## Documentation for HassFramework

### Overview

The `HassFramework` is a framework developed for SwiftUI applications to interface with Home Assistant. It primarily provides a mechanism for establishing WebSocket connections to a Home Assistant server, and handling the various events and messages that may come from it.

#### Classes and Protocols Overview

1. `HassWebSocket`: Manages the WebSocket connection, handles authentication, and manages messages.

2. `WebSocketManager`: Observes and manages the WebSocket connection's state.

3. `HassModels`: Contains data structures for decoding JSON messages from the WebSocket.

4. `HassProtocols`: Declares protocols related to event message handling and WebSocket management.

5. `HassMessageTypes`: Encapsulates the general structure of incoming messages and provides a mechanism for decoding dynamic JSON.

6. `HassEnums`: Contains enumerations for connection states and error handling.

### HassWebSocket

This class manages the WebSocket connection with the Home Assistant server, handles authentication, message sending, and event subscription.

#### Properties:

- `connectionState`: Reflects the current connection state (connecting, connected, disconnected).

- `messageId`: Used for generating unique message IDs.

- `isAuthenticated`: Indicates if the WebSocket connection is authenticated.

- `pingTimer`: Timer for sending pings.

- `eventMessageHandlers`: Array of objects that conform to `EventMessageHandler` to handle event messages.

#### Methods:

- `connect(completion:)`: Initiates a connection to the WebSocket.

- `disconnect()`: Disconnects from the WebSocket.

- `authenticate()`: Sends an authentication message to the WebSocket.

- `subscribeToEvents()`: Subscribes to "state\_changed" events.

- `sendTextMessage(\_:)`: Sends a text message over the WebSocket.

- `handleEventMessage(\_:)`: Forwards the received event message to all registered `EventMessageHandler` objects.

- `determineWebSocketMessageType(data:)`: Determines the type of the received WebSocket message.

- `setDelegate(\_:)`: Sets the delegate for receiving raw WebSocket events.

- `isConnected()`: Checks if the WebSocket is currently connected.

#### Delegate: `HassWebSocketDelegate`

Allows the `HassWebSocket` to notify another object of raw WebSocket events.

### WebSocketManager

Observes the `HassWebSocket` instance, handles received WebSocket events, and translates them into appropriate actions.

#### Properties:

- `websocket`: The `HassWebSocket` instance this manager is observing.

#### Methods:

- `connectIfNeeded()`: Connects to the WebSocket if it's currently disconnected.

- `disconnectIfNeeded()`: Disconnects from the WebSocket if it's currently connected.

#### Delegate: `HassWebSocketDelegate`

Allows the `WebSocketManager` to be notified of raw WebSocket events.

### HassModels

Contains structures to decode JSON messages from the WebSocket.

#### Data Structures:

- `WebSocketMessageType`: Enum representing different WebSocket message types.

- `HAContext`: Data structure for the context part of an event message.

- `HAAttributes`: Data structure for the attributes of an entity's state.

- `HAState`: Represents the state of a Home Assistant entity.

- `HAEventData`: Represents an event message from the WebSocket.

### HassProtocols

Defines protocols for event message handling and WebSocket management.

#### Protocols:

- `EventMessageHandler`: Requires implementing a method to handle event messages.

- `WebSocketManagerDelegate`: For objects interested in raw WebSocket events and messages.

- `WebSocketProvider`: Specifies a set of methods and properties that a WebSocket provider must have.

### HassMessageTypes

Data structures that encapsulate the general structure of incoming WebSocket messages.

#### Data Structures:

- `MessageType`: Enum representing different types of messages.

- `HAMessage`: Encapsulates the general structure of incoming messages.

- `AnyCodable`: Utility to decode dynamic JSON structures.

### HassEnums

Enumerations for different states of the WebSocket connection and potential errors.

#### Enums:

- `ConnectionState`: Represents the connection state of the WebSocket.

- `HAError`: Enum representing potential errors within `HassFramework`.

### How to use HassFramework:

1. Initialize a `HassWebSocket` instance.

2. Set up a `WebSocketManager` to manage and observe the WebSocket.

3. Interact with the `HassWebSocket` instance to connect, authenticate, send messages, and subscribe to events.

4. Implement and register any custom `EventMessageHandler` objects if you need custom handling for event messages.

\*\*Note\*\*: Ensure you've set up the `Secrets.plist` file in your app's bundle with the correct `HomeAssistantServerURL` and `HomeAssistantAccessToken` for connecting to your Home Assistant server.