**Chatbot API Documentation**

**Overview**

This API documentation describes the endpoints and data flow for a chatbot system that integrates Botkit, a Widget (UI), MongoDB, external APIs, WebSocket, Faye.js, Redux, and various authentication mechanisms including API keys, OAuth tokens, and username/password authentication.

**Authentication Mechanisms**

**1. API Key Authentication**

* **Description**: Used for server-to-server communication or for accessing public APIs without user context.
* **Usage**: Include the API key in the request headers.

**2. OAuth 2.0 Token Authentication**

* **Description**: Used for secure user authentication and authorization. Tokens are generated via OAuth 2.0 flow.
* **Usage**: Include the Bearer token in the request headers.

**3. Username/Password Authentication**

* **Description**: Basic authentication method for user login. Used to obtain an OAuth token.
* **Usage**: Credentials are sent via a secure POST request to obtain an OAuth token.

**User Authentication Endpoints**

**Endpoint: /auth/login**

* **Method**: POST
* **Description**: Authenticates a user using a username and password, returning an OAuth token for subsequent requests.
* **Request Body**:

json

Copy code

{

"username": "string",

"password": "string"

}

* **Response**:
  + **200 OK**:

json

Copy code

{

"access\_token": "oauth-token-string",

"expires\_in": 3600,

"token\_type": "Bearer"

}

* + **401 Unauthorized**: Invalid credentials.

**Endpoint: /auth/refresh**

* **Method**: POST
* **Description**: Refreshes an expired OAuth token.
* **Request Body**:

json

Copy code

{

"refresh\_token": "refresh-token-string"

}

* **Response**:
  + **200 OK**:

json

Copy code

{

"access\_token": "new-oauth-token-string",

"expires\_in": 3600,

"token\_type": "Bearer"

}

* + **401 Unauthorized**: Invalid refresh token.

**Botkit Interaction**

**Endpoint: /bot/message**

* **Method**: POST
* **Description**: Sends a user message to Botkit for processing and receives a response.
* **Authentication**:
  + **API Key**: X-API-Key: your-api-key
  + **OAuth Token**: Authorization: Bearer oauth-token-string
* **Request Body**:

json

Copy code

{

"message": "string",

"userId": "string"

}

* **Response**:
  + **200 OK**:

json

Copy code

{

"response": "string",

"context": {

"conversationId": "string",

"data": {}

}

}

* + **401 Unauthorized**: Invalid or missing token.
  + **403 Forbidden**: API key or OAuth token does not have permission.

**Real-time Communication**

**WebSocket: ws://yourserver.com/socket**

* **Description**: Establishes a WebSocket connection for real-time communication between the Widget and Botkit.
* **Authentication**:
  + Send the OAuth token as part of the connection request:
    - Authorization: Bearer oauth-token-string
* **Data Format**:
  + **Outgoing Message**:

json

Copy code

{

"token": "oauth-token-string",

"message": "string",

"userId": "string"

}

* + **Incoming Message**:

json

Copy code

{

"response": "string",

"context": {

"conversationId": "string",

"data": {}

}

}

**Faye.js: Channel /bot/interaction**

* **Description**: Uses Faye.js for publish-subscribe messaging to manage real-time interactions between the Widget and Botkit.
* **Authentication**:
  + Send the OAuth token as part of the subscription request:
    - Authorization: Bearer oauth-token-string
* **Data Format**:
  + **Publish** (from Widget to Botkit):

json

Copy code

{

"token": "oauth-token-string",

"message": "string",

"userId": "string"

}

* + **Subscribe** (from Botkit to Widget):

json

Copy code

{

"response": "string",

"context": {

"conversationId": "string",

"data": {}

}

}

**State Management (Redux Integration)**

**State Initialization**

* **Description**: Redux initializes the application state when the Widget is loaded.
* **State Structure**:

json

Copy code

{

"user": {

"userId": "string",

"authenticated": false,

"token": null

},

"conversation": {

"conversationId": "string",

"messages": []

}

}

**Actions**

**Action: USER\_LOGIN**

* **Description**: Triggers user login and updates the authentication state.
* **Payload**:

json

Copy code

{

"userId": "string",

"token": "oauth-token-string"

}

**Action: NEW\_MESSAGE**

* **Description**: Dispatches when a new message is sent by the user or received from Botkit.
* **Payload**:

json

Copy code

{

"message": "string",

"sender": "user|bot"

}

**Action: UPDATE\_CONVERSATION**

* **Description**: Updates the conversation state with the latest context and messages.
* **Payload**:

json

Copy code

{

"conversationId": "string",

"messages": []

}

**Data Storage**

**Endpoint: /conversation/history**

* **Method**: GET
* **Description**: Retrieves the conversation history for a specific user.
* **Authentication**:
  + **API Key**: X-API-Key: your-api-key
  + **OAuth Token**: Authorization: Bearer oauth-token-string
* **Request Query**:
  + userId: string
* **Response**:
  + **200 OK**:

json

Copy code

{

"conversationId": "string",

"messages": []

}

* + **404 Not Found**: No conversation history found.

**Endpoint: /conversation/save**

* **Method**: POST
* **Description**: Saves the conversation data to MongoDB.
* **Authentication**:
  + **API Key**: X-API-Key: your-api-key
  + **OAuth Token**: Authorization: Bearer oauth-token-string
* **Request Body**:

json

Copy code

{

"conversationId": "string",

"userId": "string",

"messages": []

}

* **Response**:
  + **200 OK**: Conversation saved successfully.
  + **500 Internal Server Error**: Failed to save the conversation.

**External API Interaction**

**Endpoint: /api/weather**

* **Method**: GET
* **Description**: Fetches weather data for a given location.
* **Authentication**:
  + **API Key**: X-API-Key: your-api-key
  + **OAuth Token**: Authorization: Bearer oauth-token-string
* **Request Query**:
  + location: string
* **Response**:
  + **200 OK**:

json

Copy code

{

"location": "string",

"temperature": "number",

"conditions": "string"

}

* + **404 Not Found**: Location not found.

**Endpoint: /api/user-info**

* **Method**: GET
* **Description**: Retrieves user-specific data from an external service.
* **Authentication**:
  + **API Key**: X-API-Key: your-api-key
  + **OAuth Token**: Authorization: Bearer oauth-token-string
* **Request Query**:
  + userId: string
* **Response**:
  + **200 OK**:

json

Copy code

{

"userId": "string",

"name": "string",

"preferences": {}

}

* + **404 Not Found**: User not found.

**Error Handling**

**Standard Error Responses**

* **401 Unauthorized**: Invalid or missing authentication token or API key.
* **403 Forbidden**: User or API key does not have permission to access the resource.
* **404 Not Found**: The requested resource does not exist.
* **500 Internal Server Error**: An unexpected error occurred on the server.

**Changelog**

* **Version 1.0.0**: Initial release of the Chatbot API documentation.

This API documentation outlines how to integrate multiple authentication mechanisms, providing flexibility for different use cases. Developers can choose the appropriate authentication method depending on the context (e.g., server-to-server, user authentication, etc.).