LangChain Docs

Core components › Structured output

Core components

Structured output

Copy page

Structured output allows agents to return data in a specific, predictable format. Instead of parsing natural language responses, you get typed structured data.

LangChain's prebuilt ReAct agent `createAgent` handles structured output automatically. The user sets their desired structured output schema, and when the model generates the structured data, it's captured, validated, and returned in the `structuredResponse` key of the agent's state.

```
type ResponseFormat = (
    | ZodSchema<StructuredResponseT> // a Zod schema
    | Record<string, unknown> // a JSON Schema
)

const agent = createAgent({
    // ...
    responseFormat: ResponseFormat | ResponseFormat[]
})
```

Response format

Controls how the agent returns structured data. You can provide either a Zod object or JSON schema. By default, the agent uses a tool calling strategy, in which the output is created by an additional tool call. Certain models support native structured output, in which case the agent will use that strategy instead.

You can control the behavior by wrapping `ResponseFormat` in a `toolStrategy` or

LangChain Docs

Core components › **Structured output**

```
const agent = createAgent({
    // use a provider strategy if supported by the model
    responseFormat: providerStrategy(z.object({ ... }))
    // or enforce a tool strategy
    responseFormat: toolStrategy(z.object({ ... }))
})
```

The structured response is returned in the `structuredResponse` key of the agent's final state.

> Support for native structured output features is read dynamically from the model's **profile data** if using `langchain>=1.1` . If data are not available, use another condition or specify manually:
>
> ```
> const customProfile: ModelProfile = {
>     structuredOutput: true,
>     // ...
> }
> const model = await initChatModel("...", { profile: customProfile });
> ```
>
> If tools are specified, the model must support simultaneous use of tools and structured output.

Provider strategy

Some model providers support structured output natively through their APIs (e.g. OpenAI, xAI (Grok), Gemini, Anthropic (Claude)). This is the most reliable method when available.

To use this strategy, configure a `ProviderStrategy` :

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

LangChain Docs

Core components › **Structured output**

```
function providerStrategy<StructuredResponseT>(
  schema: ZodSchema<StructuredResponseT> | JsonSchemaFormat
): ProviderStrategy<StructuredResponseT>
```

LangChain Docs

Core components › **Structured output**

Zod Schema: A zod schema

JSON Schema: A JSON schema object

LangChain automatically uses `ProviderStrategy` when you pass a schema type directly to `createAgent.responseFormat` and the model supports native structured output:

Zod Schema JSON Schema

```
import * as z from "zod";
import { createAgent, providerStrategy } from "langchain";

const ContactInfo = z.object({
    name: z.string().describe("The name of the person"),
    email: z.string().describe("The email address of the person"),
    phone: z.string().describe("The phone number of the person"),
});

const agent = createAgent({
    model: "gpt-5",
    tools: [],
    responseFormat: providerStrategy(ContactInfo)
});

const result = await agent.invoke({
    messages: [{"role": "user", "content": "Extract contact info from: John Doe
});

console.log(result.structuredResponse);
// { name: "John Doe", email: "john@example.com", phone: "(555) 123-4567" }
```

ider-native structured output provides high reliability and strict validation because the model provider enforces the schema. Use it when available.

strategy.

Tool calling strategy

For models that don't support native structured output, LangChain uses tool calling to achieve the same result. This works with all models that support tool calling (most modern models).

To use this strategy, configure a `ToolStrategy`:

```
function toolStrategy<StructuredResponseT>(
    responseFormat:
        | JsonSchemaFormat
        | ZodSchema<StructuredResponseT>
        | (ZodSchema<StructuredResponseT> | JsonSchemaFormat)[]
    options?: ToolStrategyOptions
): ToolStrategy<StructuredResponseT>
```

`schema` required

The schema defining the structured output format. Supports:

Zod Schema: A zod schema

JSON Schema: A JSON schema object

`options.toolMessageContent`

Custom content for the tool message returned when structured output is generated. If not provided, defaults to a message showing the structured response data.

`true` : Catch all errors with default error template (default)

`False` : No retry, let exceptions propagate

`(error: ToolStrategyError) => string | Promise<string>` : retry with the provided message or throw the error

Zod Schema JSON Schema Union Types

```
import * as z from "zod";
import { createAgent, toolStrategy } from "langchain";

const ProductReview = z.object({
    rating: z.number().min(1).max(5).optional(),
    sentiment: z.enum(["positive", "negative"]),
    keyPoints: z.array(z.string()).describe("The key points of the review. Lowe
});

const agent = createAgent({
    model: "gpt-5",
    tools: [],
    responseFormat: toolStrategy(ProductReview)
})

const result = await agent.invoke({
    "messages": [{"role": "user", "content": "Analyze this review: 'Great produ
})

console.log(result.structuredResponse);
// { "rating": 5, "sentiment": "positive", "keyPoints": ["fast shipping", "expe
```

Custom tool message content

`toolMessageContent` parameter allows you to customize the message that appears

LangChain Docs

Core components › **Structured output**

```
const MeetingAction = z.object({
    task: z.string().describe("The specific task to be completed"),
    assignee: z.string().describe("Person responsible for the task"),
    priority: z.enum(["low", "medium", "high"]).describe("Priority level"),
});

const agent = createAgent({
    model: "gpt-5",
    tools: [],
    responseFormat: toolStrategy(MeetingAction, {
        toolMessageContent: "Action item captured and added to meeting notes!"
    })
});

const result = await agent.invoke({
    messages: [{"role": "user", "content": "From our meeting: Sarah needs to up
});

console.log(result);
/**
 * {
 *   messages: [
 *     { role: "user", content: "From our meeting: Sarah needs to update the pr
 *     { role: "assistant", content: "Action item captured and added to meeting
 *     { role: "tool", content: "Action item captured and added to meeting note
 *   ],
 *   structuredResponse: { task: "update the project timeline", assignee: "Sara
 * }
 */
```

Without `toolMessageContent`, we'd see:

```
*     ...
*     { role: "tool", content: "Returning structured response: {'task': 'updat
*   ],
*   structuredResponse: { task: "update the project timeline", assignee: "Sara
* }
*/
```

Error handling

Models can make mistakes when generating structured output via tool calling. LangChain provides intelligent retry mechanisms to handle these errors automatically.

Multiple structured outputs error

When a model incorrectly calls multiple structured output tools, the agent provides error feedback in a `ToolMessage` and prompts the model to retry:

LangChain Docs

Core components › Structured output

```
    name: z.string().describe("Person's name"),
    email: z.string().describe("Email address"),
});

const EventDetails = z.object({
    event_name: z.string().describe("Name of the event"),
    date: z.string().describe("Event date"),
});

const agent = createAgent({
    model: "gpt-5",
    tools: [],
    responseFormat: toolStrategy([ContactInfo, EventDetails]),
});

const result = await agent.invoke({
    messages: [
        {
        role: "user",
        content:
            "Extract info: John Doe (john@email.com) is organizing Tech Confere
        },
    ],
});

console.log(result);

/**
 * {
 *   messages: [
 *     { role: "user", content: "Extract info: John Doe (john@email.com) is org
 *     { role: "assistant", content: "", tool_calls: [ { name: "ContactInfo", a
 *     { role: "tool", content: "Error: Model incorrectly returned multiple str
       { role: "tool", content: "Error: Model incorrectly returned multiple str
 .     { role: "assistant", content: "", tool_calls: [ { name: "ContactInfo", a
```

Schema validation error

When structured output doesn't match the expected schema, the agent provides specific error feedback:

```
    rating: z.number().min(1).max(5).describe("Rating from 1-5"),
    comment: z.string().describe("Review comment"),
});

const agent = createAgent({
    model: "gpt-5",
    tools: [],
    responseFormat: toolStrategy(ProductRating),
});

const result = await agent.invoke({
    messages: [
        {
        role: "user",
        content: "Parse this: Amazing product, 10/10!",
        },
    ],
});

console.log(result);

/**
 * {
 *   messages: [
 *     { role: "user", content: "Parse this: Amazing product, 10/10!" },
 *     { role: "assistant", content: "", tool_calls: [ { name: "ProductRating",
 *     { role: "tool", content: "Error: Failed to parse structured output for t
 *     { role: "assistant", content: "", tool_calls: [ { name: "ProductRating",
 *     { role: "tool", content: "Returning structured response: {'rating': 5, '
 *   ],
 *   structuredResponse: { rating: 5, comment: "Amazing product" }
 * }
 */
```

```
const responseFormat = toolStrategy(ProductRating, {
    handleError: "Please provide a valid rating between 1-5 and include a comme
)

// Error message becomes:
// { role: "tool", content: "Please provide a valid rating between 1-5 and incl
```

Handle specific exceptions only:

```
import { ToolInputParsingException } from "@langchain/core/tools";

const responseFormat = toolStrategy(ProductRating, {
    handleError: (error: ToolStrategyError) => {
        if (error instanceof ToolInputParsingException) {
        return "Please provide a valid rating between 1-5 and include a comment
        }
        return error.message;
    }
)

// Only validation errors get retried with default message:
// { role: "tool", content: "Error: Failed to parse structured output for tool
```

Handle multiple exception types:

```
        }
        if (error instanceof CustomUserError) {
        return "This is a custom user error.";
        }
        return error.message;
    }
)
```

No error handling:

```
const responseFormat = toolStrategy(ProductRating, {
    handleError: false  // All errors raised
)
```

LangChain Docs

Core components › **Structured output**

Resources

Forum

Changelog

LangChain Academy

Trust Center

Company

About

Careers

Blog

Powered by mintlify

LangChain Docs

Core components › **Structured output**