Fluid Chat

Complete Implementation & Configuration Guide v8

This comprehensive guide details the configuration process for the Fluid Chat plugin across both Web and Mobile responsive platforms. Follow these schemas and workflow structures precisely to guarantee elite application performance and seamless S3 storage synchronization.

1. Database Architecture & Schema

To fully leverage all messaging capabilities within Fluid Chat (including reactions, typing indicators, and media sharing), create the following three specific data types within your Bubble database ecosystem.

DATA TYPE: CONVERSATION

- title (Type: text)
- members_number (Type: number) - Optimizes performance by removing the need to dynamically execute a count lookup on Conversation_Members continuously.

DATA TYPE: MESSAGES

Field Name	Field Type	Description/Scope
audio_duration	number	Stores recorded file length in seconds Mobile Only.
created_by	User	Standard Bubble message creator reference.
creation_date	date	Standard timestamp field.
conversation_id	Conversation	Relational reference mapping to the target Chat Conversation.
file_names_list	text (List)	Stores clean string names of uploaded files.
files_list	text (List)	Array storing file download URLs.
images_list	image (List)	Array storing absolute image URLs.
is_edited	yes/no	Flag checking if the message text was altered post-delivery.
is_system_message	yes/no	Flag used to style standard system event updates natively.
message	text	Main message text body.
reactions	text	Serialized string mapping active user reaction emojis.
read_by	User (List)	Array tracking which participants have read the message.
response_message_id	text	Stores target Message UniqueID when replying to an item.
is_poll	Yes/no (List)	Flag indicating if the message should be rendered as an interactive poll.
poll_data	Text (List)	Stores the poll options and user votes in a unified, serialized string format (e.g., Option A|user1,user2||Option B|user3).

DATA TYPE: CONVERSATION_MEMBERS

- conversation_id (Type: Conversation)
- member (Type: User)
- is_writing (Type: yes/no)
- is_recording (Type: yes/no)
- last_seen (Type: date) - [V7 Addition] Stores the exact timestamp of the user's last "heartbeat" to manage the real-time online presence status.

DATA TYPE: CONVERSATION_STARRED

- conversation_id (Type: Conversation)
- member (Type: User)
- starred_messages (Type: text (List))

2. Fluid Chat (Web Edition Configurations)

Map the properties in the Bubble UI Property Editor exactly as detailed in the technical tables below to properly bind your relational architecture to the messaging display layout engine.

PRESENCE & CHAT HEADER CONFIGURATIONS (NEW)

Plugin Editor Field	Target Property / Expression Mapping
Show Header	Boolean flag switching the visibility of the top chat header natively.
Header Name	Name of the active conversation user (e.g., Current Page User's Name).
Header Avatar	Avatar image of the active conversation user.
User Is Online	Boolean evaluating the heartbeat timestamp. Expression: Current User's last_seen > Current date/time + (seconds): -45.
Last Seen Text	Formatted text displaying the last active status. Expression: "last seen " & Current User's last_seen:formatted as 11:32 AM.

CORE DATA BINDING PROPERTIES

Plugin Editor Field	Target Property / Expression Mapping
Message Text List	Messages.message
Message Id List	Messages.UniqueID
Message Creator Id List	Messages.created_by.UniqueID
Message Creator Names List	Messages.created_by.Name
Timestamp	Messages.creation_date

USERS & MENTIONS CONFIGURATIONS

Plugin Editor Field	Target Property / Expression Mapping
Current User Id	Current User's Unique ID
Avatar List	Messages.created_by.Image (or Profile Picture field)
Mention Names List	Messages (grouped by created_by)'s created_by's Name
Mention Ids List	Messages (grouped by created_by)'s created_by's UniqueID
Mention Avatars List	Messages (grouped by created_by)'s created_by's Image

MEDIA, ENGAGEMENT & INDICATORS

Plugin Editor Field	Target Property / Expression Mapping
Images List	Messages (format as text) -> This Message's images_list (join with) Delimiter: ||
Files List	Messages (format as text) -> This Message's files_list (join with) Delimiter: ||
Files Names List	Messages (format as text) -> This Message's file_names_list (join with,) Delimiter: ||
Reactions String	Messages.reactions
Reply to ID List	Messages.response_message_id
Is Edited List	Messages.is_edited
Custom Menu Items	Configurable text array mapping long-press/right-click trigger menu structures.
Read By Count List	Messages.read_by (count)
Required Read Count	Integer checking target minimum count value before marking bubble as officially read.
Is System Message List	Messages.is_system_message
Typing Names List	Search for Conversation_Members (Constraints: conversation = Current Conversation, is_writing = yes, member <> Current User)'s Member's Name
Recording Names List	Search for Conversation_Members (Constraints: conversation = Current Conversation, is_recording = yes, member <> Current User)'s Member's Name

3. WEB WORKFLOWS IMPLEMENTATION

🫀 The "Heartbeat" Technique in Bubble

- 1. Database Setup:

In the **User** table, instead of creating a boolean *Online (yes/no)* field, create a field named **last_active** (Type: Date).

- 2. The Invisible Workflow (In your client's App):

On the page (or within a reusable Header), create a workflow:
Event: Do every 30 seconds
Action: Make changes to Current User -> last_active = Current Date/Time

(This is the beating "heart". As long as the tab remains open, Bubble updates this date every 30 seconds).

- 3. The Math Secret (How to feed the Plugin):

Now, in your Plugin, specifically in the user_is_online field we created, the developer won't pass a static value. They will pass a dynamic expression using date math:

Current User's last_active > Current date/time +(seconds): -45

What this does: Bubble evaluates "Was the last time this user showed signs of life less than 45 seconds ago?"
- If yes -> It passes 'yes' to our plugin (the green dot lights up).
- If the user closes the browser... the Heartbeat stops. After 45 seconds, the math condition fails, Bubble passes 'no' to the plugin, and the green dot disappears automatically. Zero friction, no closed-tab bugs!

- 🕒 What about the "Last Seen" text?

Using this exact same technique, populating the last_seen_text field in our plugin becomes incredibly easy using Bubble's native formatting. The developer just needs to insert this expression:

"last seen " & Current User's last_active:formatted as 11:32 AM

Our plugin handles the rest of the visual magic (hiding the timestamp and only displaying a green "Online" if the heartbeat is actively ticking).

Workflow: Sending Images, Files, and Audio via Feed Interaction

User executes standard web selector operations clicking media/document attachments.

User updates file items actively (can optionally dismiss or add further items inside draft stage).

User hits Send button, triggering the Message Sent element event.

Action Routine: Instantiate a Create a New Thing step target data type Messages mapping fields as follows:

created_by = Current User

creation_date = Current Date/Time

conversation_id = Current actively loaded Conversation

file_names_list = FluidChat's Uploaded File Name

files_list = FluidChat's Uploaded File List (stores non-image files and audio binaries)

images_list = FluidChat's Uploaded Images List

is_edited = no

is_system_message = no

message = FluidChat's Draft Text

read_by = Current User (add item to list)

response_message_id = FluidChat's Draft Reply To ID

Workflow: Message Reactions Engine

User executes a context-menu trigger (right-click on desktop or long-press on touch screens) on a bubble.

User clicks an emoji glyph from the popover, triggering the Reaction Updated event.

Action Routine: Trigger Make Changes to Thing pointing to the targeted Message item:

reactions = FluidChat's Updated Reactions String

Workflow: Processing User @Mentions

When the character string token @ is keyed into the draft input container, the companion context user array triggers open automatically.

Selecting specific users formats tokens inside the input string. Sending triggers User Mentioned.

Downstream Routine: Bind programmatic notification actions directly downstream of this trigger hook, referencing the automated array state string: FluidChat's Mentioned Users Ids.

Workflow: Delete and Edit Message Routines

Context menu item interaction fires the underlying structural event hook: Long Press Triggered.

Branch A (Edit Mode): Only when FluidChat's Last Action is edit:

> Action: Make Changes to Thing -> target Message: is_edited = yes, message = FluidChat's Draft Text.

Branch B (Delete Mode): Only when FluidChat's Last Action is delete:

> Action: Delete Thing -> target constraint message matching: Message's UniqueID = Selected Message ID.

Workflow: Real-Time Typing Notifications Broadcast

The Fluid Chat plugin automatically detects when the current user is typing on their keyboard or holding the microphone. It outputs this via two states: Is Typing and Is Recording. Set up the following workflows on your chat page:

Event: Do every time condition is met -> When FluidChat's Is Typing is yes:

> Action: Make Changes to Thing -> target Conversation_Members (Current User's row) -> set is_writing = yes.

Event: Do every time condition is met -> When FluidChat's Is Typing is no:

> Action: Make Changes to Thing -> target Conversation_Members (Current User's row) -> set is_writing = no.

(Repeat the exact same logic for the Recording status)

Event: Do every time condition is met -> When FluidChat's Is Recording is yes:

> Action: Make Changes to Thing -> target Conversation_Members (Current User's row) -> set is_recording = yes.

Event: Do every time condition is met -> When FluidChat's Is Recording is no:

Action: Make Changes to Thing -> target Conversation_Members (Current User's row) -> set is_recording = no.

3. Feeding the Plugin (The Input)

Now that your database knows exactly what everyone is doing, you just need to feed the names of the active users into the plugin's properties. The plugin will automatically do the math (e.g., changing "John is typing..." to "John and Mary are typing...") and prioritize the Recording Microphone icon over the Typing Dots.

Map the following properties in the Fluid Chat element inspector:

Typing Names List: Search for Conversation_Members (Constraints: conversation = Current Conversation, is_writing = yes, member <> Current User)'s Member's Name

Recording Names List: Search for Conversation_Members (Constraints: conversation = Current Conversation, is_recording = yes, member <> Current User)'s Member's Name

Note: The plugin automatically translates and builds the sentence based on the Localization fields provided in the element inspector (e.g., "is typing...", "is recording audio...").

Workflow: Advanced Message Info (Read Receipts Screen)

Design your "Message Details" UI using Bubble native groups or a new page (as seen in WhatsApp).

Add a workflow event: When Fluid Chat - Message Info Clicked.

Action 1: Set a state or navigate to your Details Page/Group, passing the Fluid Chat's Selected Message ID so the page knows which message to display at the top.

Action 2: Use a Repeating Group on that page. Set its Data Source to: Search for Messages (UniqueID = Fluid Chat's Selected Message ID):first item's read_by.

Display the users' avatars, names, and timestamps dynamically within the Repeating Group using Bubble's native dynamic data.

Workflow: Optimized Infinite Scroll Pagination (Lazy Loading Setup)

Static Legacy Routing: Simply pull query list using data binding constraint creation_date Descending Order = no.

Performant Lazy Loading Routing: Construct primary source data search parameters specifying: creation_date Descending Order = yes, appended with modifier constraint operator items until "message_limit", concluded with post-sort query operator constraint: sorted by creation_date Descending Order = no.

Design Note: Declare a page-level Number custom state named message_limit initializing with base configuration boundaries (e.g., 20, 30, or 40).

When scroll bounds hit upper limits, the element automatically fires event hook: Load More Triggered.

Action Routine: Trigger Set State mapping parameter: message_limit = parent_container's message_limit + X (X represents your matching incremental load stepping window constant).

Workflow: Starring & Unstarring Messages

Trigger: User executes a context-menu trigger and taps the Star icon, firing the Star Pressed element event.

Action Routine: Trigger Make Changes to Thing (or Create a New Thing if the user's Starred entity for this chat doesn't exist yet) targeting the **STARRED** data type.

Logic:

- If Current User's Starred's message_ids_list *contains* Fluid Chat's selected_message_id: **Remove** Fluid Chat's selected_message_id.
- If it *does not contain* it: **Add** Fluid Chat's selected_message_id.

Workflow: Creating a Poll

Trigger: User configures a poll in the native attachment menu and clicks "Send Poll", triggering the Poll Created element event.

Action Routine: Instantiate a Create a New Thing step targeting the **Messages** data type, mapping fields as follows:

- **message** = Fluid Chat's new_poll_question
- **poll_data** = Fluid Chat's new_poll_options
- **is_poll** = yes
- **conversation_id** = Current actively loaded Conversation
- **created_by** = Current User

Workflow: Voting on a Poll

Trigger: User taps a poll option bubble, triggering the Poll Vote Pressed element event.

Action Routine: Trigger Make Changes to Thing pointing to the targeted **Message** item (Search for Messages where UniqueID = Fluid Chat's selected_message_id).

Field Mapping:

- **poll_data** = Fluid Chat's updated_poll_string

4. Fluid Chat (Mobile Native Edition Configurations)

The Native Mobile framework handles asset references dynamically using high-speed native device storage. Ensure the following specific property values are configured inside your native mobile application element viewport wrappers.

Mobile Plugin Editor Field	Target Property / Expression Mapping
Current User ID	Current User.Unique ID
Message IDs	Messages.UniqueID
Messages List	Messages.message
Creation Dates List	Messages.creation_date
Creator IDs	Messages.created_by.UniqueID
Creator Names	Messages.created_by.Name
Images List	Messages.images_list
File List	Messages.files_list
Audio Duration List	Messages.audio_duration
Incoming Attachment	Map directly to your local page-level string custom state variable tracking local cache image URI data before executing pipeline uploads.

5. Mobile Workflows & Local Cache Pipelines

Workflow: High-Speed Gallery Media Previews & Execution

User taps the Paperclip interface button component, executing the Attachment Pressed native container trigger event.

Action 1: Run Bubble Native Action Open camera library. Config requirements: Set Type configuration value profile to Multiple Photos; ensure Optimize Image size configuration parameter is actively Checked/Ticked. Crucial for 100ms UI rendering.

Action 2: Run Set State updating your local page attachment state string tracking variable: Attachment state = Result of Step 1 (Open Camera Library)'s each item's URL joined with "|||" (Allows instant native multi-image preview render in the canvas layout without network overhead).

User taps the primary Send action icon button, firing element event: Send Pressed.

Action 3: Run Create a New Thing under targeted table Messages:

created_by = Current User | creation_date = Current Date | conversation_id = Conversation

images_list = FluidChat's Uploaded Files URLs

message = FluidChat's Typed Text | is_edited/is_system_message = no

read_by = Current User | response_message_id = FluidChat's Draft Reply To ID

Workflow: High-Speed Real-Time Native Camera Processing

User taps the Camera icon button component, firing event: Camera Pressed.

Action 1: Run Bubble Native Action Open camera. Requirement: Ensure Optimize Image size configuration parameter is actively Checked/Ticked.

Action 2: Run Set State pointing to your page custom attachment tracker state variable: Attachment state = Result of Step 1 (Open Camera)'s URL

User clicks Send, firing standard Send Pressed mobile event context layout.

Action 3: Execute Create a New Thing under targeted data table Messages mapping fields identical to the Gallery Media pipeline structure explained above.

Workflow: Asynchronous Voice Note Capture & Binary S3 Pipeline Upload

User performs tap gesture interaction over microphone container asset component, triggering native event: Mic Pressed.

Action 1: Invoke plugin context action module: Start Recording.

User taps native Send action vector button container inside operational audio tracking view state layout interface, triggering target event: ready_to_upload (Send Audio).

Action 2: Invoke operational pipeline plugin script action: Upload to Bubble. Set the dynamic string parameter property Endpoint precisely matching: https://your-app.bubbleapps.io/version-test/fileupload

The hardware audio binary uploads to Amazon S3 in the background. On total upload success, the engine fires event trigger hook: Voice Uploaded Finished.

Action 3: Instantiate final structural data mapping record step via Create a New Thing targeting Messages schema fields:

created_by = Current User | creation_date = Current Date

files_list = FluidChat's Uploaded Voice URL split by "|||"

audio_duration = FluidChat's recording_duration

conversation_id = Conversation

is_edited = no | is_system_message = no | read_by = Current User

Cancellation Handling: If the user actively taps the trash container asset layer during live tracking capture modes, event hook Cancel Mic Pressed fires. Simply bind plugin element utility action wrapper Stop Recording directly beneath this event hook loop to clean local file cache paths automatically.