ISSN: 2754-6683

Journal of Marketing & Supply Chain Management

SCIENTIFIC Research and Community

Review Article

Open Access

Embedded Private Chat for Adding Real-Time Collaboration for Capital Markets Applications

Ananth Majumdar

USA

ABSTRACT

Capital market participants use multiple first party (internally built) and third party (externally built and hosted) applications as part of their business. As part of their day to day usage of these applications, there is a need for collaboration with internal (same-company) and external (cross-company) users. For this purpose, they use different channels to communicate like email, text messages or phone calls. Doing this is cumbersome and might expose them to regulatory risk. This paper details a method to embed a private chat within the first party and third party applications such that users can collaborate without having to switch context. This also allows them to seamlessly share data from within the applications and build various automations to optimize their workflows.

***Corresponding author**

Ananth Majumdar, USA.

Received: January 03, 2022; **Accepted:** January 10, 2022; **Published:** January 24, 2022

Introduction

Capital market participants, including traders, analysts, and portfolio managers, rely on a diverse ecosystem of applications to conduct their daily operations. These applications can be broadly categorized into two types:

1. **First-Party Applications (Internally Built):**
- Internal trading platforms: Proprietary systems designed to execute trades, manage positions, and analyze market data in real-time.
- Dealer portals: Custom-built interfaces that allow dealers to manage inventory, price securities, and interact with clients.
- Risk management systems: In-house tools for assessing and monitoring various types of financial risks.

2. **Third-Party Applications (Externally Built and Hosted):**
- Order Management Systems (OMS): Sophisticated platforms that facilitate the entire order lifecycle, from placement to execution and settlement.
- Execution Management Systems (EMS): Specialized tools for optimizing trade execution across multiple venues and asset classes.
- Market data terminals: Professional-grade software providing real-time financial information, news, and analytics.

As financial professionals navigate these various applications throughout their workday, there is a constant need for collaboration with colleagues and counterparties. This collaboration can be internal (within the same company) or external (across different organizations). Traditionally, users have relied on separate communication channels to facilitate this collaboration, including:

- Email: For sending detailed messages, attachments, and maintaining a record of conversations.
- Text messages: For quick, informal communications and time-sensitive updates.
- Phone calls: For real-time discussions and complex negotiations.

However, this approach of using multiple, disconnected communication channels presents several challenges:

- Inefficiency: Constantly switching between applications and communication tools disrupts workflow and reduces productivity.
- Context loss: Important context from the trading or analysis application may be lost when communicating through external channels.
- Data security: Sharing sensitive financial information through unsecured or unmonitored channels can pose significant risks.
- Regulatory compliance: Financial institutions are subject to strict regulations regarding communication and data sharing, which can be difficult to enforce across disparate channels.
- Audit trail: Maintaining a comprehensive audit trail of all interactions related to a particular trade or decision becomes challenging when communications are scattered across multiple platforms.

To address these issues, this paper proposes a method for embedding a private chat functionality directly within both first-party and third-party applications used in capital markets. This integrated approach offers several key benefits:

- Seamless context switching: Users can communicate without leaving their primary work environment, maintaining focus and efficiency.
- Enhanced data sharing: The embedded chat can allow for direct sharing of relevant data, charts, or analysis from within the application, ensuring accuracy and context.
- Improved compliance: By containing communications within approved systems, firms can better monitor and control

information flow to meet regulatory requirements.

- Workflow optimization: The integrated chat system can serve as a foundation for building automated workflows, such as trade approvals or risk alerts.
- Cross-application collaboration: Users can potentially communicate across different applications within the same chat ecosystem, bridging the gap between various tools and systems.

Secure Messaging Platform

There are a variety of messaging platforms all of which vary in their terms of security and privacy guarantees. Most of the platforms in use today use secure HTTP to transmit messages and save the messages encrypted in a database providing both transport security and security at rest.

In the system with true customer owned keys, the messaging providing application (central server) doesn't have access to the raw messages. It only works with the encrypted messages and some metadata specifying the tokens about the routing of messages.

In this system, each firm that wishes to use the secure messaging infrastructure will have a component on their premises which will interact with the hardware security manager (HSM) to generate the keys and work with the messaging client to encrypt the messages for sending to the central server.

Key wrapping is a method used to securely encrypt keys, allowing them to be safely stored or transmitted. It involves using a Key Encryption Key (KEK) to encrypt a Data Encryption Key (DEK) or other keys. This process ensures that the DEK is protected, even if the storage medium is compromised. By using this key wrapping method, a firm's internal key manager can exchange keys with the central server, using this to authenticate the users of the firm. To increase the privacy of conversations, each conversation is encrypted with a separate conversation key. By using sophisticated key wrapping cryptographic methods outlined in, it is possible to exchange the wrapped keys around the server and the users' client validated by the firm's key manager ensuring the authenticity and privacy of messages [1]. The keys are encrypted with a multilayer wrapping where the conversation key is wrapped with the user's account key which is then wrapped with the firm key so that privacy is protected at each layer.

It is also possible for users of different firms to also exchange messages securely. Since the messages in each conversation are encrypted with a separate encryption key, the central server can help exchange these keys using public key infrastructure. Firm 1 can sign the conversation key with firm 2's public key and send it to the server. The server can then send this signed key to the firm 2. Firm 2 can extract the conversation key with their private key gaining access to the key to decrypt the messages.

To further increase the privacy, the firm's key manager can rotate the keys per a fixed time like daily or weekly to reduce the surface area of exposed messages even if one conversation key leaks.

Approach

By allowing applications to embed the secure messaging platform described above into client portals, third party solution providers it should be possible to add real-time collaboration capabilities to those apps. A platform can register with the secure messaging platform which enables it to load the secure messaging platform in an iFrame mode within the third party platform.

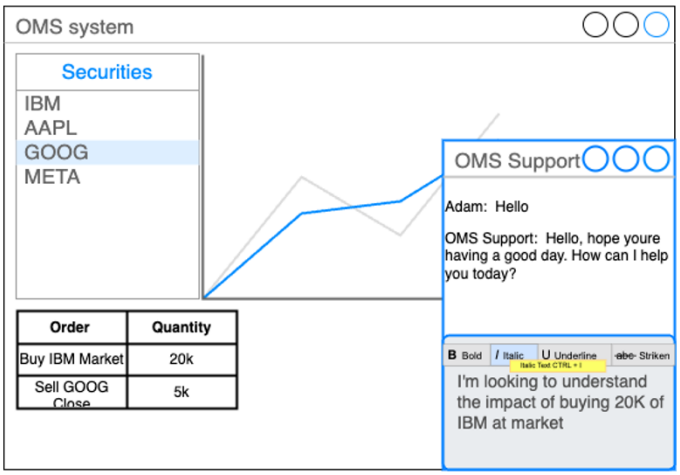

Figure 1: Mockup of Embedded Secure Collaboration

In Figure 1, the box on the right showing the OMS support window represents the embedded chat window. It is embedded in the OMS system allowing to chat directly in the application without leaving and context switching.

Authentication and Routing

There are three different scenarios possible depending on the use-case of embedded chat

- User already has an account with the chat platform provider and is visible to the outside network
- User has an account but it is private to the company and hence not visible to the outside network
- User doesn't have an account with the chat platform provider

This is how the authentication and routing works for each of these cases.

User Already has an Account and is Visible to the Outside Network

Since the user already has an account with the chat platform and it is visible to the outside network, the user can use the same account to login to the embedded chat. To do this, the embedded application can prompt the user to enter the email and based on the email determine the chat instance of that user and display the custom login page of that instance. Users can enter the credentials and login to the chat platform instance.

User Already has an Account and is not Visible to the Outside Network

Since the user already has an account with the chat platform but it is not visible to the outside network, it cannot be used as is in all the embedded applications. If it is a first party app, the user can then login to the application but for third party applications this is not possible. For first party applications, it would work exactly as it would work if it is visible to the outside network. The chat service would also get the login request and the email. Since the embedded application has registered with the chat platform, it can look up the registry to identify the owning company context. If the owning company is the same as the user's company it will redirect the user to the company's chat instance for login. The user can then login and continue to chat.

If the user is not part of the same company context as the embedding app, then the user cannot have an instance to login.

In this case the embedding application can use the chat platform's API to create a user on demand on a chat platform instance owned by the embedding app and allow the user to collaborate.

User Does not have an Account

If the user doesn't have an account then the user doesn't have an instance of the chat platform to login. In this case the embedding application can request the creation of an account using the chat platform API and once the user is created it can allow the user to login and collaborate.

Depending on the need of collaboration required in this application, the user can be set to be private where this user is only visible within the context of the embedded app and the user can collaborate within the embedded app. Alternatively if the user needs to collaborate outside of the embedded app context, their account can be set to be visible to the whole network and user can use this account to collaborate from within other applications also.

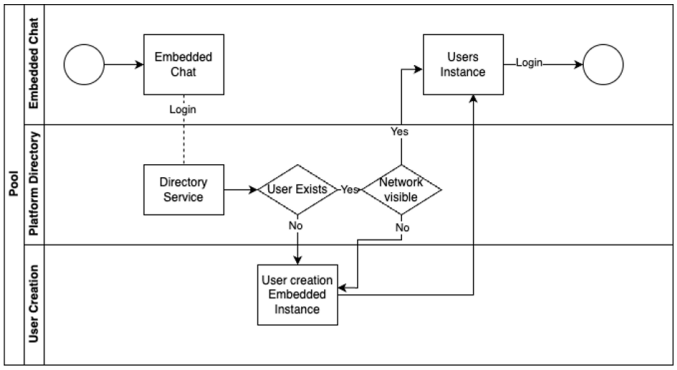

Figure 2: Flowchart Showing Authentication Routing for Embedded Chat

User Creation

As discussed in scenario 3.1.2 and 3.1.3 if the account doesn't exist or exists privately then to make it possible for the user to access the embedded chat, a user account needs to be created. The embedding app can use the user creation API provided by the chat platform to create the user. The chat platform has to quickly create the user and make it available for the user to login so that the whole experience can be seamless. These users can be created in the embedding app specific instance to expeditiously setup the users. The user detail information can be populated based on the information available to the third party platform.

Integrations and Automations

By embedding the chat application within the third party application, it is possible for automating various workflows using chat. The data from the application can be shared from the embedding app to chat. By using automated bot accounts, the third party apps can send messages into the chat. They can also have custom renderers for the messages which allows chat data to be shared back into the third-party platform. The chat can also be leveraged to send the pre-trade and post-trade fills information into the chat. It can also be used to capture trade approvals, send risk alerts and for sharing market colors among other use cases.

Auditing

For any financial service applications, it is important to have a record of the full transaction so that they comply with all the regulations [2]. It is needed to present the data for auditing purposes. This can also help capture the full transaction history over the application. The chat color can help capture qualitiative data to fully understand and analyze the transactions and improve the performance of various functions.

Focused View

With an embedded chat is possible for the embedded app to control the type of chat and completely style the chat such that the chat seems part of the app. They can decide to hide certain options and controls which they don't want to allow as part of the app. The embedding app can use the chat platform API to setup a chat with the participants included in the conversation. They could logically setup a chat for each object whether it be a trade, an RFQ or a deal whichever is the main thing the users work in the application. They could also use the API to close the conversation when the object is archived. This helps the users stay focused on the workflow within the app without any distractions.

Full View

Depending on the functionality in the app, there would be a need for users in the embedding app to have mulitple chats for one object in the app. For example if they are discussing a deal, there might be a need to have a chat with external parties discussing the deal. They could also have an internal room to discuss within their company before providing the details to their clients. In this case the embedding app needs to allow the user to switch chats. For this case, a fuller view of the chat app might be embedded. This will include not just one chat but a navigation bar which shows a list of chats. The user can select a chat and switch between the chats.

Evaluation and Results

This solution was evaluated with multiple different embedding applications. It was embedded in a single-dealer platform, an OMS application. In the single dealer platform, having a chat helped increase the customer satisfaction scores for the platform by 12%. For the OMS application, once the chat platform was made available, the trade breaks were significantly reduced and the resolution time for trade breaks went down by 50%.

Conclusion

This paper provides an overview of the problems faced by users of first and third party apps due to communicating outside of the applications using various communication methods like email, text and calls. It outlines a solution to this problem using a secure and private chat platform. It provides an overview of how an embedded chat within a first party and third party application can help resolve these issues and improve efficiency with automations. It also gives the details of the implementation and ways to customize the solution.

References

1. David MÄRaihi, David Gurle, Michael Harmon, Jon McLachlan, Ivan Rylach, et al. (2019) Secure End to End Communications.
2. SEC fines JPMorgan for noncompliant communication https://www.cnbc.com/2021/12/17/jpmorgan-agrees-to-125-million-fine-for-letting-employees-use-whatsapp-to-evade-regulators.html.