UCCX & SocialMiner: Bubble Chat and Post Chat Rating Log Analysis

Contents

Introduction

This document describes the Bubble chat flow along with the end-to-end log analysis for a working bubble chat, which can be used as a working reference to troubleshoot the issues.

The Unified Contact Center Express (UCCX) solution with the release of UCCX and SocialMiner (SM) 11.6(2) have added the new bubble chat feature.

The Bubble Chat (or Chat Bubble) feature allows you to reach a business using a minimally intrusive, floating chat webform that moves with the web page (with scrolling), is totally customizable and also instantly updates any customization without the need to re-deploy the webform on the site.

Contributed by Jayant Suneja, Arunabh Bhattacharjee, Cisco Engineering.

Logs Required

In order to trace the whole flow, logs cover the chat initiator (customer) to the UCCX.

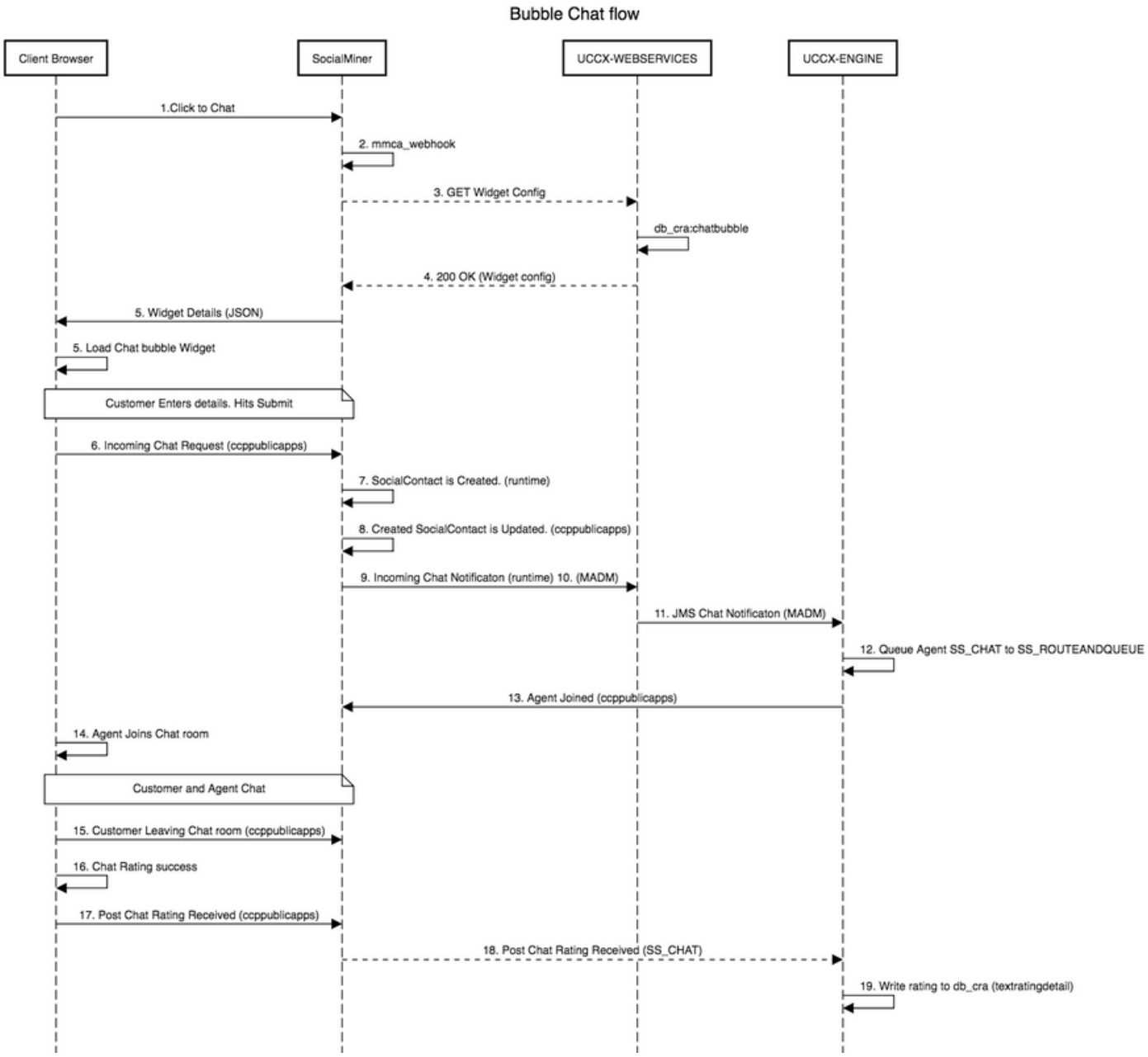
- Client Console Logs: These are the browser console logs where the end usmer initiates the chat.
- SM Logs: **ccppublicapps** logs, **runtime** logs, **tomcat** logs.
- UCCX logs: MIVR logs (**Engine logs** with **SS_CHAT** and **SS_ROUTEANDQUEUE** debugging), MADM logs (**CCX Admin** logs With **UCCX_WEBSERVICES**).

The rest of the analysis (after the chat room is joined) is identical to the classic chat flow (i.e simple XMPP eventing between the SocialMiner Agent gadget on Finesse and the end customer - who are connected over an XMPP tunnel)

Bubble Chat Flow

The flow can be summarized by the 19 steps shown in this image.

Each step is outlined in the logs.



Logs Analysis

When a customer clicks on the Click to chat button, the Bubble chat form is presented using the JavaScript code in the chat widget.

The Bubble chat form does server-side rendering - where it fetches the chat configuration to load the UI for the end user to start a chat.

If you open the chat widget, you will see the URL configured in this format:

https://<SOCIALMINER>**/ccp/ui/BubbleChat.html?host=**<SOCIALMINER>**&wid=**<WIDGET_ID>**&secure=true**
The <SOCIALMINER> and <WIDGET_ID> are the SocialMiner FQDN and the widget ID respectively.

The configuration received here is in the JSON form.

Step 1. Client Console logs: Click to Chat.

1. Once end customer presses **Click to chat** on the webpage it does a GET to SM through these three URLs.

2. Note the Widget ID is sent in the third request.

3. All these should return with a successful **200 OK** in order to load bubble chat window.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 2. SM ccppublicapps: DB dip to mmca_webhook

1. SM does a db dip to the **mmca_webhook** table on the SM Informix database and fetches the Webhook URL for this incoming chat request with the help of Widget ID. In our case, it is **widegetid=1**.

2. SM then uses this webhook URL to fetch the widget configuration from UCCX.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 3. UCCX MADM: Receives Request for the Widget Config.

1. The UCCX Webservices as a part of MADM receives this request to get the widget configuration.

2. The Widget configuration contains the font typefaces, colours, the styling of the chat widget, form info, etc.

3. UCCX then does a DB dip and fetches widget configuration from the **chatbubble** table on UCCX Informix **(db_cra)**.

4. This configuration is returned back to the user's browser via the **ccppublicapps** API as a JSON response.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]
```

```
3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 4. SM Publicapps:200 OK on the Widget Config.

SM receives the 200 OK for the GET request it sent to CCX to fetch chat widget config.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 5. Client console logs: Response to Client Browser that shows Details in JSON.

1. Here is a sample of JSON response received on the client side as part of all the above operations.

2. This JSON is used to load the bubble chat window through the configured font, problem statement, color etc as received.

   Note: All the above operations are done in order to load bubble chat window so the customer can now fill up his details to start to chat with the contact center. Further flow once customer submits chat details is the same as it was in the classic/ legacy chat flow.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 6. SM Publicapps: Incoming Chat Request.

The incoming chat request is received by SM and social contact is created:

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 7. SM Runtime: Social Contact is Created.

Social contact is created by the SocialMiner engine (Runtime service).

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 8. SM Publicapps: Updates the created SocialContact to the public API.

Social contact is created update is seen in **ccppublicapps** logs.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 9. SM Runtime: Sends the Notification to CCX Webservice API on MADM.

Notification is sent from SM to CCX web services apprising CCX that there is new incoming chat.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 10. UCCX MADM: Webservices shows Incoming Chat Request.

Incoming chat contact is received by CCX on MADM with UCCX_WEBSERVICES enabled to debug.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 11. UCCX MADM: Send JMS message to CCX Engine.

The engine is now informed of the new contact via JMS message bus, i.e UCCX MADM (Administration Webservice API) informs the MIVR (Engine) to queue this contact.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
```

```
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 12. UCCX MIVR:

The Chat Subsystem (SS_CHAT) queues the chat contact and the Route and Queue Subsystem (SS_ROUTEANDQUEUE) allocates the chat to the agent. This is seen in the UCCX Engine logs (MIVR) with SS_CHAT and SS_ROUTEANDQUEUE enabled to debug.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

The agent is sent a notification to accept chat and room is created between user and agent. The flow of all this same as legacy chat so not covering much from logs for this flow.

Step 13. SM Publicapps: Agent Joins and Chats Session has Started.

The chat session has been established once the XMPP tunnel is created. The user's chat widget and the SocialMiner chat gadget (hosted on Finesse) are now connected via XMPP and you can see the presence events being sent and received.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 14. Client Console Logs: Agent Joins Chat Room.

The client-side logs show that the agent joins the chat room. The same is displayed in the chat widget.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

> **Note**: Customer now ends the chat. Here the flow is a bit different as compared to legacy chat as there is a new feature of **post chat rating** added with bubble chat.

Step 15. SM Publicapps: User Leaves the Chat Room.

A user leaves the chat and now the agent is alone in the chat room. This also shows that the user leaves the room.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

> **Note**: If a post chat rating is enabled the SM receives this chat rating once the user submits the post chat rating.

Step 16. Client console logs: User Browser shows Chat Rating Success.

Successful submission of chat rating using a 200 OK received on the user browser.

```
1) GET https://sm-fqdn/ccp/ui/BubbleChat.html?host=sm-fqdn&wid=1&secure=true
[HTTP/1.1 200 OK 0ms]

2) GET https://sm-fqdn/ccp/ui/js/ccp/bubblechat/ccp-chat-components.js
[HTTP/1.1 200 OK 0ms]

3) GET https://sm-fqdn/ccp/bubblechat?wid=1
[HTTP/1.1 200 OK 289ms]
```

Step 17. SM Publicapps: ccppublicapps Receives the Post Chat Rating from the Browser.

Rating feedback received at SM.

```
10.78.91.166: Aug 06 2018 09:33:34.277 +0530: %CCBU__CCPPUBLICAPPS-6-CHAT_FEEDBACK_RECEIVED :
%[ChatFeedback=com.cisco.ccbu.ccp.publicapps.api.chat.ChatFeedback@d82623[rating=4]][Session=
3F8B8C08D7E8144C7B1AD7AF144A4C1E][social_contact_id=0D66B2241000016500235A740A4E5BA6]:
Received chat feedback
```

> **Note**: When chat feedback is received at SM with rating info that information is first saved in socialcontact on SM datastore before it notifies CCX. In case SM datastore is down, submission of chat rating fails with a snippet of "CCPPUBLICAPPS-3-UPDATE_CHAT_SOCIALCONTACT_EXTENSION_FIELD_FAILED"

Step 18. UCCX MIVR: XMPP Update Received from SM.

SM sends XMPP update to CCX apprising it of the rating received from the end user.

10.78.91.166: Aug 06 2018 09:33:34.277 +0530: %CCBU__CCPPUBLICAPPS-6-**CHAT_FEEDBACK_RECEIVED** : %[ChatFeedback=com.cisco.ccbu.ccp.publicapps.api.chat.ChatFeedback@d82623**[rating=4**]][Session=3F8B8C08D7E8144C7B1AD7AF144A4C1E][social_contact_id=0D66B2241000016500235A740A4E5BA6]: Received chat feedback

Step 19. UCCX MIVR: Write Chat Rating to Database.

The submitted chat rating is written to the CCX database and saved in the **textratingdetail** table, which is a newly added table on UCCX 11.6(2).

The main purpose of this table is to store Chat Ratings for reporting purposes.

3723276: Aug 06 09:33:34.299 IST %MIVR-SS_ROUTEANDQUEUE-7-UNK:[Smack Listener Processor (1)] RouteAndQueueSubsystemLogger: com.cisco.wf.subsystems.routeandqueue.aggregator.historical.HistoricalManager : **Writing Historical Record: TRDR**: ContactID=**0D66B2241000016500235A740A4E5BA6**, **rating=4**, ratingTime=java.util.GregorianCalendar[time=1533528214299,areFieldsSet=true, areAllFieldsSet=true,lenient=true,zone=sun.util.calendar.ZoneInfo[id="GMT",offset=0,dstSavings=0,
useDaylight=false,transitions=0,lastRule=null],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1, YEAR=2018,MONTH=7,WEEK_OF_YEAR=32,WEEK_OF_MONTH=2,DAY_OF_MONTH=6,DAY_OF_YEAR=218,DAY_OF_WEEK=2, DAY_OF_WEEK_IN_MONTH=1,AM_PM=0,HOUR=4,HOUR_OF_DAY=4,MINUTE=3,SECOND=34,MILLISECOND=299, ZONE_OFFSET=0,DST_OFFSET=0]