

Bot API Documentation

Revision history

Name	Date	Version	Summary of changes
Dino Diviacchi	08/14/2019	1.0	First version
Dino Diviacchi	04/11/2019	1.1	Included information on setting up a WebSocket

Contents

Revision history.....	2
Contents.....	3
Introduction	5
Eikon Messenger	5
Setup.....	6
Parameters	10
Authentication	11
Pre-requisite: Generate an APP Key	11
Request an Authentication Token.....	12
Use the Authentication Token in All API Calls	12
API Calls	13
1-1 Message Send.....	13
List Chatrooms.....	14
Join a Chatroom.....	15
Send a Message to a Chatroom	16
Leave a Chatroom	17
Create a Bilateral Chatroom	18
Send Invites	19
Get Participants List.....	21
Remove Participants.....	22
Get Invites.....	23
Post Invite	24
Message Formatting	25
Errors & Error Codes	25
Message Rate Throttling.....	25
API Limitations	25
WebSocket Connection	26
WebSocket Limitation	26
Establishing a Connection	26
Maintaining a Connection	27
WebSocket & Bot Behavior.....	28
Bilateral Chatroom Behavior	28
Managed Chatroom Behavior	28
1-to-1 Behavior	29
WebSocket Events.....	30

chatroomInvite	30
chatroomJoin	31
chatroomLeave	31
chatroomPost.....	32
message	32

Introduction

Welcome to the new Bot API which provides a set of available API calls to build automated workflows or bots for Eikon Messenger. The Bot API allows your applications to connect with and pass information into Eikon's Messenger service programmatically.

This document provides instructions on how to access and use our early beta Server Bot API to post content or messages via the API. Please note that this is an early version that provides a unidirectional API with a subset of the features that will be available in the official release, and the final version may differ in functionality and format.

The Bot API is a REST API and, in the future, will offer additional functionality for the return responses in Q4 2019. All request and response bodies are encoded in JSON.

Eikon Messenger

Eikon Messenger is an open messaging platform which provides many benefits:

- It is free to use, and available to anyone in financial services. Users can simply register via the Refinitiv website, or Refinitiv can assist in onboarding:
<https://www.refinitiv.com/en/products/eikon-trading-software/eikon-messenger-secure-messaging>
- It is integrated into the Eikon Desktop with a global community of over 300,000 trusted contacts in 30,000+ firms across 180+ countries.
- It is a compliant and secure messaging platform, allowing firms to fulfil regulatory obligations, with all chats automatically captured via our hosted network.
- It is a powerful communication tool that provides desktop, mobile, and web access, and allows sharing messages, data, files, charts, screenshots, and emoticons with your contacts
- It is an open network that allows individuals to communicate seamlessly and securely with contacts on other federated networks like CME.

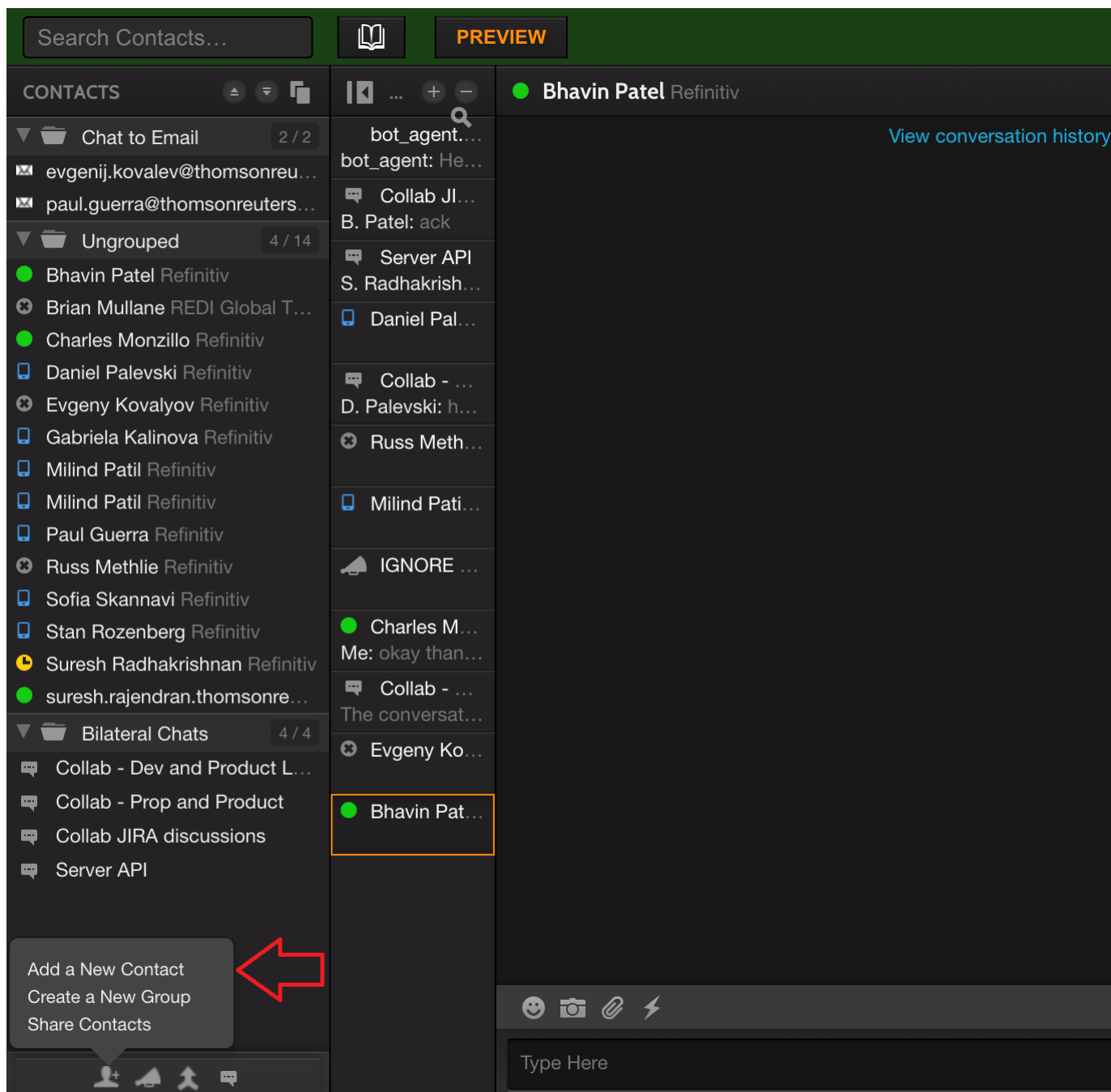
Setup

To access the BOT Messenger APIs, you must first contact your local account manager to ensure the proper licensing is setup. You must provide a bot name and a main contact email.

The Refinitiv team will then provision and set up the bot. Once this is done the email user you provided will receive an automated email with how to set up a password for the bot. In order to login the bot, you will need to follow the Authentication section below.

In order to have access to the bot, you must go to your personal Eikon Messenger to add the bot to your contacts, using “Add a New Contact” from the menu in the lower left corner, as illustrated in the following screenshots.

If you would like to have an interactive bot then you will need to establish a WebSocket connection. You will find information on how to establish this connection in the WebSocket section of the document.



Add a Contact

bot_agent.mybot@refinitiv.com

Messenger

Add Contact

After you add the bot, it will show up under your contacts (on the left side):

Search Contacts... PREVIEW

CONTACTS

- Chat to Email (2 / 2)
 - evgenij.kovalev@thomsonreuters.com
 - paul.guerra@thomsonreuters.com
- Ungrouped (4 / 15)
 - Agent BOT Test Dino EM BOTS (DEV) Refinitiv**
 - Bhavin Patel Refinitiv
 - Brian Mullane REDI Global Technologies LLC
 - Charles Monzillo Refinitiv
 - Daniel Palevski Refinitiv
 - Evgeny Kovalyov Refinitiv
 - Gabriela Kalinova Refinitiv
 - Milind Patil Refinitiv
 - Milind Patil Refinitiv
 - Paul Guerra Refinitiv
 - Russ Methlie Refinitiv
 - Sofia Skannavi Refinitiv
 - Stan Rozenberg Refinitiv
 - Suresh Radhakrishnan Refinitiv
 - suresh.rajendran.thomsonreuters.com@reuters...
- Bilateral Chats (4 / 4)
 - Collab - Dev and Product Leads
 - Collab - Prop and Product
 - Collab JIRA discussions
 - Server API

CONVERSATIONS

- Agent BOT Test Dino EM BOTS (DEV) Refinitiv
 - A. BOT Test Dino: Hello from Dino's API
 - Collab JIRA discussions
 - B. Patel: ack
 - Server API 11
 - bot_agent: Hello again! Testing Stats dashboard, pl...
 - Daniel Palevski Refinitiv
 - Collab - Dev and Product Leads
 - D. Palevski: hey @suresh.rajendran, where we at wi...
 - Russ Methlie Refinitiv
 - Milind Patil Refinitiv
 - IGNORE BLAST
 - Charles Monzillo Refinitiv
 - Me: okay thanks
 - Collab - Prop and Product
 - The conversation was buzzed.
 - Evgeny Kovalyov Refinitiv
- Bhavin Patel Refinitiv

Add a Contact

Sent join request to BOT_Agent.TestDino@tho...

Email Address

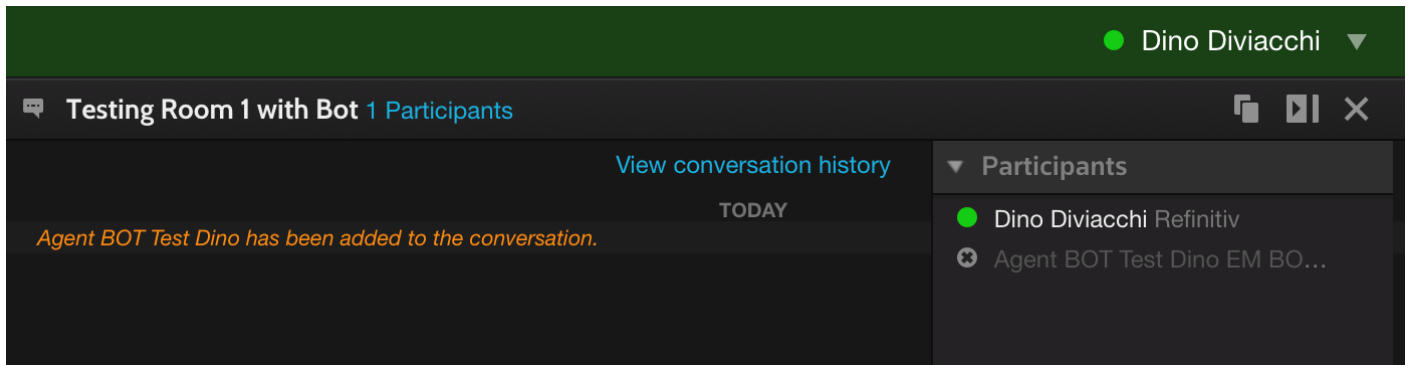
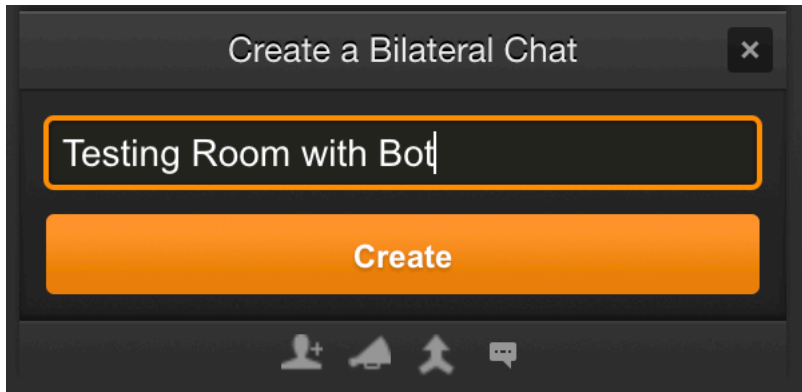
Eikon Messenger

Add Contact

Type Here

Please note that you and your messenger users will need to be logged in if you want to receive a message from a bot for the Beta release; this limitation should be fixed in Q4 2019.

If you want to send messages to a chatroom, you need to create a Bilateral chat with the bot and then drag the bot into the room. The bot must be dragged into any chatroom that you would like the bot to be a part of. It is important to note that the bot must be dragged into the room before you can use it. The bot will show up greyed out in the participant list on the right-hand side of Eikon Messenger. See the following screenshot as an example.



You will then be able to send messages to chatrooms using the API calls as described further below.

Parameters

Let us first describe all the parameters you can use in the API calls.

Client_ID

Client_ID is the APP Key that you generate to authenticate with Messenger APIs. You need to generate a key with products for API type EDP API at a minimum. Please refer to the next section on how to set this up.

Username & Password

Username is your bot's name. The email address you provided earlier will have received a welcome email containing the bot's username and password.

STS_TOKEN

The STS_TOKEN is an authentication token you add to the header of each HTTP request. You will get this token when you use the Authentication API call described in the next section.

chatroomId

The chatroomId is an ID for a particular chatroom. You can retrieve a list of chatroom IDs by using the chatrooms API call, and then you can use that chatroomId in additional HTTP operations.

Authentication

PRE-REQUISITE: GENERATE AN APP KEY

Authentication is done using an APP Key. An APP key is the Refinitiv term for API key.

You must generate an APP Key for your application, in order to authenticate with EDP and hit the Messenger end points. This is a once-of action.

In Eikon Desktop, go to the Search Bar and type 'APP KEY', then select the App Key Generator. Alternatively, you can find this is in the "Developer" section of the App Library.

Use the APP Key Generator app to generate the APP Key:

1. Enter an App Display Name
2. Select the tick box next to EDP API as the API Type
3. Click 'Register' New App button

App Display Name	Product ID	App Key	Owner	Company	Created	API Type	Status
------------------	------------	---------	-------	---------	---------	----------	--------

You will then see a row for your new app with an APP KEY item. Copy this APP Key into the "client_id" parameter, which is mentioned below in the Authentication header. In return, you will receive a response with a token (STS_TOKEN); that token will need to be passed in the header of all other HTTP commands.

If you do not have access to Eikon Desktop you can go to this link to obtain an App Key:

<https://apidocs.edp.thomsonreuters.com>

We strongly recommend you keep these keys hidden within your application.

REQUEST AN AUTHENTICATION TOKEN

To use the API calls you need an authentication token, which can be requested using a specific API call.

HTTP Request:

```
POST https://api.refinitiv.com/auth/oauth2/beta1/token
```

Request Header:

```
Content-Type: application/x-www-form-urlencoded
```

Request Body (replace the parts in triangular brackets with your parameters):

```
grant_type=password&username=<myBotUserName>&password=<myBotPassword>&client_id=<myBotAppKey>&scope=trapi.messenger&takeExclusiveSignOnControl=true
```

Success Response Payload:

```
{
  "access_token": "dKWnVPdGpGS1hl1SFRZZGo5emxLaU1TcnZXc19sekN3asdfy",
  "refresh_token": "f8dd081b-cccc-cccc-cccc-info",
  "expires_in": "300",
  "scope": "trapi.messenger",
  "token_type": "Bearer"
}
```

Example request using Curl:

```
Curl -d
'grant_type=password&username=BOT_Agent.Test%40refinitiv.com&password=Pa$$w0rd&client_id=ff444ffff44444ffff444&scope=trapi.messenger&takeExclusiveSignOnControl=true' \
-H "Content-Type: application/x-www-form-urlencoded" \
-s -X POST "https://api.refinitiv.com/auth/oauth2/beta1/token"
```

Notes:

In reality the access token (STS_Token) will return a much longer string than what is illustrated above.

Important: the token expires after 300 seconds; regularly refresh the token using a token manager.

USE THE AUTHENTICATION TOKEN IN ALL API CALLS

The header of **all** API calls must include the token:

```
Authorization: Bearer <STS_Token>
```

Example:

```
Authorization: Bearer dKWnVPdGpGS1hl1SFRZZGo5emxLaU1TcnZXc19sekN3asdfy
```

API Calls

1-1 MESSAGE SEND

Purpose:

Send a 1-1 message to a single user, identified by his email.

HTTP Request:

```
POST https://api.refinitiv.com/messenger/betal/message
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body (replace the parts in *italics* with your parameters):

```
{
  "recipientEmail": "email@refinitiv.com",
  "message": "Hello from my API"
}
```

Success Response Payload:

```
{
  "messageId": "f6472f7a-9c3f-cccc-cccc-0017a4771404",
  "timestamp": "2019-07-01T20:36:54.024383Z"
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -d
"{\"recipientEmail\": \"email@refinitiv.com\", \"message\": \"Hello from my API\"}" -X
POST https://api.refinitiv.com/messenger/betal/message;
```

LIST CHATROOMS

Purpose:

Get a list of chatrooms (and their Ids) that you are associated with on Eikon Messenger. Note that there are two types of Chatrooms on Eikon Messenger:

- *Bilateral* chatrooms - created by users, can only contain two companies for compliance reasons.
- *Managed* chatrooms - created by company administrators. You must make a request to be added

HTTP Requests:

```
GET https://api.refinitiv.com/messenger/betal/chatrooms
```

```
GET https://api.refinitiv.com/messenger/betal/managed_chatrooms
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Success Response Payload:

```
{
  "chatrooms": [
    {
      "chatroomId": "groupchat-1d8c807de411a92a584434",
      "name": "Trading Room",
      "createdTime": "2019-05-23T12:45:38Z"
    },
    {
      "chatroomId": "groupchat-34e21f3b7a2056b5f52034",
      "name": "Commodities Room",
      "createdTime": "2019-05-29T13:11:58Z"
    },
    {
      "chatroomId": "groupchat-bfc63868135f95ce8ddef7",
      "name": "Sales Team Room",
      "createdTime": "2019-05-30T19:41:39Z"
    }
  ]
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN"
https://api.refinitiv.com/messenger/betal/chatrooms
```

JOIN A CHATROOM

Purpose:

Once you have a list of chat rooms you can join the bot to a chat room, using the chatroom Id.

HTTP Requests:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/join
```

```
POST https://api.refinitiv.com/messenger/betal/managed_chatrooms/<chatroomId>/join
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body:

Even though this is a POST request, there is no request body required.

Success Response Payload:

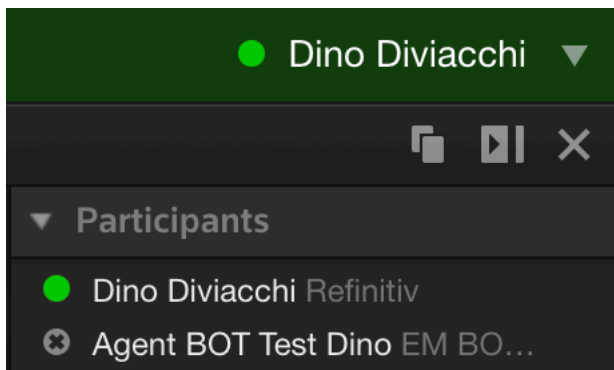
```
{
  "member":
  {
    "email": "bot_agent.test@refinitiv.com",
    "uuid": "GENTC-111",
    "affiliation": "admin",
    "role": "moderator"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
```

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/join
```

When the bot has joined the room, the bot's ID text will turn white like in the following screenshot.



SEND A MESSAGE TO A CHATROOM

Purpose:

Once a bot is joined to a room, it can send a message to the room.

HTTP Requests:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/post
```

```
POST https://api.refinitiv.com/messenger/betal/managed_chatrooms/<chatroomId>/post
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body:

```
{
  "message": "Hello from API"
}
```

Success Response Payload:

```
{
  "post":
  {
    "chatroomId": "Test with Bot",
    "messageId": "3172364",
    "message": "Hello from API",
    "member":
    {
      "email": "bot_agent.test@refinitiv.com",
      "uuid": "GENTC-11111",
      "affiliation": "admin",
      "role": "moderator"
    },
    "timestamp": "2019-07-01T20:59:48.731397357Z"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -d '{"message\\":\\"Hello from API\\"}' -X
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/post;
```

LEAVE A CHATROOM

Purpose:

Make a bot leave a bilateral or managed chat room.

HTTP Request to leave a bilateral chatroom:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/leave
```

```
POST https://api.refinitiv.com/messenger/betal/managed_chatrooms/<chatroomId>/leave
```

Request Header:

```
Authorization: Bearer <STS-Token>
```

Request Body:

Even though this is a POST request, there is no request body required.

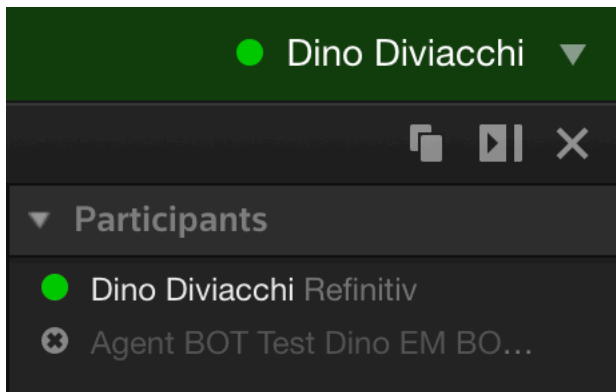
Success Response Payload:

```
{
  "member":
  {
    "email": "bot_agent.test@refinitiv.com",
    "uuid": "GENTC-11111",
    "affiliation": "admin",
    "role": "none"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/leave
```

When the bot leaves the room, the bot's ID will be greyed out again like in the screenshot below.



CREATE A BILATERAL CHATROOM

Purpose:

A bot can create a chatroom. Note that you will not see this chatroom on your Eikon Messenger until the bot invites you and you accept the invite.

HTTP Requests:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body:

```
{
  "name": "Equity Trader Room"
}
```

Success Response Payload:

```
{
  "room": {
    "chatroomId": "groupchat-7f92ac05cda7821158524954",
    "name": "Equity Trader Room",
    "createdTime": "2019-09-27T08:46:15.460564606Z"
  },
  "owner": {
    "email": "email@refinitiv.com",
    "uuid": "ABC-384",
    "affiliation": "owner",
    "role": "none"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -d '{"name": "Name of Room"}' -X
POST https://api.refinitiv.com/messenger/betal/chatrooms
```

SEND INVITES

Purpose:

Allow a bot to send invites to users to join a chatroom.

HTTP Request to leave a bilateral chatroom:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/invite
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body:

```
{
  "emails": ["email1@refinitiv.com, email2@refinitiv.com"]
}
```

Success Response Payload:

```
{
  "inviteResults": [
    {
      "email": "email1@refinitiv.com",
      "invite": {
        "inviteId": 1111,
        "inviteStatus": "pending",
        "created": "2019-09-27T08:54:50.864529437Z",
        "from": {
          "email": "bot_agent.test@refinitiv.com",
          "uuid": "abc-1111",
          "affiliation": "owner",
          "role": "none"
        },
        "to": {
          "email": " email1@refinitiv.com ",
          "uuid": "abc-2222",
          "affiliation": "admin"
        }
      }
    }
  ]
}
```

```
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X  
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/invite
```

GET PARTICIPANTS LIST

Purpose:

Allow a bot to get a list of participants in a chatroom.

HTTP Request to leave a bilateral chatroom:

```
GET https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/participants
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Success Response Payload:

```
{
  "participants": [
    {
      "email": "email@refinitiv.com",
      "uuid": "ABCD",
      "affiliation": "owner",
      "role": "moderator"
    },
    {
      "email": "bot_agent.test@refinitiv.com",
      "uuid": "ABCD",
      "affiliation": "admin",
      "role": "moderator"
    }
  ]
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
GET https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/participants
```

REMOVE PARTICIPANTS

Purpose:

A bot can remove participants from a chatroom. It is important to note that the bot can only remove participants from a chatroom that the bot created.

HTTP Request to leave a bilateral chatroom:

```
POST
https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/participants/remove
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Request Body:

```
{
  "emails": ["email1@refinitiv.com"]
}
```

Success Response Payload:

```
{
  "participants": [
    {
      "email": "email@refinitiv.com",
      "uuid": "ABCD",
      "affiliation": "owner",
      "role": "moderator"
    },
    {
      "email": "bot.test@refinitiv.com",
      "uuid": "ABCD",
      "affiliation": "admin",
      "role": "moderator"
    }
  ]
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/participants
/remove
```

GET INVITES

Purpose:

Allow a bot to get invites to a chatroom.

HTTP Request to leave a bilateral chatroom:

```
GET https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/invites
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Success Response Payload:

```
{
  "member":
  {
    "email": "bot_agent.test@refinitiv.com",
    "uuid": "GENTC-11111",
    "affiliation": "admin",
    "role": "none"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
GET https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/invites
```

POST INVITE

Purpose:

Allow a bot to send an invite to a chatroom.

HTTP Request to leave a bilateral chatroom:

```
POST https://api.refinitiv.com/messenger/betal/chatrooms/<chatroomId>/invite
```

Request Header:

```
Authorization: Bearer <STS_Token>
```

Success Response Payload:

```
{
  "member":
  {
    "email": "bot_agent.test@refinitiv.com",
    "uuid": "GENTC-11111",
    "affiliation": "admin",
    "role": "none"
  }
}
```

Example request using Curl:

```
curl -H "Authorization: Bearer $STS_TOKEN" -X
POST https://api.refinitiv.com/messenger/betal/chatrooms/{chatroom_id}/invite
```



Message Formatting

Note: Eikon Messenger supports tabular data, hyperlinks and a full set of emoji that can be found here: <https://www.webfx.com/tools/emoji-cheat-sheet/>

Here is a sample post that illustrates some of these capabilities:

```
{ "Message": "USD BBL EU AM Assessment at 11:30 UKT\nName\tAsmt\t10-Apr-19\tFair Value\t10-Apr-19\tHst Cls\nBRT Sw APR19\t70.58\t05:07\t(up) 71.04\t10:58\t70.58\nBRT Sw MAY19\t70.13\t05:07\t(dn) 70.59\t10:58\t70.14\nBRT Sw JUN19\t69.75\t05:07\t(up) 70.2\t10:58\t69.76"} }
```




And here is the result:

 BOT_Agent messenger04 testing for Dawn

View conversation

TO

13:58:52 BOT_Agent messenger04 USD BBL EU AM Assessment at 11:30 UKT

Name	Asmt	10-Apr-19	Fair Value	10-Apr-19	Hst Cls
BRT Sw APR19	70.58	05:07	 71.04	10:58	70.58
BRT Sw MAY19	70.13	05:07	 70.59	10:58	70.14
BRT Sw JUN19	69.75	05:07	 70.2	10:58	69.76

Errors & Error Codes

A list of error codes will be provided soon.

Message Rate Throttling

A throttling mechanism limits messages to a maximum rate of one message per second.

An error is returned if this limit is exceeded.

API Limitations

Uni-directional for now, thus you can only send and not receive messages.

This is not available to federated users.

WebSocket Connection

WebSockets are used to provide a persistent connection between two endpoints. They allow for access to real-time communication and allow a bot to listen for events in order to create a more interactive bot. In order to have this capability, you need to set up a WebSocket for each bot. In order to setup a WebSocket connection please follow the below instructions.

In a later section you will see a list of events that could stream across a WebSocket that a developer can use while coding a bot. These events occur as a user or bot interact with the Messenger.

WEBSOCKET LIMITATION

Please note there will be regular server maintenance on weekends so bots need to be able to reconnect with the WebSocket and a developer should take that into account while writing his code.

ESTABLISHING A CONNECTION

Purpose:

Establish an initial connection for a WebSocket with Eikon Messenger's server. Setting this socket up will be different for each programming language.

1. Authenticate

First retrieve an STS_TOKEN from our API gateway to ensure you are authenticated and authorized for using the Bot API. See the Authentication section at the beginning of the document for obtaining an STS token.

2. Obtain Connection Details

Next request Eikon Messenger WebSocket endpoints and retrieve connection information and settings for establishing a connection with the Eikon Messenger server. Make an HTTP call to the endpoint below in order to retrieve the WebSocket URL:

```
https://api.collab.refinitiv.com/messenger/betal/stream/events
```

Making a call to this connection will return a payload with a WebSocket endpoint (URL) that you will use for connecting to the WebSocket. The reason this approach was taken, was in case the WebSocket endpoint changes in the future, so that it will limit the impact on your use of the WebSocket and bot. The WebSocket endpoint will look something similar to this:

```
wss://beta.api.collab.refinitiv.com:443/services/nt/api/messenger/v1/stream
```

3. Connect

Create a WebSocket with Eikon Messenger server by using the provided connection information from point two. A new WebSocket connection should be created with to this endpoint and the appropriate connection payload should be sent with the bot's STS_TOKEN to ensure they are properly authenticated.

4. Validate Auth token

During the third point, the Eikon Messenger server will validate the provided STS_TOKEN from the client/bot against the STS authentication service and on failure, the socket will be closed.

MAINTAINING A CONNECTION

Purpose:

Keep the WebSocket connection persistent by refreshing the token, otherwise you will lose the connection.

1. Re-Authenticate

The client will retrieve a new STS_TOKEN from our API gateway to ensure the bot is authenticated and authorized for using the API. This must be completed, at most, every 5 minutes. See the Authentication section at the beginning of the document for obtaining an STS token.

2. Validate STS-Token

The client will resend the STS_TOKEN retrieved in the previous point and it **must** be completed before the 5 minute session timeout.

3. Validate Session

The Eikon Messenger server will validate the new STS_TOKEN with the STS Authorization Service.

WebSocket & Bot Behavior

Purpose:

This section will describe a bot's presence and their behavior in chatrooms or with a 1-1 message.

BILATERAL CHATROOM BEHAVIOR

Presence Behavior

- Bots will see the presence of others whom are in the chatroom when they first join a chatroom while a WebSocket connection is open.
- Bots will continue to see users' presence updates for as long as they remain joined. This means a bot can see as users join and leave the room and will receive the chatroomJoin and chatroomLeave events (more on events in the next section).
- Bots can join a room before or during an established WebSocket connection.
- Bots who disconnect from a WebSocket connection will be parted from all of their bilateral chatrooms and will have to rejoin if they want to be part of the room and conversation.
- Bots will **NOT** see their presence or another bot's presence.

Post Message Behavior

- Bots will receive chatroomPost events from users who post to the rooms they are in.
- Bots will **NOT** receive their own chatroom post message events or chatroom post message events from other bots.

MANAGED CHATROOM BEHAVIOR

Presence Behavior

- Presence Events are propagated within 30 seconds due to a buffer handling any overload of users joining a managed chatroom.
- Bots will **ONLY** receive their own presence to a room when they first join if a WebSocket connection is open. They will **NOT** receive the presence of all users who are currently in the room when they join a room.
- Bots will continue to see users' presence updates for as long as they remain joined. This means a bot can see a user join and leave the room and will receive the chatroomJoin and chatroomLeave events
- Bots can join a room before or during an established WebSocket connection.
- Bots who disconnect from a WebSocket connection will **NOT** be parted from all of their rooms.
- Bots will **NOT** see their presence or another bot's presence.

Post Message Behavior

- Bots will receive chatroomPost events from users who post messages to the rooms they are in.
- Bots will **NOT** receive their own chatroom post message events or chatroom post message events from other bots.

1-TO-1 BEHAVIOR

User to Bot

- Bot Online (WebSocket connected): Bots will receive messages from users and users will receive delivery receipts on the UI.
- Bot Offline (WebSocket not connected): Users will receive an auto reply on the bot's behalf that says, "This bot is currently unavailable to receive messages."

Bot to Bot

- Not Supported: Bots will not be able to interact with other bots.

WebSocket Events

This section will provide a list of events streaming across a WebSocket that a developer can use while coding a bot. These events occur as a user or bot interact with the Messenger.

connected

When a user first connects the WebSocket, this event will be the first event received over the WebSocket

Prerequisite:

Authenticating and connecting the Websocket

Payload:

```
{
  "event": "connected",
  "reqId": " 302.784654",
}
```

chatroomInvite

When a user invites a bot to a chatroom, this event will be received over the WebSocket

Prerequisite:

None

Payload:

```
{
  "event": "chatroomInvite",
  "chatroomId": "groupchat-9437f32bfe8b354ec7c47c6",
  "from": {
    "email": "email@refinitiv.com",
    "company": "Refinitiv"
  },
  "status": "pending"
}
```

chatroomJoin

When a user or the bot joins a chatroom the chatroomJoin event occurs. Once a bot joins a room it enables the bot to receive all events for a given chatroom. You will not get other bot chatroomJoin events.

Prerequisite:

The bot must be already joined to a chatroom to receive another user's chatroomJoin event

Payload:

```
{
  "event": "chatroomJoin",
  "roomPresence": {
    "chatroomId": "groupchat-9437f32bfe8b354ec7c47c6",
    "chatroomType": "chatroom",
    "user": {
      "email": "bot_agent.name@refinitiv.com",
      "company": "Refinitiv"
    },
    "affiliation": "admin",
    "role": "moderator"
  }
}
```

chatroomLeave

When a bot or user leave the chatroom. This prevents a bot from receiving any events for that chatroom.

Prerequisite:

The bot must be already joined to a chatroom

Payload:

```
{
  "event": "chatroomLeave",
  "roomPresence": {
    "chatroomId": "groupchat-9437f32bfe8b354ec7c47c6",
    "chatroomType": "chatroom",
  }
}
```

```
    "user": {
      "email": " bot_agent.name@refinitiv.com ",
      "company": "Refinitiv"
    },
    "affiliation": "admin",
    "role": "none"
  }
}
```

chatroomPost

When a bot or user posts a message to a chatroom.

Prerequisite:

The bot must be already joined to a chatroom

Payload:

```
{
  "event": "chatroomPost",
  "post": {
    "chatroomId": "groupchat-9437f32bfe8b354ec7c47c6",
    "chatroomType": "chatroom",
    "sender": {
      "email": "bot_agent.name@refinitiv.com"
    },
    "message": "Hello Everyone, thanks for the invite, I can provide information.",
    "messageId": "19",
    "timestamp": "2019-07-23T11:52:25.8211106Z"
  }
}
```

message

When a user sends a bot a 1-1 message.

Prerequisite:

None

Payload:

```
{
  "event": "message",
  "message": {
    "sender": {
      "email": "bot_agent.name@refinitiv.com",
      "company": "Refinitiv",
    },
    "message": "Hello Everyone, thanks for the invite, I can provide information",
    "messageId": "19",
    "timestamp": "2019-07-23T11:52:25.8211106Z"
  }
}
```

