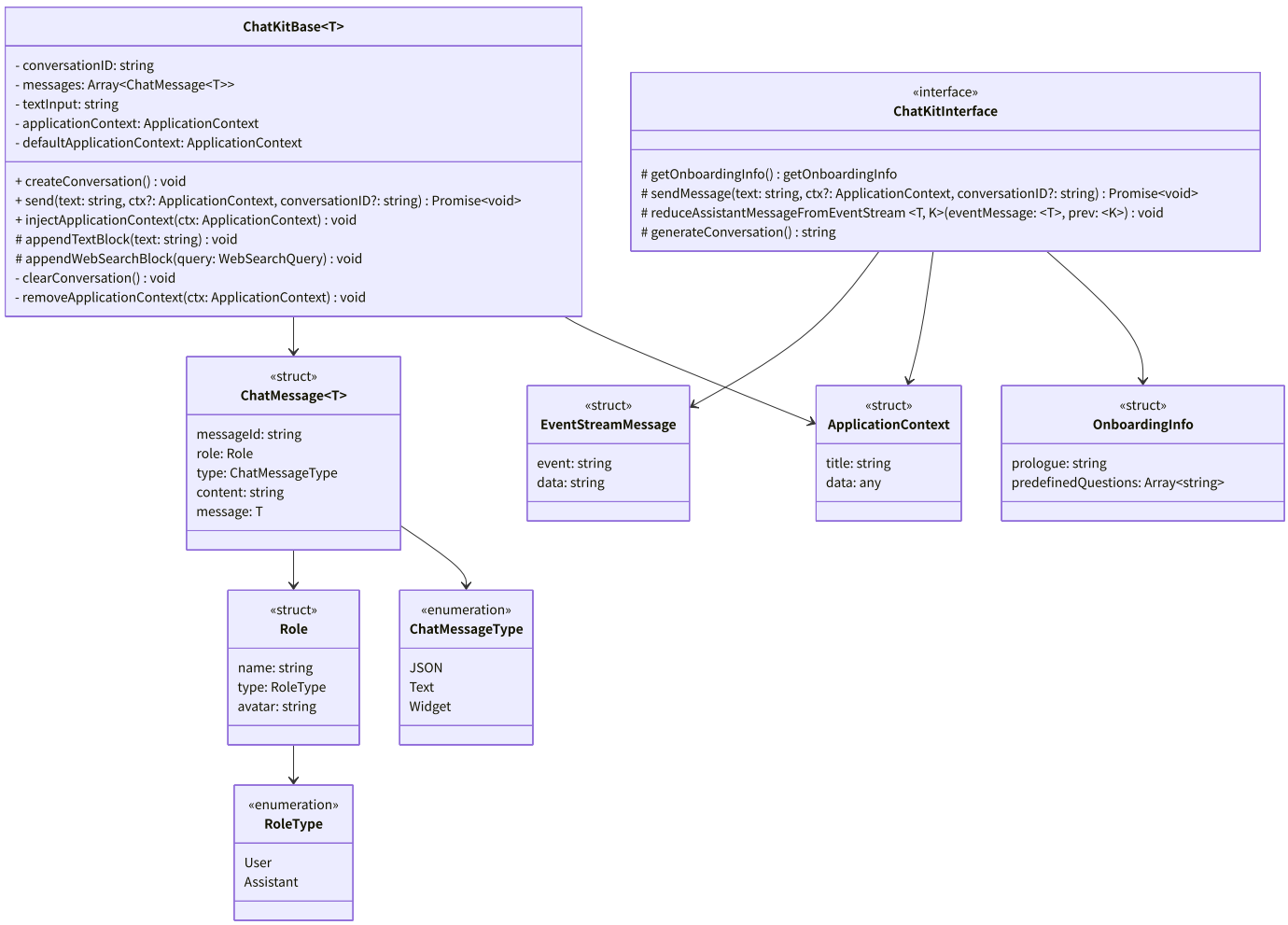
ChatKit

一、类和接口设计



1.1 class ChatKitBase<T>

ChatKitBase 是 AI 对话组件的核心类。该类是一个 React 组件，包含标准的交互界面和交互逻辑。

注意：开发者不能够直接挂载 `ChatKitBase` 到 Web 应用，而是需要创建一个子类继承 `ChatKitBase` 并实现 *`ChatKitInterface`* 中定义的方法。例如：实现一个 `ChatKitCoze` 子类，该子类实现调用 Coze 平台的智能体 API 并将 SSE 数据流转换为文本输出，开发者可以将 `ChatKitCoze` 挂载到 Web 应用中。

属性

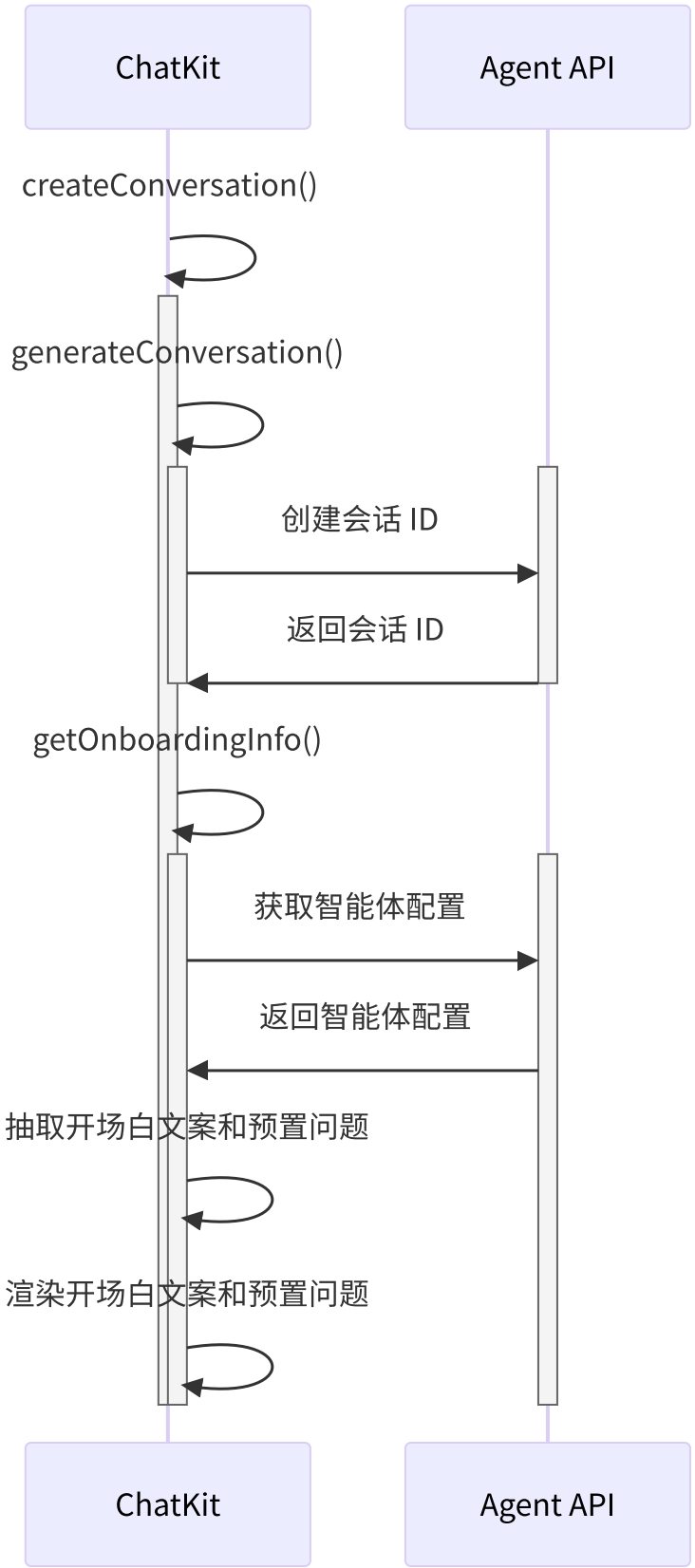
属性名	类型	说明
`conversationID`	`string`	会话 ID，每次新建会话时由后端返回新的会话唯一标识。在发送对话消息时，会将 `conversationID` 作为参数传入 `sendMessage()` 方法。
`messages`	`Array<ChatMessage<T>>`	消息列表。 <T>: 表示原始的消息结构
`textInput`	`string`	用户输入的文本。
`applicationContext`	`ApplicationContext`	和用户输入文本相关的上下文。
`defaultApplicationContext`	`ApplicationContext`	当没有指定 `applicationContext` 时的默认应上下文。
`token`	`string`	调用接口时携带的令牌，放置到请求头： `Authorization:Bearer {token}`
`refreshToken`	`() -> Promise<string>`	刷新 token 的方法，由集成方传入。

方法

`public createConversation(): void`

创建新的会话。`createConversation()` 方法内部会调用子类实现的 `generateConversation()` 方法。

视图

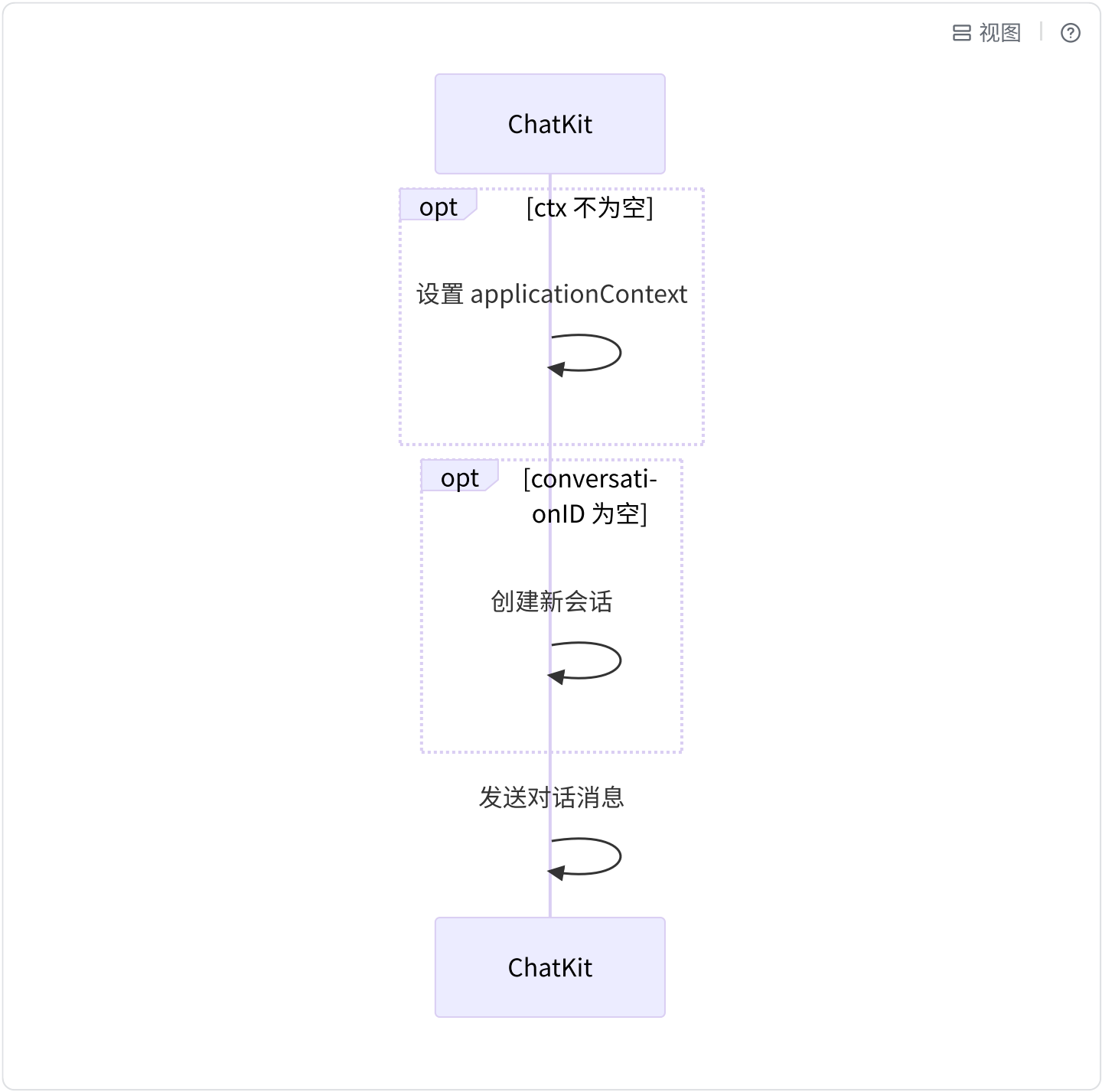



```
public send(text: string, ctx?: ApplicationContext, conversationID?: string): Promise<void>
```

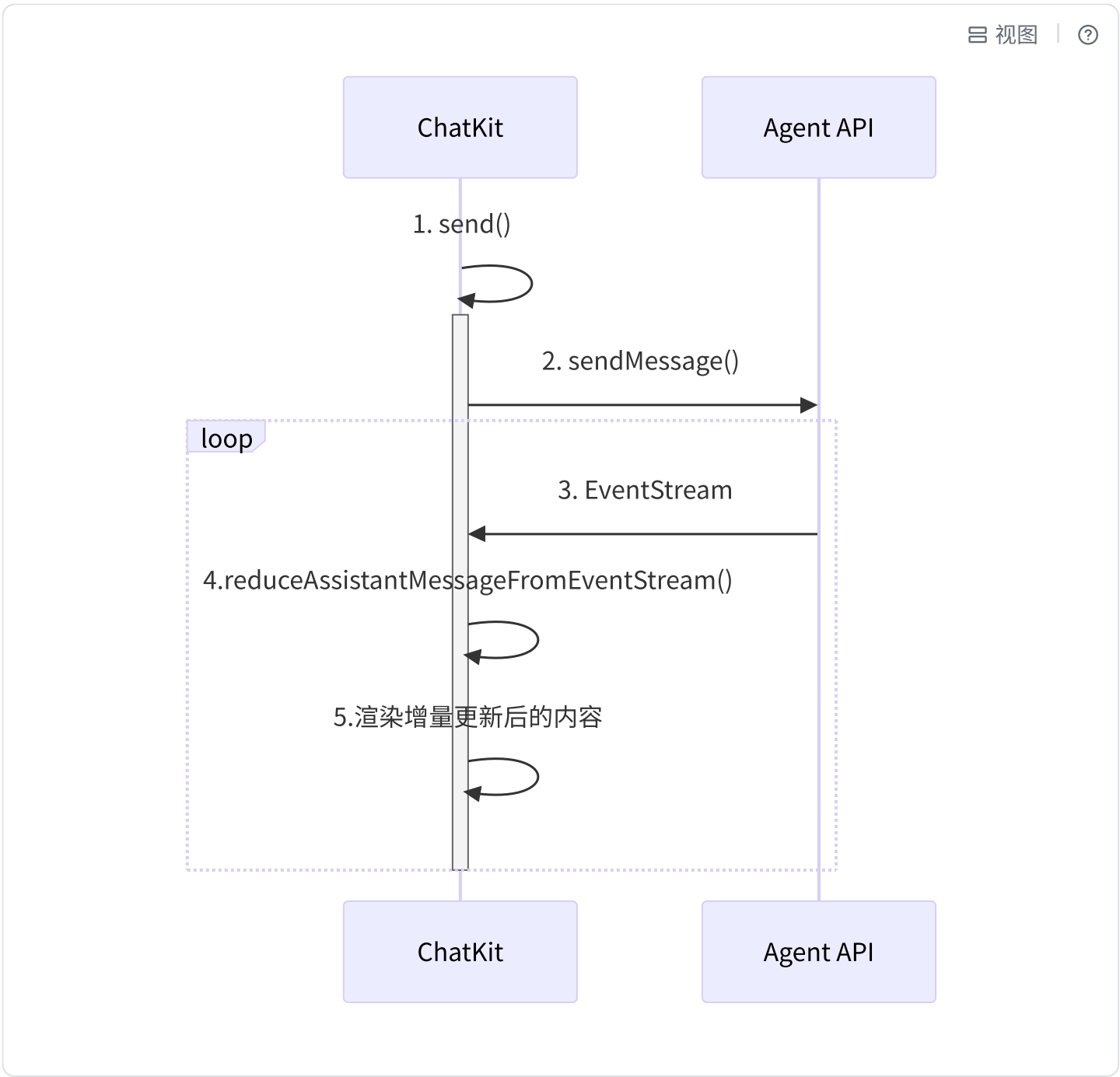
发送消息。该方法是暴露给集成方进行调用的接口，`send()` 方法内部会调用子类实现的 `sendMessage()` 方法。

参数名	参数类型	说明
`text`	`string`	发送给后端的用户输入的文本。
`ctx`	`ApplicationContext`	随用户输入文本一起发送的应用上下文。
`conversationID`	`string`	发送的对话消息所属的会话 ID。

下图是调用 send() 方法时对应用上下文和会话的处理：



下图是 ChatKit 从 Agent API 接收处理 EventStream 数据流的流程：



ChatKit 向 Agent 发起对话的消息处理流程：

1. ChatKit 调用 `send()` 方法发起处理流程。
2. `send()` 方法内部调用 `sendMessage()` 向 Agent API 服务发起对话请求（SSE）。
3. Agent API 服务持续输出 EventStream。
4. 调用 `reduceAssistantMessageFromEventStream()` ，解析 Event Message 并增量更新到 `messages[].message` 对象。
5. 从 `messages[].message` 中提取需要渲染到界面的元素。这一步由子类实现抽取逻辑，再调用 `ChatKitBase` 提供的渲染方法。

```
public injectApplicationContext(ctx: ApplicationContext): void
```

向 ChatKit 注入应用上下文。

参数名	参数类型	说明
`ctx`	`ApplicationContext`	要注入的应用上下文

`private clearConversation(): void`

清除会话中的对话消息及会话ID。

`private removeApplicationContext(): void`

移除注入的应用上下文。

`protected appendTextBlock(text: string): void`

使用 Markdown 渲染 AI 助手返回的文本内容。该方法由子类调用。

参数名	参数类型	说明
`text`	`string`	要渲染到界面的文本，每次都传完整的文本。

`protected appendWebSearchBlock(query: WebSearchQuery): void`

渲染 AI 助手执行 Web 搜索的执行详情。该方法由子类调用。

参数名	参数类型	说明
`query`	`WebSearchQuery`	Web 搜索的执行详情

1.2 interface ChatKitInterface

该接口定义了 ChatKit 的一些抽象方法。

方法

`protectedd getOnboardingInfo(): OnboardingInfo`

获取开场白和预置问题。该方法需要由子类继承并重写，以适配扣子、Dify 等 LLMOps 平台的接口。返回开场白信息结构体。

注意：该方法是一个无状态无副作用的函数，不允许修改 state。

`protected generateConversation(): string`

新建会话。该方法需要由子类继承并重写，以适配扣子、Dify 等 LLMOps 平台的接口。成功返回会话 ID。

注意：该方法是一个无状态无副作用的函数，不允许修改 state。

`protected sendMessage(text: string, ctx?: ApplicationContext, conversationID?: string): Promise<ChatMessage>`

向后端发送消息。该方法需要由子类继承并重写，以适配扣子、Dify 等 LLMOps 平台的接口。发送成功后，返回发送的消息结构。

注意：该方法是一个无状态无副作用的函数，不允许修改 state。

参数名	参数类型	说明
`text`	`string`	发送给后端的用户输入的文本。
`ctx`	`ApplicationContext`	随用户输入文本一起发送的应用上下文。
`conversationID`	`string`	发送的对话消息所属的会话 ID。

`protected reduceAssistantMessageFromEventStream(eventMessage: T, prev: K): void`

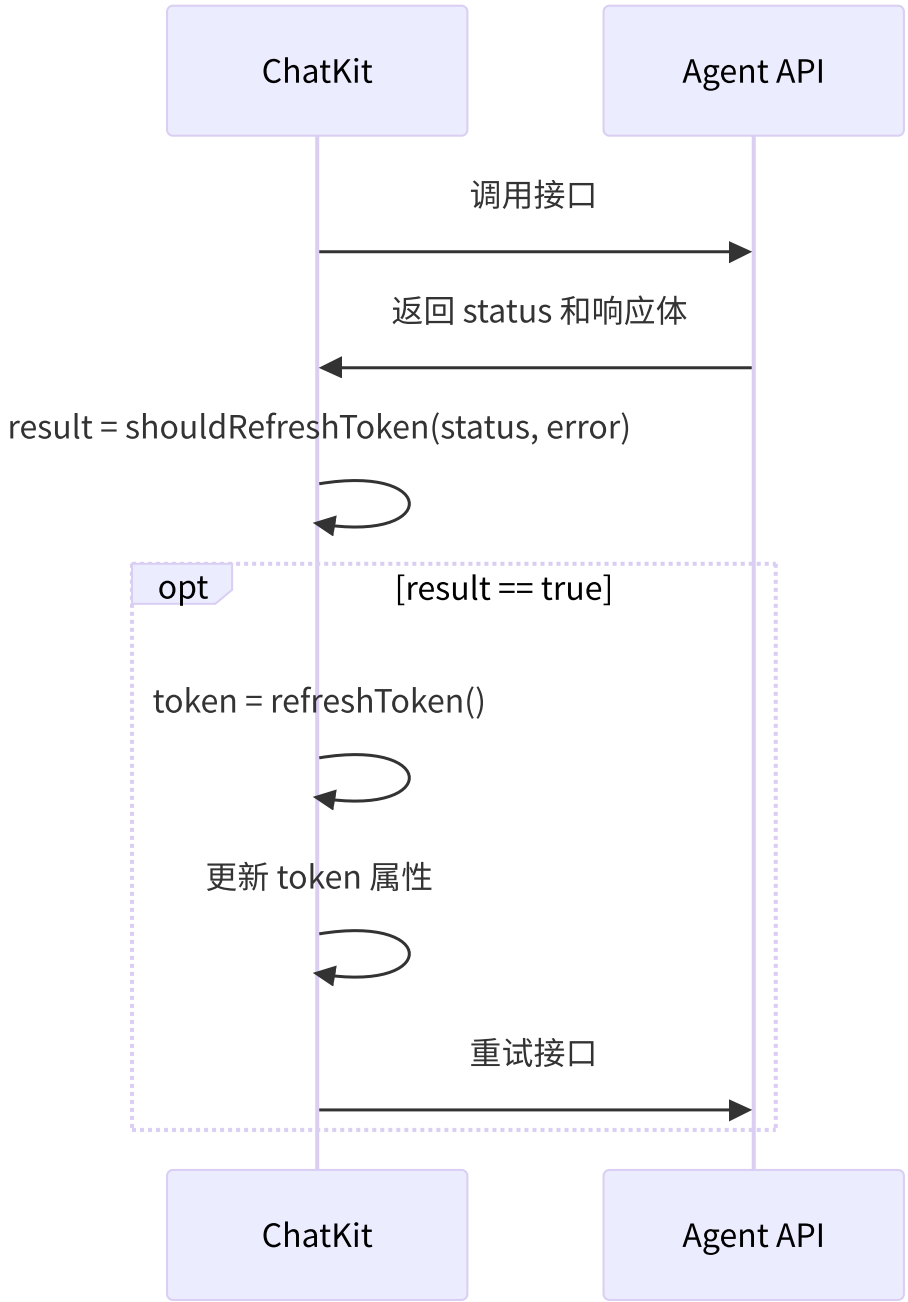
将 API 接口返回的 EventStream 增量解析成完整的 AssistantMessage 对象。该方法需要由子类实现。

注意：该方法是一个无状态无副作用的函数，不允许修改 state。

参数名	参数类型	说明
`eventMessage`	`T`	接收到的一条 Event Message
`prev`	`K`	上一次增量更新后的 AssistantMessage 对象

`protected shouldRefreshToken(status: number, error: object): boolean`

当发生异常时检查是否需要刷新 token。返回 `true` 表示需要刷新 token，返回 `false` 表示无需刷新 token。该方法需要由子类继承并重写，以适配扣子、Dify 等 LLMOps 平台的接口。



注意：重试接口后再次检查响应，如果仍然提示 token 失效，则放弃重新获取 token。

1.3 type WebSearchQuery

调用 Web 搜索的详情

属性

属性名	类型	说明
`input`	`string`	预置问题

`results`	`Array<WebSearchResult>`	Web 搜索结果集合

1.4 type WebSearchResult

单条 Web 搜索的结果

属性

属性名	类型	说明
`content`	`string`	搜索结果的内容摘要
`icon`	string	搜索结果的来源网站图标 URL
`link`	string	搜索结果的来源地址
`media`	string	搜索结果的来源网站名称
`title`	string	搜索结果的来源文章标题

1.5 type OnboardingInfo

开场白信息，包含开场白文案和预置问题。

属性

属性名	类型	说明
`prologue`	`string`	开场白文案
`predefinedQuestions`	`Array<string>`	预置问题

1.6 type ApplicationContext

与用户输入的文本相关的应用上下文。

属性

属性名	类型	说明

`title`	`string`	显示在输入框上方的应用上下文标题。
`data`	`any`	该应用上下文实际包含的数据。

1.7 type ChatMessage<T>

展示在消息区消息列表中的一条消息。

属性

属性名	类型	说明
`messageId`	`string`	一条消息的 ID。
`role`	`Role`	发送该消息的角色。
`type`	`ChatMessageType`	该条消息的类型。
`content`	`string`	该条消息的内容。
`raw`	`T`	原始的消息结构。

1.8 type Role

属性

属性名	类型	说明
`name`	`string`	角色的名称： • 如果 `type` 是 `Assistant`，则名称为 “AI 助手” • 如果 `type` 是 `User`，则名称为用户的昵称/显示名
`type`	`RoleType`	发送该消息的角色
`avatar`	`string`	角色的头像，可以是 URL、Base64 或 SVG。

1.9 type EventStreamMessage

从 SSE 接收到的 EventStream 消息。

属性

属性名	类型	说明
`event`	`string`	EventStream 的事件类型。
`data`	`string`	EventStream 的事件数据。

1.10 enum ChatMessageType

消息的类型。

值

值	说明
`Text`	Markdown 文本类型
`JSON`	JSON 类型
`Widget`	Widget 组件

1.11 enum RoleType

发送该消息的角色。

值

值	说明
`User`	用户
`Assistant`	AI 助手