Support & Order Chat - Socket.io Events Documentation

Overview

The backend uses **Socket.io** with a `/support` namespace to handle real-time messaging for both **support chat** and **order chat**. This document describes all WebSocket events available for the Flutter restaurant/market app.

Connection Setup

Server URL

```
wss://avvento-server.onrender.com/support
```

Authentication

Pass the JWT token during connection via the `auth` object:

```
import 'package:socket_io_client/socket_io_client.dart' as IO;

final socket = IO.io(
  'https://avvento-server.onrender.com/support',
  IO.OptionBuilder()
      .setTransports(['websocket', 'polling'])
      .setAuth({'token': 'YOUR_JWT_TOKEN'})
      .enableReconnection()
      .setReconnectionDelay(1000)
      .setReconnectionDelayMax(5000)
      .setReconnectionAttempts(5)
      .build(),
);
```

Authentication Methods (any one)

Method	Example
`auth` object (recommended)	`{ token: "eyJhbGciOi..." }`
Query parameter	`?token=eyJhbGciOi...`
Header	`Authorization: Bearer eyJhbGciOi...`

On Successful Connection

- User is automatically joined to a personal room: `user:{userId}`
- This room is used for receiving `new-conversation` events without joining any specific conversation

On Failed Connection

- Socket is disconnected by the server
- Reasons: missing token, invalid/expired token, no userId in token payload

---

Events

Client → Server Events

1. `join-conversation`

Join a conversation room to start receiving real-time messages and updates for that conversation.

When to emit: When the user opens a chat screen (support chat or order chat).

Payload:

```
{
  "conversationId": "6650abc123def456"
}
```

Field	Type	Required	Description
`conversationId`	`String`	Yes	The MongoDB `_id` of the conversation

Server Response:

```
{
  "success": true,
  "conversationId": "6650abc123def456"
}
```

Error Cases:

- User is not authenticated → `UnauthorizedException`
- User does not have access to this conversation → `ForbiddenException`
- Conversation does not exist → `NotFoundException`

Flutter Example:

```
socket.emit('join-conversation', {'conversationId': conversationId});
```

---

2. `leave-conversation`

Leave a conversation room to stop receiving real-time messages and updates.

When to emit: When the user navigates away from a chat screen.

Payload:

```
{
  "conversationId": "6650abc123def456"
}
```

Field	Type	Required	Description
`conversationId`	`String`	Yes	The MongoDB `_id` of the conversation

Server Response:

```
{
  "success": true,
  "conversationId": "6650abc123def456"
}
```

Flutter Example:

```
socket.emit('leave-conversation', {'conversationId': conversationId});
```

Server → Client Events

3. `new-message`

Received when a new message is sent in a conversation that the user has joined.

When received: After a participant sends a message via `POST /support/messages`.

Prerequisite: Must have joined the conversation room via `join-conversation`.

Payload:

```json
{
  "_id": "665abc123def456789",
  "conversation": "665def456abc789012",
  "sender": {
    "_id": "664ghi789abc012345",
    "name": "أحمد محمد",
    "role": "user"
  },
  "type": "text",
  "content": "مرحبا، أحتاج مساعدة في طلبي",
  "isRead": false,
  "createdAt": "2025-02-14T10:30:00.000Z",
  "updatedAt": "2025-02-14T10:30:00.000Z"
}
```

Field	Type	Description
`_id`	`String`	Message unique ID
`conversation`	`String`	Conversation ID this message belongs to
`sender`	`Object`	Sender info
`sender._id`	`String`	Sender user ID
`sender.name`	`String`	Sender display name
`sender.role`	`String`	Sender role: `user`, `admin`, `restaurant`, `market`, `pharmacy`
`type`	`String`	Message type: `text`, `image`, `system`
`content`	`String`	Message content (text or image URL)
`isRead`	`Boolean`	Whether the message has been read
`createdAt`	`String`	ISO 8601 timestamp
`updatedAt`	`String`	ISO 8601 timestamp

Flutter Example:

```dart
socket.on('new-message', (data) {
  final message = Map<String, dynamic>.from(data);
  final content = message['content'];
  final senderName = message['sender']['name'];
  final type = message['type']; // text, image, system

  // Add message to chat list
  messages.add(message);
});
```

---

4. `conversation-updated`

Received when a conversation's status or metadata changes.

When received:

- Admin closes or reopens a conversation via `PATCH /support/conversations/:id`
- Order chat is auto-closed when the order is delivered or cancelled

- Conversation subject or metadata is updated

Prerequisite: Must have joined the conversation room via `join-conversation`.

Payload:

```json
{
  "_id": "665def456abc789012",
  "participants": [
    {
      "_id": "664ghi789abc012345",
      "name": "أحمد محمد",
      "role": "user"
    },
    {
      "_id": "664jkl012def345678",
      "name": "المدير",
      "role": "admin"
    }
  ],
  "subject": "مشكلة في الطلب #1234",
  "status": "closed",
  "supportType": "order",
  "relatedOrder": "665mno345ghi678901",
  "lastMessage": {
    "_id": "665pqr678jkl901234",
    "content": "تم إغلاق المحادثة",
    "type": "system",
    "createdAt": "2025-02-14T11:00:00.000Z"
  },
  "unreadCount": 0,
  "createdAt": "2025-02-14T10:00:00.000Z",
  "updatedAt": "2025-02-14T11:00:00.000Z"
}
```

Field	Type	Description
`_id`	`String`	Conversation unique ID
`participants`	`Array`	List of participant objects with `_id`, `name`, `role`
`subject`	`String`	Conversation subject/title
`status`	`String`	Current status: `open`, `closed`, `resolved`
`supportType`	`String`	Type: `general` (support chat) or `order` (order chat)
`relatedOrder`	`String?`	Order ID (only for `supportType: "order"`)
`lastMessage`	`Object?`	Last message in the conversation
`unreadCount`	`Number`	Number of unread messages
`createdAt`	`String`	ISO 8601 timestamp
`updatedAt`	`String`	ISO 8601 timestamp

Flutter Example:

```
socket.on('conversation-updated', (data) {
  final conversation = Map<String, dynamic>.from(data);
  final status = conversation['status'];

  if (status == 'closed') {
    // Show "conversation closed" banner
    // Disable message input
  } else if (status == 'open') {
    // Re-enable message input
  }
});
```

---

5. `new-conversation`

Received when a new conversation is created that involves this user.

When received:

- Admin creates a support chat targeting this user
- A user opens an order chat (admin receives this event)
- Any new conversation is created where this user is a participant

Prerequisite: Only requires socket connection. **No need to join any room** — this event is sent to the user's personal room `user:{userId}` which is auto-joined on connection.

Payload:

```
{
  "_id": "665pqr678jkl901234",
  "participants": [
    {
      "_id": "664ghi789abc012345",
      "name": "أحمد محمد",
      "role": "user"
    },
    {
      "_id": "664jkl012def345678",
      "name": "المدير",
      "role": "admin"
    }
  ],
  "subject": "طلب #1234",
  "status": "open",
  "supportType": "order",
  "relatedOrder": "665mno345ghi678901",
  "createdAt": "2025-02-14T12:00:00.000Z",
  "updatedAt": "2025-02-14T12:00:00.000Z"
}
```

Field	Type	Description
`_id`	`String`	New conversation unique ID
`participants`	`Array`	Participants with `_id`, `name`, `role`
`subject`	`String`	Conversation subject
`status`	`String`	Always `open` for new conversations
`supportType`	`String`	`general` or `order`
`relatedOrder`	`String?`	Order ID (only for order chats)
`createdAt`	`String`	ISO 8601 timestamp
`updatedAt`	`String`	ISO 8601 timestamp

Field	Type	Description

Flutter Example:

```
socket.on('new-conversation', (data) {
  final conversation = Map<String, dynamic>.from(data);

  // Show notification badge on chat icon
  // Add to conversations list
  // Optionally show a snackbar notification
});
```

REST API Endpoints (Used with Socket Events)

Support Chat

Method	Endpoint	Roles	Description
`POST`	`/support/conversations`	Admin, Restaurant	Create a new support conversation
`POST`	`/support/conversations/with-admin`	User, Restaurant, Market, Pharmacy	Create support conversation with admin
`GET`	`/support/conversations`	All authenticated	Get my conversations
`GET`	`/support/conversations/:id`	Participants	Get a specific conversation
`PATCH`	`/support/conversations/:id`	Admin	Update conversation (status, subject)
`POST`	`/support/messages`	Participants	Send a message
`GET`	`/support/conversations/:id/messages`	Participants	Get messages (paginated)
`POST`	`/support/conversations/:id/read`	Participants	Mark messages as read

Order Chat

Method	Endpoint	Roles	Description
`POST`	`/support/order-chat/:orderId`	All authenticated	Create or get order chat
`GET`	`/support/order-chat/:orderId`	Participants	Get order chat conversation
`GET`	`/support/order-chat/:orderId/messages`	Participants	Get order chat messages (paginated)

Query Parameters for Messages

Parameter	Type	Default	Description
`page`	`Number`	`1`	Page number
`limit`	`Number`	`50`	Messages per page

Connection Lifecycle

```
┌─────────────────────────────────────────────────────────┐
│                      APP START                          │
│                          │                              │
│                    connect(token)                        │
│                          │                              │
│              Auto-joins user:{userId} room              │
│              Starts receiving: new-conversation          │
│                          │                              │
├─────────────────────────────────────────────────────────┤
│                    OPEN CHAT SCREEN                      │
│                          │                              │
│              joinConversation(conversationId)            │
│                          │                              │
│              Starts receiving:                           │
│                - new-message                             │
│                - conversation-updated                    │
│                          │                              │
├─────────────────────────────────────────────────────────┤
│                   CLOSE CHAT SCREEN                      │
│                          │                              │
│              leaveConversation(conversationId)           │
│                          │                              │
│              Stops receiving:                            │
│                - new-message                             │
│                - conversation-updated                    │
│                          │                              │
├─────────────────────────────────────────────────────────┤
│                       LOGOUT                            │
│                          │                              │
│                    disconnect()                          │
│                          │                              │
│              All listeners removed                       │
│              Socket connection closed                    │
└─────────────────────────────────────────────────────────┘
```

Order Chat Flow (Step by Step)

```
1. User opens order chat
   └─ POST /support/order-chat/{orderId}
   └─ Returns conversation object (creates if not exists)
   └─ Admin receives "new-conversation" via socket

2. User joins conversation room
   └─ emit('join-conversation', { conversationId })

3. Admin joins conversation room
   └─ emit('join-conversation', { conversationId })

4. User sends a message
   └─ POST /support/messages { conversationId, content }
   └─ Both receive "new-message" via socket

5. Admin replies
   └─ POST /support/messages { conversationId, content }
   └─ Both receive "new-message" via socket

6. Order is delivered or cancelled
   └─ Server auto-closes the order chat
   └─ Both receive "conversation-updated" with status: "closed"
   └─ A system message "تم إغلاق محادثة الطلب" is sent
```

Flutter Package

Add to `pubspec.yaml`:

```
dependencies:
  socket_io_client: ^2.0.3+1
```

Install:

```
flutter pub get
```