# Swarms API Documentation

The Swarms API provides endpoints to interact with various language models, manage agent configurations, and handle token counting. This documentation covers the available endpoints, input and output models, and detailed examples for each endpoint.

URL: `https://api.swarms.world`

## Key Features
- Dynamic Model Switching: Easily switch between different language models based on user input.
- Token Counting: Efficiently count tokens using the tiktoken library.
- Agent Configuration: Configure and run agents with detailed settings for various tasks.
- CORS Handling: Support for Cross-Origin Resource Sharing (CORS) to allow web-based clients to interact with the API.

## Endpoints

### `/v1/models`

Method: `GET`

Response Model: `List[str]`

Description:
This endpoint returns a list of available model names. It is useful for clients to query and understand

which models are available for use.

Response Example:

```json
[
    "OpenAIChat",
    "GPT4VisionAPI",
    "Anthropic"
]
```

Example Usage:

```python
import requests

response = requests.get("http://api.swarms.world/v1/models")
print(response.json())
```

### `/v1/agent/completions`

Method: `POST`

Request Model: `AgentInput`

Response Model: `AgentOutput`

URL: `http://api.swarms.world/v1/agent/completions`

Description:
This endpoint handles the completion request for an agent configured with the given input parameters. It processes the request and returns the completion results.

Request Example:

```json
{
    "agent_name": "Swarm Agent",
    "system_prompt": "Summarize the following text",
    "agent_description": "An agent that summarizes text",
    "model_name": "OpenAIChat",
    "max_loops": 1,
    "autosave": false,
    "dynamic_temperature_enabled": false,
    "dashboard": false,
    "verbose": false,
    "streaming_on": true,
    "saved_state_path": null,
    "sop": null,
    "sop_list": null,
    "user_name": "User",
    "retry_attempts": 3,
    "context_length": 8192,
    "task": "This is a sample text that needs to be summarized."
}
```

Response Example:

```json
{
    "agent": {
        "agent_name": "Swarm Agent",
        "system_prompt": "Summarize the following text",
        "agent_description": "An agent that summarizes text",
        "model_name": "OpenAIChat",
        "max_loops": 1,
        "autosave": false,
        "dynamic_temperature_enabled": false,
        "dashboard": false,
        "verbose": false,
        "streaming_on": true,
        "saved_state_path": null,
        "sop": null,
        "sop_list": null,
        "user_name": "User",
        "retry_attempts": 3,
        "context_length": 8192,
        "task": "This is a sample text that needs to be summarized."
    },
    "completions": {
        "choices": [
            {
                "index": 0,
                "message": {
                    "role": "Swarm Agent",
                    "content": "The sample text summarizes how to perform text summarization using an agent.",
                    "name": null
                }
            }
        ],
        "stream_choices": null,
        "usage_info": {
            "prompt_tokens": 10,
            "completion_tokens": 15,
            "total_tokens": 25
        }
    }
}
```

Example Usage:

```python
import requests
from pydantic import BaseModel
from typing import List

class AgentInput(BaseModel):
    agent_name: str = "Swarm Agent"
    system_prompt: str = None
    agent_description: str = None
    model_name: str = "OpenAIChat"
    max_loops: int = 1
    autosave: bool = False
    dynamic_temperature_enabled: bool = False
    dashboard: bool = False
    verbose: bool = False
    streaming_on: bool = True
    saved_state_path: str = None
    sop: str = None
    sop_list: List[str] = None
    user_name: str = "User"
    retry_attempts: int = 3
    context_length: int = 8192
    task: str = None
```

```
agent_input = AgentInput(task="Generate a summary of the provided text.")
response = requests.post("http://api.swarms.world/v1/agent/completions", json=agent_input.dict())
print(response.json())
```

## Models

### AgentInput

The `AgentInput` class defines the structure of the input data required to configure and run an agent.

Parameter	Type	Default	Description
`agent_name`	`str`	"Swarm Agent"	The name of the agent.
`system_prompt`	`str` or `None`	`None`	The system prompt to guide the agent's behavior.
`agent_description`	`str` or `None`	`None`	A description of the agent's purpose.
`model_name`	`str`	"OpenAIChat"	The name of the language model to use.
`max_loops`	`int`	1	The maximum number of loops the agent should perform.
`autosave`	`bool`	`False`	Whether to enable autosave functionality.

|

`dynamic_temperature_enabled`	`bool`	`False`	Whether dynamic temperature adjustment is enabled.
`dashboard`	`bool`	`False`	Whether to enable the dashboard feature.
`verbose`	`bool`	`False`	Whether to enable verbose logging.
`streaming_on`	`bool`	`True`	Whether to enable streaming of responses.
`saved_state_path`	`str` or `None`	`None`	Path to save the agent's state.
`sop`	`str` or `None`	`None`	Standard operating procedures for the agent.
`sop_list`	`List[str]` or `None`	`None`	A list of standard operating procedures.
`user_name`	`str`	"User"	The name of the user interacting with the agent.
`retry_attempts`	`int`	3	Number of retry attempts for failed operations.
`context_length`	`int`	8192	Maximum context length for the model's input.
`task`	`str` or `None`	`None`	The task description for the agent to perform.

### AgentOutput

The `AgentOutput` class defines the structure of the output data returned by the agent after processing a request.

Parameter	Type	Description
`agent`	`AgentInput`	The input configuration used to create the agent.
`completions`	`ChatCompletionResponse`	The response generated by the agent.

## Functions

### count_tokens

The `count_tokens` function counts the number of tokens in a given text using the `tiktoken` library.

Parameters:

- `text` (`str`): The text to be tokenized and counted.

Returns:

- `int`: The number of tokens in the text.

Example Usage:

```python
text = "This is a sample text to count tokens."
```

token_count = count_tokens(text)
print(f"Token count: {token_count}")

### model_router

The `model_router` function switches to the specified language model based on the provided model name.

Parameters:

- `model_name` (`str`): The name of the model to switch to.

Returns:

- An instance of the specified language model.

Example Usage:

```python
model_name = "OpenAIChat"
model_instance = model_router(model_name)
```

## Additional Information and Tips

- **Error Handling**: Ensure robust error handling by catching exceptions and returning meaningful HTTP status codes and messages.

- **Model Selection**: When adding new models, update the \`model_router\` function and the \`/v1/models\` endpoint to include the new model names.

- **Token Management**: Keep track of token usage to optimize API costs and manage rate limits effectively.