Version 1.25

LSEG Workspace | Messenger

User Message Feed Technical Overview

LSEG DATA & ANALYTICS

Contents

About User Message Feed

The User Message Feed application empowers organisations to access messages programmatically through the LSEG Messenger platform. Designed for customer IT administrators and developers, this feature provides visibility over messages exchanged through LSEG Messenger by users within their organisation.

Deployment support

Currently, User Message Feed supports LSEG Messenger users with the hosted internet deployment method only.

Intended readership

This document is intended for developers who are responsible for the automation of message feeds for the following business areas within financial organisations:

- Sales
- Commodities
- FX
- Broker

Further information

To:

- Request product assistance, contact Support.
- Access other LSEG Workspace technical content, see the Workspace technical documentation site.
- Provide feedback on Workspace technical content, contact DocFeedback@lseg.com.

Prerequisites

To run User Message Feed, the requirements listed in the sections below must be met.

Network requirements

Domain whitelisting

To ensure seamless connectivity and functionality of the User Message Feed, the following domains must be accessible from the customer's infrastructure:

- api.refinitiv.com
- cdn.refinitiv.com
- messenger.collaboration.refinitiv.com
- a-fallback.collaboration.refinitiv.com
- b-fallback.collaboration.refinitiv.com
- c-fallback.collaboration.refinitiv.com
- d-fallback.collaboration.refinitiv.com
- e-fallback.collaboration.refinitiv.com

Ports

No custom ports are required. The application operates over standard HTTPS using:

- TCP Port 443

You must ensure that outbound access to the above domains over port 443 is permitted through your firewall or proxy settings.

Certificate installation

To prevent the User Message Feed from being blocked by endpoint security tools, clients must work with their IT teams to whitelist the Ably certificate on all machines running the app.

You can download the certificate from: https://ably.com/

This step is essential to enable a secure, encrypted connection between the app and the Messenger platform.

Software requirements

The following software packages must be installed:

Application	Version	Requirements
Java Platform Standard Edition	Java SE version 17 or higher	This must be installed on computer that will be used to execute the User Message Feed application
User Message Feed	message-feed-0.0.42.0.zip	These can be downloaded from LSEG Developer Community https://developers.lseg.com/en/api-catalog/lseg-messenger/lseg-messenger/download
Example plugin file	message-feed-plugins-example.zip	

LSEG Workspace accounts

To request the creation of the following accounts and assignment of licenses, contact your account manager.

Service account licenses

A service account with the following license(s) is required. This account is used to subscribe to all chats to access delivered messages. You can create multiple message feed sessions concurrently under a single service account.

Service accounts are created in the Platform Administration Application (PAA).

License type	License	Deployment method
Base	EAP LSEG MESSENGER	Elektron Delivery
Add-on	MESSENGER USER FEED	Elektron Delivery

User account licenses

User ID(s) of whom you want to monitor messages sent through their LSEG Messenger. The user ID requires the following license(s).

License type	License	Deployment method
Base	Workspace MESSENGER or any Workspace product variant with embedded LSEG Messenger	Hosted-Internet/Customer-Managed
Add-on	EAP MESSENGER FEED	Hosted-Internet/Customer-Managed

Important:

- The service account and the user account(s) must be located under the same location account (A-Number).
- Service account creation, user account creation, and license assignment can be done through EAS or PAA, depending on how you currently manage users and licenses for your company. For further information, refer to User and licence management.
- User-Account must include both base product and add-on, installed with the same deployment method.
- Once a service account is created, contact LSEG to register the service account.

Running User Message Feed

To run User Message Feed, do the following:

1. Extract `message-feed-0.0.42.0.zip` downloaded from prerequisites to your preferred location. The extracted files will be in the correct folder structure.

```
/message-feed
   message-feed-app-x.x.x.jar
   message-feed-handler-x.x.x.jar
   /plugins
       message-feed-plugin-example.jar
```

File name	Description
`message-feed-app-x.x.x.x.jar`	The main message feed application file
`message-feed-handler-x.x.x.x.jar`	The message feed common library: ChatroomMessageHandler[1] interface
`message-feed-plugin-example.jar`	This is an example of the custom message feed plugin provided by LSEG. The plugin prints out all the messages sent by the monitored user ID to the Java console. Customer can customise how they want to process the monitored messages as needed. LSEG provides example of the source code (example development plugins file: message-feed-plugins-example.zip) which can be downloaded from LSEG Developer Community.

2. Set up plugins: Customers can develop their own plugins by following Customer-defined plugins and place the plugins JAR file in the `/plugins` folder, before running a command to start the app (see step 3).

3. Prepare the following parameters to start the message feed app:

Mandatory parameters

Parameter	Description	Example
`Dserviceaccount`	*<service account UUID>* is used to define the service account	service account: GE-AAAAAA
`Dpwd`	*<service account password>* is used to specify the service account password	password: abcXY-Z123-1123-kuhe
`DtrackUUID or DtrackEmail`	*<user UUID to track>* or *<user UserId to track>* is used to specify the user email address to monitor.	UUID:GE-AAAAAA

Optional parameters

Parameter	Description	Example
`Dplugin.dir`	*<plugin directory>* is used to specify the directory path for your plugin jar files. The default directory is `/message-feed/plugins`. You must save at least one plugin jar file under this folder.	`C:\custom path\message-feed\plugins`
`Denv`	*<environment>* is used to define one or more supported environments: `prod`, `qa`, `trd`, `beta`, and `dev`. The default value is `prod`.	`trd`
`Dablyverboselog`	`True` or `false` is used to specify whether to enable verbose log on the console. The default value is `false`.	`true`
`Dloglevel`	`TRACE`, `DEBUG`, `INFO`, `ERROR`, or `FATAL` can be used to define the log entry detail level. The default value is `DEBUG`.	`INFO`

[1] `com.refinitiv.collab.platform.msgfeed.handler.ChatroomMessageHandler`

4. To start the User Message Feed app, from the `message-feed` folder, run either of the following commands, replacing the parameter values with the information prepared in step 3.

```
java "-Djava.system.class.loader=com.refinitiv.collab.platform.msgfeed.keep.DecryptClassLoader" -jar
message-feed-0.0.42.0.jar "-Dserviceaccount=<service account UUID>" "-Dpwd=<service account password>"
"-DtrackUUID=<user UUID to track>" "-Denv=<environment>"
```

or

```
java "-Djava.system.class.loader=com.refinitiv.collab.platform.msgfeed.keep.DecryptClassLoader" -jar
message-feed-0.0.42.0.jar "-Dserviceaccount=<service account UUID>" "-Dpwd=<service account password>"
“-DtrackEmail=<user UserId to track>” "-Denv=<environment>"
```

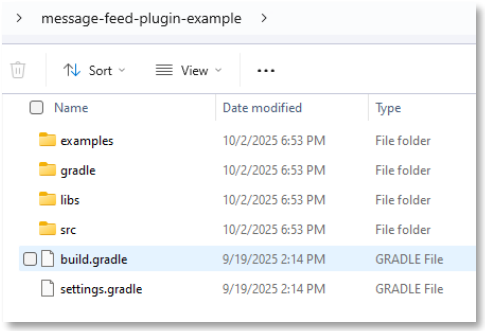
For example:

```
java "-Djava.system.class.loader=com.refinitiv.collab.platform.msgfeed.keep.DecryptClassLoader" -jar
message-feed-0.0.42.0.jar "-Dserviceaccount=GE-D7WT7HTECM4U" "-Dpwd=89fd6b8c-fdfe-44ce-9447-
be5a02db689a" “-DtrackEmail=test@lseg.com” "-Denv=prod"
```

Messages are now received by the User Message Feed app and are sent to your plugins in this example the example plugins will then print all the messages on the active console. Customer can develop their own custom jar file to handle messages.

Customer-defined plugins

LSEG provides an example of the source code (message-feed-plugins-example.zip) which can be downloaded from the LSEG Developer Community. The contents of this file are shown opposite.



`CustomerPlugin.jar` files are used to handle messages and are developed by the customer. At least one of the customer-developed jar files must be stored in the `/plugins` directory.

The plugin jar file must be created as a *fat jar*. That is, all libraries should be included in this jar. For more information about creating a fat jar, refer to the following website:

https://www.baeldung.com/gradle-fat-jar

Each plugin jar file must implement at least one `ChatroomMessageHandler`[2] interface. This interface is maintained by the Messenger Platform Team and can be found in the common library, `message-feed-handler-x.x.x.jar`.

Plugin processing overview

When the User Message Feed app receives a user message, the message is forwarded to the ChatroomMessageHandler interfaces that are implemented in the customer-defined plugin jar files and stored in the `/plugins` directory. For example, if a customer creates two plugin jar files, and the interface is implemented once in each file, the received messages are handled twice, once for each file.

Deleting plugin jar files

You can update or add new plugin jar files even when the User Message Feed app is running. However, to delete plugin jar file(s), you must do the following:

1. Shutdown the User Message Feed app.
2. Delete the plugin jar file(s).
3. Restart the User Message Feed app.

Message structure

Messages are passed into the handler as JSON strings. The structure of a JSON string can be found in the common library, `message-feed-handler-x.x.x.jar`, under `com.refinitiv.collab.platform.msgfeed.Data.ChatMessageEvent`.

For an example of the full message structure, refer to Appendix I: Message structure.

[2] `com.refinitiv.collab.platform.msgfeed.handler.ChatroomMessageHandler`

Support

Planned maintenance windows

Messenger feed backend maintenance could interrupt the active User Message Feed sessions that are run over a weekend. In this event, customers may find that:

- User Message Feed sessions error, and
- To continue monitoring the delivered messages by UUID(s), all User Message Feed apps must be restarted.

Rectifying the Session already exists issue

A User Message Feed app session cannot start if another session is active that uses the same service account and the user account pair that is activated. In this event, the `Session already exists!` error is shown.

To rectify this issue, contact the person running a User Message Feed app session with the same account pair to request that their app session is stopped.

Appendix I: Message structure

The message structure, below, includes properties for multiple different event types::

```
@Data
@ToString(callSuper = true)
@EqualsAndHashCode(callSuper = true)
@JsonInclude(JsonInclude.Include.NON_NULL)
public class ChatMessageEvent extends AbstractEvent<ChatMessageEvent.EventData> implements MessageEvent {

    public ChatMessageEvent() {
        this.eventType = MessageEventType.ChatMessageEvent.getValue();
    }

    @Data
    @JsonInclude(JsonInclude.Include.NON_NULL)
    public static class EventData {
        @JsonProperty("clientRequestId")
        private String clientRequestId;
        @JsonProperty("clientRequestDate")
        private String clientRequestDate;
        @JsonProperty("chatRoomId")
        private String chatRoomId;
        @JsonProperty("complianceWarningApprove")
        private String complianceWarningApprove;
        @JsonProperty("messengerUuid")
        private String messengerUuid;
        @JsonProperty("userUuid")
        private String userUuid;
        @JsonProperty("messageId")
        private String messageId;
        @JsonProperty("createAt")
        private String createAt;
        @JsonProperty("messageType")
        private String messageType;
        @JsonProperty("message")
        private String message;
        @JsonProperty("attachments")
        private List<Attachment> attachments;
        @JsonProperty("blastMessage")
        private Boolean blastMessage;
        @JsonProperty("blastId")
        private String blastId;
        @JsonProperty("command")
        private JsonNode command;
        @Getter(AccessLevel.NONE)
        @Setter(AccessLevel.NONE)
        @JsonProperty("isBot")
        private boolean isBot;
        @JsonProperty("messageExtension")
        private JsonNode messageExtension;
        @JsonProperty("replyTargetMessageId")
        private String replyTargetMessageId;
        @JsonProperty("replyTarget")
        private JsonNode replyTarget;
        @JsonProperty("urlDetails")
        private JsonNode urlDetails;

        public boolean getIsBot() {
            return this.isBot;
        }

        public void setIsBot(boolean isBot) {
            this.isBot = isBot;
        }
    }
}

@Data
@ToString
@JsonInclude(JsonInclude.Include.NON_NULL)
```

```
public class Attachment {
    @JsonProperty("attachmentReference")
    private String attachmentReference;
    @JsonProperty("attachmentType")
    private String attachmentType;
    @JsonProperty("attachmentName")
    private String attachmentName;
}

@Data
@ToString(callSuper = true)
@EqualsAndHashCode(callSuper = true)
@JsonInclude(JsonInclude.Include.NON_NULL)
public class MsgFeedChatMessageEvent extends ChatMessageEvent {
    @JsonProperty("additionalData")
    private Map<String, String> additionalData=new HashMap<>();
}
```

The following example is of a message event:

```
{
  "eventData": {
    "clientRequestId": "new:blc:3MmgLoxXdO4kHWt4fzbuCa:5e424630-63af-4b4f-92a6-2d3ffa368295",
    "clientRequestDate": "2026-03-09T07:32:34.221Z",
    "chatRoomId": "blc:3MmgLoxXdO4kHWt4fzbuCa",
    "messengerUuid": "GE-SFHSMUJB4HIU",
    "userUuid": "GE-SFHSMUJB4HIU",
    "messageId": "20260309073235039#139hkeQBHzhVW9rtdkhA8E",
    "createAt": "2026-03-09T07:32:35.039Z",
    "messageType": "text/plain",
    "message": "Hello World",
    "attachments": [
      {
        "attachmentReference": "5858834d-1286-47d9-a7d7-50503a29b879",
        "attachmentType": "application/pdf",
        "attachmentName": "MQ_UMF.pdf"
      }
    ],
    "isBot": false
  },
  "eventType": "chatroom.message",
  "additionalData": {
    "chatRoomName": "BLC FX Options",
    "userId": "peter.martin@lseg.com"
  }
```

}

lseg.com

LSEG DATA & ANALYTICS

Document version: 125.01