_K_M.gguf",
prompts = c("a cat", "a dog"),
model_type = "flux", devices = 0:1
)

## End(Not run)

```

sd_generate_multiref *Generate an image conditioned on multiple reference images*

Description

Runs generation with one or more reference images, as used by edit / reference-conditioned models (e.g. Qwen-Image, FLUX control/edit variants). The references are passed straight through to the underlying generate_image C-API (ref_images); the active model decides how to use them, so this only has effect on models that support reference conditioning.

Usage

```

sd_generate_multiref(
  ctx,
  prompt,
  refs,
  negative_prompt = "",
  width = 512L,
  height = 512L,
  auto_resize_ref_image = TRUE,
  increase_ref_index = FALSE,
  sample_method = SAMPLE_METHOD$EULER,
  scheduler = SCHEDULER$DISCRETE,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  clip_skip = -1L,
  eta = 0,
  batch_count = 1L
)

```

Arguments

ctx	SD context from sd_ctx
prompt	Text prompt
refs	A list of sd_image lists (each with width, height, channel, data), e.g. from sd_load_image .

negative_prompt Negative prompt (default "")
width, height Output size in pixels
auto_resize_ref_image If TRUE (default), references are resized to fit the model's expected reference size.
increase_ref_index If TRUE, reference latents get increasing positional indices (model-specific; default FALSE).
sample_method, scheduler Sampler / scheduler (name or enum value)
sample_steps, cfg_scale, seed, clip_skip, eta Standard sampling controls
batch_count Number of images (default 1)

Value

List of sd_image lists.

See Also

[sd_generate](#), [sd_encode_image](#)

sd_image_to_array *Convert SD image to R numeric array*

Description

Converts the raw uint8 SD image format to a [height, width, channels] numeric array with values in [0, 1] suitable for R image processing.

Usage

```
sd_image_to_array(image)
```

Arguments

image SD image list (width, height, channel, data)

Value

3D numeric array [height, width, channels] in [0, 1]

`sd_img2img`*Generate images with img2img*

Description

Generate images with `img2img`

Usage

```
sd_img2img(  
    ctx,  
    prompt,  
    init_image,  
    negative_prompt = "",  
    width = NULL,  
    height = NULL,  
    sample_method = SAMPLE_METHOD$EULER,  
    sample_steps = 20L,  
    cfg_scale = 7,  
    seed = 42L,  
    batch_count = 1L,  
    scheduler = SCHEDULER$DISCRETE,  
    clip_skip = -1L,  
    strength = 0.75,  
    eta = 0,  
    vae_mode = "auto",  
    vae_auto_threshold = 1048576L,  
    vae_tile_size = 64L,  
    vae_tile_overlap = 0.25,  
    vae_tile_rel_x = NULL,  
    vae_tile_rel_y = NULL,  
    vae_tiling = NULL,  
    cache_mode = c("off", "easy", "ucache"),  
    cache_config = NULL  
)
```

Arguments

<code>ctx</code>	SD context created by <code>sd_ctx</code>
<code>prompt</code>	Text prompt describing desired image
<code>init_image</code>	Init image in <code>sd_image</code> format. Use <code>sd_load_image</code> to load from file.
<code>negative_prompt</code>	Negative prompt (default "")
<code>width</code>	Image width in pixels (default 512)
<code>height</code>	Image height in pixels (default 512)

sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
strength	Denoising strength (0.0 = no change, 1.0 = full denoise, default 0.75)
eta	Eta parameter for DDIM-like samplers
vae_mode	VAE processing mode: "normal" (no tiling), "tiled" (always tile), or "auto" (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default "auto".
vae_auto_threshold	Pixel area fallback threshold for vae_mode = "auto" when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when width * height exceeds this value. Default 1048576L (1024x1024 pixels).
vae_tile_size	Tile size in latent pixels for tiled VAE (default 64). Ignored when vae_tile_rel_x/vae_tile_rel_y are set.
vae_tile_overlap	Overlap ratio between tiles, 0.0-0.5 (default 0.25)
vae_tile_rel_x	Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tile_rel_y	Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tiling	Deprecated. Use vae_mode instead. If TRUE, equivalent to vae_mode = "tiled".
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache — skips redundant denoising steps), or "ucache" (UCache). Can speed up sampling 20-40% with minor quality impact.
cache_config	Optional fine-tuned cache config from sd_cache_params . Overrides cache_mode when provided.

Value

List of SD images

`sd_inverse_noise_scale`*Undo final-step latent scaling (low-level)*

Description

Applies the denoiser's inverse noise scaling after the last step. A no-op for discrete CompVis denoisers (SD1/SD2/SDXL).

Usage

```
sd_inverse_noise_scale(ctx, x, sigma_last)
```

Arguments

<code>ctx</code>	SD context
<code>x</code>	Latent <code>sd_tensor</code> after the last step
<code>sigma_last</code>	Last sigma of the schedule (typically 0)

Value

An `sd_tensor`.

`sd_list_models`*List registered models*

Description

Returns a data frame of all models recorded in the sd2R model registry, with a column indicating which are currently loaded in memory.

Usage

```
sd_list_models()
```

Value

Data frame with columns: `id`, `model_type`, `loaded`, `diffusion_path`

sd_load_image	<i>Load image from file as SD image</i>
---------------	---

Description

Reads a PNG file and converts it to the SD image format (list with width, height, channel, data) suitable for `img2img`.

Usage

```
sd_load_image(path, channels = 3L)
```

Arguments

path	Path to image file (PNG)
channels	Number of output channels (3 for RGB, default)

Value

SD image list (width, height, channel, data as raw vector)

sd_load_model	<i>Load a registered model</i>
---------------	--------------------------------

Description

Loads a model by its registry id. Returns a cached context if already loaded, otherwise creates a new `sd_ctx`. Additional arguments override registry defaults.

Usage

```
sd_load_model(id, ...)
```

Arguments

id	Model identifier from registry
...	Additional arguments passed to <code>sd_ctx</code> , overriding registry defaults (e.g. <code>vae_decode_only = FALSE</code>)

Details

If loading fails due to insufficient VRAM, automatically unloads the least recently used model and retries.

Value

SD context (external pointer)

Examples

```
## Not run:
ctx <- sd_load_model("flux-dev")
imgs <- sd_txt2img(ctx, "a cat in space")

# Override defaults
ctx <- sd_load_model("flux-dev", vae_decode_only = FALSE, verbose = TRUE)

## End(Not run)
```

sd_load_pipeline	<i>Load pipeline from JSON</i>
------------------	--------------------------------

Description

Load pipeline from JSON

Usage

```
sd_load_pipeline(path)
```

Arguments

path Path to a JSON file saved by [sd_save_pipeline](#).

Value

An sd_pipeline object.

sd_node	<i>Create a pipeline node</i>
---------	-------------------------------

Description

Create a pipeline node

Usage

```
sd_node(type, ...)
```

Arguments

type Node type: "txt2img", "img2img", "upscale", or "save".
 ... Parameters for the node (passed to the corresponding function).

Value

A list with class "sd_node".

sd_noise_scale *Scale noise into the starting latent (low-level)*

Description

Applies the denoiser's noise scaling for the first sigma, producing the starting x for the sampling loop. For txt2img pass init_latent = NULL.

Usage

```
sd_noise_scale(ctx, noise, sigma0, init_latent = NULL)
```

Arguments

ctx SD context
 noise Noise sd_tensor (defines geometry)
 sigma0 First sigma of the schedule
 init_latent Optional starting latent (img2img); NULL for txt2img

Value

An sd_tensor — the scaled starting latent.

sd_pipeline *Create a pipeline from nodes*

Description

Nodes are executed sequentially. The image output of each node is passed as input to the next node.

Usage

```
sd_pipeline(...)
```

Arguments

... sd_node objects in execution order.

Value

A list with class "sd_pipeline".

sd_preview_start	<i>Enable live generation previews</i>
------------------	--

Description

Installs the preview callback so that, during the next generation, the most recent intermediate frame is written to path (a single PPM file, updated atomically). Poll it with [sd_read_preview](#). Call [sd_preview_stop](#) when done.

Usage

```
sd_preview_start(path, mode = PREVIEW$PROJ, interval = 1L, denoised = TRUE)
```

Arguments

path	File path for the preview PPM (e.g. a tempfile).
mode	Decode mode, one of PREVIEW: "proj" (fast, rough), "tae" (tiny autoencoder; needs taesd_path in sd_ctx), "vae" (full VAE; slow). Default "proj".
interval	Emit a preview every N sampling steps (default 1).
denoised	If TRUE (default), preview the denoised estimate; otherwise the noisy latent.

Details

Most users pass `preview = TRUE` to [sd_generate](#) instead, which wires this up automatically.

Value

Invisibly, path.

See Also

[sd_read_preview](#), [sd_preview_stop](#)

sd_preview_stop	<i>Disable live generation previews</i>
-----------------	---

Description

Removes the preview callback and cleans up the temporary .tmp file.

Usage

```
sd_preview_stop()
```

Value

Invisibly NULL.

See Also

[sd_preview_start](#)

sd_profile_get	<i>Get raw profile events</i>
----------------	-------------------------------

Description

Returns a data frame of captured events with columns stage, kind ("start"/"end"), and timestamp_ms.

Value

Data frame of profile events.

sd_profile_start	<i>Start profiling</i>
------------------	------------------------

Description

Clears the event buffer and begins capturing stage timings from sd.cpp.

Value

No return value, called for side effects.

sd_profile_stop	<i>Stop profiling</i>
-----------------	-----------------------

Description

Stops capturing stage events. Call `sd_profile_get` to retrieve.

Value

No return value, called for side effects.

sd_profile_summary	<i>Build a profile summary from raw events</i>
--------------------	--

Description

Matches start/end events by stage and computes durations.

Usage

```
sd_profile_summary(events)
```

Arguments

events	Data frame from <code>sd_profile_get()</code> with columns stage, kind, timestamp_ms.
--------	---

Value

Data frame with columns stage, start_ms, end_ms, duration_ms, duration_s. Has class "sd_profile" for pretty printing.

sd_read_preview	<i>Read the current preview frame</i>
-----------------	---------------------------------------

Description

Reads the latest preview PPM written by the running generation and returns it as an `sd_image` list. Returns NULL if no preview exists yet (e.g. generation has not produced a frame). Optionally writes a PNG copy.

Usage

```
sd_read_preview(path, png_path = NULL)
```

Arguments

path	The preview PPM path passed to sd_preview_start .
png_path	Optional path; if set, the frame is also written there as PNG via sd_save_image .

Value

An `sd_image` list (width, height, channel, data), or NULL if unavailable.

See Also

[sd_preview_start](#)

sd_register_model	<i>Register a model in the sd2R model registry</i>
-------------------	--

Description

Adds or updates a model entry in the sd2R model registry file. The registry lives in `tools::R_user_dir("sd2R", "config")` by default and can be overridden via the `SD2R_REGISTRY_DIR` environment variable. The directory is created only when a model is actually registered. Paths and defaults are stored for later use by [sd_load_model](#).

Usage

```
sd_register_model(id, model_type, paths, defaults = list(), overwrite = FALSE)
```

Arguments

id	Unique model identifier (e.g. "flux-dev", "sd15-base")
model_type	Model architecture: "sd1", "sd2", "sdxl", "flux", "flux2", "sd3"
paths	Named list of file paths. Recognized names: diffusion, model (alias for diffusion), vae, clip_l, clip_g, t5xxl, taesd, control_net.
defaults	Named list of generation defaults (optional). Recognized: steps, cfg_scale, scheduler, width, height, sample_method.
overwrite	If FALSE (default), error when id already exists

Value

Invisible model id

Examples

```

## Not run:
sd_register_model(
  id = "flux-dev",
  model_type = "flux",
  paths = list(
    diffusion = "models/flux1-dev-Q4_K_S.gguf",
    vae = "models/ae.safetensors",
    clip_l = "models/clip_l.safetensors",
    t5xxl = "models/t5xxl_fp16.safetensors"
  ),
  defaults = list(steps = 25, cfg_scale = 3.5, width = 1024, height = 1024)
)

## End(Not run)

```

sd_remove_model	<i>Remove a model from the registry</i>
-----------------	---

Description

Removes the model entry from the sd2R model registry and unloads it from memory if loaded.

Usage

```
sd_remove_model(id)
```

Arguments

id	Model identifier
----	------------------

Value

No return value, called for side effects.

sd_run_pipeline	<i>Run a pipeline</i>
-----------------	-----------------------

Description

Executes nodes sequentially. The first node must be "txt2img" (produces an image from nothing). Subsequent nodes receive the previous node's image output.

Usage

```
sd_run_pipeline(pipeline, ctx, upscaler_ctx = NULL, verbose = FALSE)
```

Arguments

pipeline	An sd_pipeline object.
ctx	A Stable Diffusion context created by sd_ctx .
upscaler_ctx	Optional upscaler context created by sd_upscale_image setup. Required if the pipeline contains an "upscale" node. Pass the result of sd_create_upscaler(path) .
verbose	Logical. Print progress messages. Default FALSE.

Value

The final image (sd_image list), or the path string if the last node is "save".

sd_sample	<i>Run the sampling loop (low-level)</i>
-----------	--

Description

Runs the full denoising loop given pre-computed conditioning and an explicit noise tensor. Noise is supplied by the caller for determinism; use `seed` to generate it reproducibly, or pass `noise` directly.

Usage

```
sd_sample(
  ctx,
  cond,
  uncond = list(crossattn = NULL, vector = NULL, concat = NULL),
  latent_shape = NULL,
  init_latent = NULL,
  noise = NULL,
  strength = 1,
  sample_method = SAMPLE_METHOD$EULER,
  scheduler = SCHEDULER$DISCRETE,
  sample_steps = 20L,
  cfg_scale = 7,
  eta = 0,
  seed = 42L,
  custom_sigmas = NULL
)
```

Arguments

ctx	SD context
cond	Positive conditioning from sd_encode_text
uncond	Negative conditioning from sd_encode_text . Pass an empty conditioning (all NULL) to disable CFG.

latent_shape	Integer vector c(W, H, C) in latent space; used to generate noise when noise is not supplied. Ignored if noise is given.
init_latent	Optional starting latent for img2img (from sd_encode_image); NULL for txt2img.
noise	Optional explicit noise sd_tensor. When NULL, standard normal noise of latent_shape is generated using seed.
strength	img2img denoising strength (ignored for txt2img)
sample_method	Sampling method (name or SAMPLE_METHOD value)
scheduler	Scheduler (name or SCHEDULER value)
sample_steps	Number of steps
cfg_scale	CFG scale
eta	Eta for DDIM-like samplers
seed	Seed for noise generation when noise is NULL
custom_sigmas	Optional explicit sigma schedule (overrides scheduler)

Value

An sd_tensor list — the denoised latent x_0. Pass to [sd_decode_latent](#).

See Also

[sd_encode_text](#), [sd_decode_latent](#)

sd_sample_stepwise *Run the sampling loop step-by-step in R (low-level)*

Description

Equivalent to [sd_sample](#) for the Euler / Euler-a samplers, but runs the loop in R so a callback can observe or interrupt each step (e.g. live preview). For Euler (no ancestral noise) the result is bit-for-bit equal to sd_sample; Euler-a differs (R RNG vs ggml RNG for the ancestral term). Other samplers are not supported here — use [sd_sample](#).

Usage

```
sd_sample_stepwise(
  ctx,
  cond,
  uncond = list(crossattn = NULL, vector = NULL, concat = NULL),
  latent_shape = NULL,
  init_latent = NULL,
  noise = NULL,
  width = 512L,
  height = 512L,
  sample_method = SAMPLE_METHOD$EULER,
```

```

    scheduler = SCHEDULER$DISCRETE,
    sample_steps = 20L,
    cfg_scale = 7,
    seed = 42L,
    custom_sigmas = NULL,
    on_step = NULL
)

```

Arguments

ctx	SD context
cond	Positive conditioning from sd_encode_text
uncond	Negative conditioning; empty (all NULL) disables CFG
latent_shape	Integer c(W, H, C) in latent space, used to make noise when noise is NULL
init_latent	Optional starting latent (img2img); NULL for txt2img
noise	Optional explicit noise sd_tensor; generated from seed and latent_shape when NULL
width, height	Generation size in PIXELS (for the sigma schedule)
sample_method	SAMPLE_METHOD\$EULER or \$EULER_A
scheduler	Scheduler (name or SCHEDULER value)
sample_steps	Number of steps
cfg_scale	CFG scale
seed	Seed for noise generation when noise is NULL
custom_sigmas	Optional explicit sigma schedule (overrides scheduler)
on_step	Optional callback function(step, total, x, denoised) called after each step; return FALSE to stop early.

Value

An sd_tensor — the denoised latent x_0.

See Also

[sd_sample](#), [sd_decode_latent](#)

sd_sampler_begin	<i>Open / close a step-wise sampling window (low-level)</i>
------------------	---

Description

Between begin and end the diffusion model keeps its GPU compute buffer alive across [sd_denoise_step](#) calls, avoiding a large realloc per step. Must be paired; sd_sampler_end frees the buffer. Not reentrant. [sd_sample_stepwise](#) manages this for you.

Usage

```
sd_sampler_begin(ctx)
```

```
sd_sampler_end(ctx)
```

Arguments

ctx	SD context
-----	------------

Value

Invisibly NULL.

sd_sampler_sigmas	<i>Sigma schedule for a sampler (low-level)</i>
-------------------	---

Description

Returns the sigma schedule that [sd_sample_stepwise](#) iterates over, for a given scheduler / step count / generation size.

Usage

```
sd_sampler_sigmas(
  ctx,
  scheduler = SCHEDULER$DISCRETE,
  sample_steps = 20L,
  width = 512L,
  height = 512L,
  sample_method = SAMPLE_METHOD$EULER
)
```

Arguments

ctx	SD context from sd_ctx
scheduler	Scheduler (name or SCHEDULER value)
sample_steps	Number of steps
width, height	Generation size in PIXELS (same as passed to generation)
sample_method	Sampling method (name or SAMPLE_METHOD value); only used to pick a default scheduler when scheduler is a default.

Value

Numeric vector of length `sample_steps + 1`; the last value is 0.

See Also

[sd_denoise_step](#), [sd_sample_stepwise](#)

sd_save_image	<i>Save SD image to PNG file</i>
---------------	----------------------------------

Description

Save SD image to PNG file

Usage

```
sd_save_image(image, path)
```

Arguments

image	SD image (list with width, height, channel, data) as returned by <code>sd_txt2img()</code> or <code>sd_img2img()</code> . Can also be a 3D numeric array [height, width, channels] with values in [0, 1].
path	Output file path (should end in .png)

Value

The file path (invisibly).

sd_save_pipeline	<i>Save pipeline to JSON</i>
------------------	------------------------------

Description

Save pipeline to JSON

Usage

```
sd_save_pipeline(pipeline, path)
```

Arguments

pipeline	An <code>sd_pipeline</code> object.
path	File path (should end in .json).

Value

The file path, invisibly.

sd_scan_models	<i>Scan a directory for models and register them</i>
----------------	--

Description

Scans for .safetensors and .gguf files, guesses component roles and model types from filenames, groups multi-file models (Flux), and registers them.

Usage

```
sd_scan_models(dir, overwrite = FALSE, recursive = FALSE)
```

Arguments

dir	Directory to scan
overwrite	If TRUE, overwrite existing entries (default FALSE)
recursive	Scan subdirectories (default FALSE)

Details

Single-file models (SD 1.5, SDXL) are registered individually. Multi-file Flux models are grouped when diffusion + supporting files (VAE, CLIP, T5) are found in the same directory.

Value

Character vector of registered model ids (invisible)

Examples

```
## Not run:
sd_scan_models("/mnt/models/")
sd_list_models()

## End(Not run)
```

sd_supports_ref_images	<i>Does the loaded model support reference images?</i>
------------------------	--

Description

Reports whether the model in ctx consumes reference images (edit / control / DiT families: Flux, Flux.2, SD3, Qwen-Image, Z-Image). Passing refs to other models aborts inside ggml, so [sd_generate_multiref](#) uses this to fail cleanly first.

Usage

```
sd_supports_ref_images(ctx)
```

Arguments

ctx SD context from [sd_ctx](#)

Value

Logical scalar.

sd_system_info	<i>Get system information</i>
----------------	-------------------------------

Description

Returns information about the stable-diffusion.cpp backend.

Usage

```
sd_system_info()
```

Value

List with system info, version, and core count

sd_txt2img	<i>Generate images from text prompt</i>
------------	---

Description

Generate images from text prompt

Usage

```
sd_txt2img(
  ctx,
  prompt,
  negative_prompt = "",
  width = 512L,
  height = 512L,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
```

```

batch_count = 1L,
scheduler = SCHEDULER$DISCRETE,
clip_skip = -1L,
eta = 0,
control_image = NULL,
control_strength = 0.9,
vae_mode = "auto",
vae_auto_threshold = 1048576L,
vae_tile_size = 64L,
vae_tile_overlap = 0.25,
vae_tile_rel_x = NULL,
vae_tile_rel_y = NULL,
vae_tiling = NULL,
cache_mode = c("off", "easy", "ucache"),
cache_config = NULL
)

```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt describing desired image
negative_prompt	Negative prompt (default "")
width	Image width in pixels (default 512)
height	Image height in pixels (default 512)
sample_method	Sampling method (see <code>SAMPLE_METHOD</code>)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see <code>SCHEDULER</code>)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
control_image	Optional control image for ControlNet (sd_image format)
control_strength	ControlNet strength (default 0.9)
vae_mode	VAE processing mode: "normal" (no tiling), "tiled" (always tile), or "auto" (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default "auto".
vae_auto_threshold	Pixel area fallback threshold for vae_mode = "auto" when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when width * height exceeds this value. Default 1048576L (1024x1024 pixels).

vae_tile_size	Tile size in latent pixels for tiled VAE (default 64). Ignored when vae_tile_rel_x/vae_tile_rel_y are set.
vae_tile_overlap	Overlap ratio between tiles, 0.0-0.5 (default 0.25)
vae_tile_rel_x	Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tile_rel_y	Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tiling	Deprecated. Use vae_mode instead. If TRUE, equivalent to vae_mode = "tiled".
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache — skips redundant denoising steps), or "ucache" (UCache). Can speed up sampling 20-40% with minor quality impact.
cache_config	Optional fine-tuned cache config from sd_cache_params . Overrides cache_mode when provided.

Value

List of SD images. Each image is a list with width, height, channel, and data (raw vector of RGB pixels). Use [sd_save_image](#) to save or [sd_image_to_array](#) to convert.

sd_txt2img_highres *High-resolution image generation via patch-based pipeline*

Description

Generates a large image by independently rendering overlapping patches at the model's native resolution, then stitching them with linear blending. An optional img2img harmonization pass can smooth seams further.

Usage

```
sd_txt2img_highres(
  ctx,
  prompt,
  negative_prompt = "",
  width = 2048L,
  height = 2048L,
  tile_size = NULL,
  overlap = 0.125,
  img2img_strength = NULL,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
```

```

    eta = 0,
    vae_mode = "auto",
    vae_auto_threshold = 1048576L,
    vae_tile_size = 64L,
    vae_tile_overlap = 0.25
)

```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt
negative_prompt	Negative prompt (default "")
width	Target image width in pixels
height	Target image height in pixels
tile_size	Patch size in pixels. NULL = auto-detect from model_type attribute on ctx (512 for SD1/SD2, 1024 for SDXL/Flux/SD3). Must be divisible by 8.
overlap	Overlap between patches as fraction of tile_size, 0.0-0.5 (default 0.125).
img2img_strength	If not NULL, run a final img2img pass over the stitched image at this denoising strength (e.g. 0.3) to harmonize seams. Requires vae_decode_only = FALSE in the context. Default NULL (disabled).
sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Base random seed. Each patch gets seed + patch_index. Use -1 for random.
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
vae_mode	VAE tiling mode for the harmonization pass (default "auto": VRAM-aware, see sd_txt2img).
vae_auto_threshold	Pixel area fallback threshold for auto VAE tiling when VRAM query is unavailable
vae_tile_size	Tile size for VAE tiling (default 64)
vae_tile_overlap	Overlap for VAE tiling (default 0.25)

Value

SD image (list with width, height, channel, data)

Examples

```
## Not run:
ctx <- sd_ctx("sd15.safetensors", model_type = "sd1")
img <- sd_txt2img_highres(ctx, "a panoramic mountain landscape",
                          width = 2048, height = 1024)
sd_save_image(img, "panorama.png")

## End(Not run)
```

sd_txt2img_tiled *Tiled diffusion sampling (MultiDiffusion)*

Description

Generates images at any resolution using tiled sampling: at each denoising step the latent is split into overlapping tiles, each tile is denoised independently by the UNet, and results are merged with Gaussian weighting. VRAM usage is bounded by tile size, not output resolution.

Usage

```
sd_txt2img_tiled(
  ctx,
  prompt,
  negative_prompt = "",
  width = 2048L,
  height = 2048L,
  sample_tile_size = NULL,
  sample_tile_overlap = 0.25,
  sample_method = SAMPLE_METHOD$EULER,
  sample_steps = 20L,
  cfg_scale = 7,
  seed = 42L,
  batch_count = 1L,
  scheduler = SCHEDULER$DISCRETE,
  clip_skip = -1L,
  eta = 0,
  vae_mode = "auto",
  vae_auto_threshold = 1048576L,
  vae_tile_size = 64L,
  vae_tile_overlap = 0.25,
  vae_tile_rel_x = NULL,
  vae_tile_rel_y = NULL,
  cache_mode = c("off", "easy", "ucache"),
  cache_config = NULL
)
```

Arguments

ctx	SD context created by sd_ctx
prompt	Text prompt describing desired image
negative_prompt	Negative prompt (default "")
width	Target image width in pixels (can exceed model native resolution)
height	Target image height in pixels
sample_tile_size	Tile size in latent pixels (default NULL = auto from model_type: 64 for SD1/SD2, 128 for SDXL/Flux/SD3). One latent pixel = vae_scale_factor image pixels (typically 8).
sample_tile_overlap	Overlap between tiles as fraction of tile size, 0.0-0.5 (default 0.25).
sample_method	Sampling method (see SAMPLE_METHOD)
sample_steps	Number of sampling steps (default 20)
cfg_scale	Classifier-free guidance scale (default 7.0)
seed	Random seed (-1 for random)
batch_count	Number of images to generate (default 1)
scheduler	Scheduler type (see SCHEDULER)
clip_skip	Number of CLIP layers to skip (-1 = auto)
eta	Eta parameter for DDIM-like samplers
vae_mode	VAE processing mode: "normal" (no tiling), "tiled" (always tile), or "auto" (VRAM-aware: queries free GPU memory via Vulkan and compares against estimated peak VAE usage; tiles only when VRAM is insufficient). Default "auto".
vae_auto_threshold	Pixel area fallback threshold for vae_mode = "auto" when VRAM query is unavailable (no Vulkan, CPU backend, etc.). Tiling activates when width * height exceeds this value. Default 1048576L (1024x1024 pixels).
vae_tile_size	Tile size in latent pixels for tiled VAE (default 64). Ignored when vae_tile_rel_x/vae_tile_rel_y are set.
vae_tile_overlap	Overlap ratio between tiles, 0.0-0.5 (default 0.25)
vae_tile_rel_x	Relative tile width as fraction of latent width (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
vae_tile_rel_y	Relative tile height as fraction of latent height (0-1) or number of tiles (>1). NULL = use vae_tile_size. Takes priority over vae_tile_size.
cache_mode	Step caching mode: "off" (default), "easy" (EasyCache — skips redundant denoising steps), or "ucache" (UCache). Can speed up sampling 20-40% with minor quality impact.
cache_config	Optional fine-tuned cache config from sd_cache_params . Overrides cache_mode when provided.

Details

Requires tiled VAE (enabled automatically via `vae_mode = "auto"`).

Value

List of SD images

Examples

```
## Not run:
ctx <- sd_ctx("sd15.safetensors", model_type = "sd1")
imgs <- sd_txt2img_tiled(ctx, "a vast mountain landscape",
                        width = 2048, height = 1024)
sd_save_image(imgs[[1]], "landscape.png")

## End(Not run)
```

SD_TYPE

Weight types (ggml quantization types)

Description

Weight types (ggml quantization types)

Usage

SD_TYPE

Format

An object of class `list` of length 15.

sd_unload_all

Unload all models from memory

Description

Removes all cached contexts. Registry is preserved.

Usage

`sd_unload_all()`

Value

No return value, called for side effects.

sd_unload_model	<i>Unload a model from memory</i>
-----------------	-----------------------------------

Description

Removes the cached context for the given model id. The model remains in the registry and can be reloaded with [sd_load_model](#).

Usage

```
sd_unload_model(id)
```

Arguments

id	Model identifier
----	------------------

Value

No return value, called for side effects.

sd_upscale_image	<i>Upscale an image using ESRGAN</i>
------------------	--------------------------------------

Description

Upscale an image using ESRGAN

Usage

```
sd_upscale_image(esrgan_path, image, upscale_factor = 4L, n_threads = 0L)
```

Arguments

esrgan_path	Path to ESRGAN model file
image	SD image to upscale (list with width, height, channel, data)
upscale_factor	Upscale factor (default 4)
n_threads	Number of CPU threads (0 = auto-detect)

Value

Upscaled SD image

`sd_vulkan_device_count`*Get number of Vulkan GPU devices*

Description

Returns the number of Vulkan-capable GPU devices available on the system. Useful for deciding whether to use [sd_generate_multi_gpu](#).

Usage

```
sd_vulkan_device_count()
```

Value

Integer, number of Vulkan devices (0 if Vulkan is not available)

Index

* datasets

- LORA_APPLY_MODE, 3
 - PREDICTION, 3
 - PREVIEW, 4
 - RNG_TYPE, 4
 - SAMPLE_METHOD, 4
 - SCHEDULER, 5
 - SD_CACHE_MODE, 7
 - SD_TYPE, 45
- LORA_APPLY_MODE, 3
- PREDICTION, 3
- PREVIEW, 4
- RNG_TYPE, 4
- runApp, 7
- SAMPLE_METHOD, 4
- SCHEDULER, 5
- sd_api_start, 5
- sd_api_stop, 6
- sd_app, 7
- SD_CACHE_MODE, 7
- sd_cache_params, 8, 17, 23, 41, 44
- sd_convert, 8
- sd_ctx, 6, 9, 13, 15–18, 20, 22, 25, 28, 33, 36, 39, 40, 42, 44
- sd_decode_latent, 12, 15, 16, 34, 35
- sd_default_params, 13, 17
- sd_denoise_step, 14, 35, 36
- sd_encode_image, 12, 15, 21, 34
- sd_encode_text, 14, 15, 33–35
- sd_generate, 11, 13, 15, 16, 18, 19, 21, 28
- sd_generate_multi_gpu, 18, 47
- sd_generate_multiref, 20, 38
- sd_image_to_array, 21, 41
- sd_img2img, 17, 22
- sd_inverse_noise_scale, 24
- sd_list_models, 24
- sd_load_image, 15, 20, 22, 25
- sd_load_model, 25, 31, 46
- sd_load_pipeline, 26
- sd_node, 26
- sd_noise_scale, 27
- sd_pipeline, 27
- sd_preview_start, 4, 17, 28, 29, 31
- sd_preview_stop, 28, 29
- sd_profile_get, 29, 30
- sd_profile_start, 29
- sd_profile_stop, 30
- sd_profile_summary, 30
- sd_read_preview, 17, 28, 30
- sd_register_model, 31
- sd_remove_model, 32
- sd_run_pipeline, 32
- sd_sample, 12, 15, 16, 33, 34, 35
- sd_sample_stepwise, 14, 34, 35, 36
- sd_sampler_begin, 14, 35
- sd_sampler_end, 14
- sd_sampler_end (sd_sampler_begin), 35
- sd_sampler_sigmas, 36
- sd_save_image, 12, 31, 37, 41
- sd_save_pipeline, 26, 37
- sd_scan_models, 38
- sd_supports_ref_images, 38
- sd_system_info, 39
- sd_txt2img, 17, 39, 42
- sd_txt2img_highres, 41
- sd_txt2img_tiled, 43
- SD_TYPE, 45
- sd_unload_all, 45
- sd_unload_model, 46
- sd_upscale_image, 33, 46
- sd_vulkan_device_count, 47