Foundation Models / Tool

Protocol

Tool

A tool that a model can call to gather information at runtime or perform side effects.

iOS 26.0+ | iPadOS 26.0+ | Mac Catalyst 26.0+ | macOS 26.0+ | visionOS 26.0+

```
protocol Tool<Arguments, Output> : Sendable
```

Mentioned in

- Generating content and performing tasks with Foundation Models
- Categorizing and organizing data with content tags
- Expanding generation with tool calling

Overview

Tool calling gives the model the ability to call your code to incorporate up-to-date information like recent events and data from your app. A tool includes a name and a description that the framework puts in the prompt to let the model decide when and how often to call your tool.

A `Tool` defines a `call(arguments:)` method that takes arguments that conforms to `ConvertibleFromGeneratedContent`, and returns an output of any type that conforms to `PromptRepresentable`, allowing the model to understand and reason about in subsequent interactions. Typically, `Output` is a `String` or any `Generable` types.

```
struct FindContacts: Tool {
    let name = "findContacts"
    let description = "Find a specific number of contacts"
```

```swift
    @Generable
    struct Arguments {
        @Guide(description: "The number of contacts to get", .range(1...10))
        let count: Int
    }

    func call(arguments: Arguments) async throws -> [String] {
        var contacts: [CNContact] = []
        // Fetch a number of contacts using the arguments.
        let formattedContacts = contacts.map {
            "\($0.givenName) \($0.familyName)"
        }
        return formattedContacts
    }
}
```

Tools must conform to `Sendable` so the framework can run them concurrently. If the model needs to pass the output of one tool as the input to another, it executes back-to-back tool calls.

You control the life cycle of your tool, so you can track the state of it between calls to the model. For example, you might store a list of database records that you don't want to reuse between tool calls.

Prompting the model with tools contributes to the available context window size. When you provide a tool in your generation request, the framework puts the tool definitions — name, description, parameter information — in the prompt so the model can decide when and how often to call the tool. After calling your tool, the framework returns the tool's output back to the model for further processing.

To efficiently use tool calling:

- Reduce `Guide(description:)` descriptions to a short phrase each.
- Limit the number of tools you use to three to five.
- Include a tool only when its necessary for the task you want to perform.
- Run an essential tool before calling the model and integrate the tool's output in the prompt directly.

If your session exceeds the available context size, it throws `LanguageModelSession.GenerationError.exceededContextWindowSize(_:)`. When you encounter the context window limit, consider breaking up tool calls across new `LanguageModelSession` instances. For more information on managing the context window size, see TN3193: Managing the on-device foundation model's context window.

Topics

Invoking a tool

```
func call(arguments: Self.Arguments) async throws -> Self.Output
```

A language model will call this method when it wants to leverage this tool.

Required

```
associatedtype Arguments : ConvertibleFromGeneratedContent
```

The arguments that this tool should accept.

Required

```
associatedtype Output : PromptRepresentable
```

The output that this tool produces for the language model to reason about in subsequent interactions.

Required

Getting the tool properties

```
var description: String
```

A natural language description of when and how to use the tool.

Required

```
var includesSchemaInInstructions: Bool
```

If true, the model's name, description, and parameters schema will be injected into the instructions of sessions that leverage this tool.

Required Default implementation provided.

```
var name: String
```

A unique name for the tool, such as "get_weather", "toggleDarkMode", or "search contacts".

Required Default implementation provided.

```
var parameters: GenerationSchema
```

A schema for the parameters this tool accepts.

Required Default implementation provided.

Relationships

Inherits From

`Sendable`, `SendableMetatype`

See Also

Tool calling

- Expanding generation with tool calling

  Build tools that enable the model to perform tasks that are specific to your use case.

- Generate dynamic game content with guided generation and tools

  Make gameplay more lively with AI generated dialog and encounters personalized to the player.