# Pet Style Generation Service - Integration Guide

## Overview

This service transforms pet photos into specific artistic styles using **DreamShaper 8** (a fine-tuned Stable Diffusion 1.5 model). It runs locally and generates high-aesthetic outputs.

## Architecture

* **Model**: Lykon/dreamshaper-8 (Auto-downloaded on first run ~4GB).
* **Scheduler**: DPMSolver++ (Optimized for speed and sharpness).
* **Input**: Local file path (JPEG/PNG).
* **Output**: Local file path (saved image).

## Setup Instructions

1. **System Requirements**:
   * **OS**: Linux (Ubuntu 20.04+), Windows, or macOS.
   * **GPU**: NVIDIA GPU (8GB+ VRAM) recommended for <5s generation.
   * **CPU/Mac**: Supported but slower (30s - 3mins).
2. **Installation**:  
   chmod +x setup\_env.sh  
   ./setup\_env.sh

## Python API Contract

### 1. Initialization

Initialize the service **once** at application startup.

from pet\_style\_service import PetStyleService  
  
# Initialize once (loads model into VRAM)  
service = PetStyleService()

### 2. Transformation

The transform\_image method now requires a species parameter to prevent species-swapping (e.g., turning a dog into a cat).

output\_path = service.transform\_image(  
 input\_image\_path="/tmp/uploads/user\_photo.jpg",  
 output\_path="/tmp/results/processed\_image.png",  
 style\_prompt="Cyberpunk style, neon lights",  
 species="dog", # CRITICAL: 'dog', 'cat', 'hamster', etc.  
 strength=0.75 # Optional: 0.6 to 0.8 is the sweet spot  
)

| **Parameter** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| input\_image\_path | str | Required | Absolute path to the source image. |
| output\_path | str | Required | Absolute path where the result will be saved. |
| style\_prompt | str | Required | The artistic style (e.g., "Pixar style"). |
| species | str | "dog" | **New:** The animal type. Helps the AI maintain identity. |
| strength | float | 0.75 | Creativity level (0.0 - 1.0). Higher = more style, less original structure. |

### 3. Error Handling

The method returns None if generation fails.

result = service.transform\_image(...)  
if result is None:  
 # Handle error (e.g., return 500 status to frontend)  
 print("Image generation failed.")

## Troubleshooting Common Issues

* **final\_sigmas\_type zero is not supported**: This means diffusers is trying to use an incompatible scheduler algorithm. Ensure you are using the provided pet\_style\_service.py which explicitly sets algorithm\_type="dpmsolver++".
* **"It looks like a human"**: Reduce strength to 0.65 or ensure species is set correctly. The service has built-in negative prompts to suppress human features.
* **Slow Performance**: Ensure PyTorch is using CUDA.  
  import torch  
  print(torch.cuda.is\_available()) # Should be True