AT&T Service Developer Guide

In-App-Messaging API: Polling

Publication Date: February 2015

Legal Disclaimer

Table of Contents

1 Introduction

This document is intended for developers who are using the In-App Messaging API to manage the user's message store. This document describes how an app can remain in sync with the user's message store.

There are two ways for an app to remain in sync with a user's message store: message polling and notifications. This document describes how to use message polling. It is assumed that the app has successfully created an index for the user, and that the index is initialized. For more information on managing the index, see the In-App Messaging: Index Management document.

2 Polling for Messages

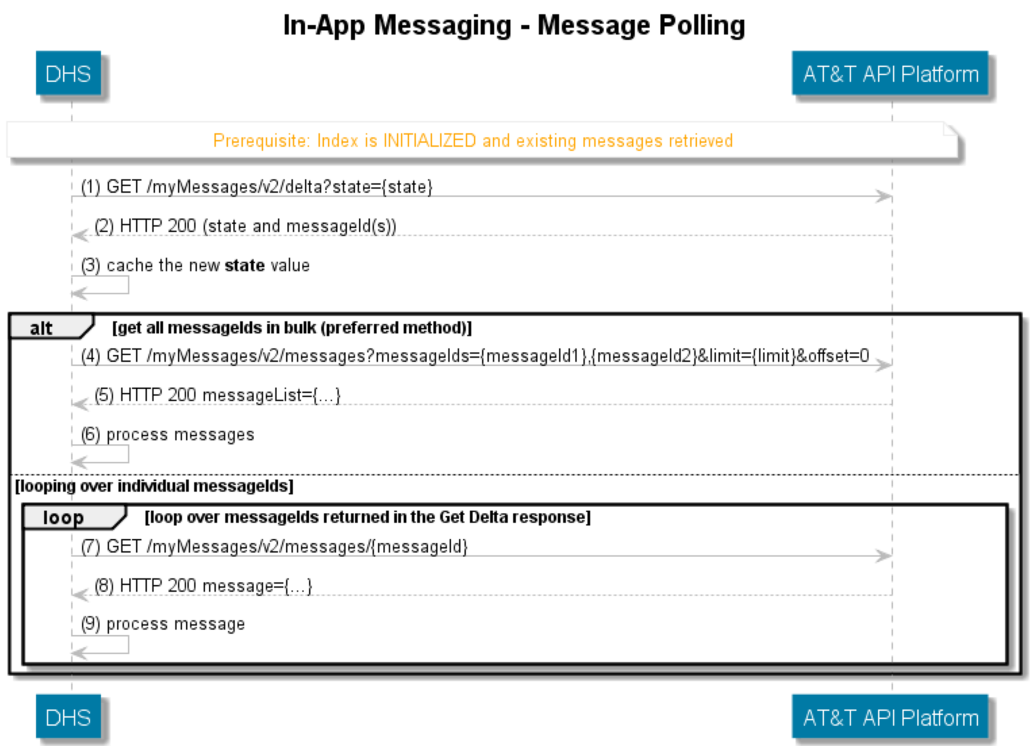
During startup, your app should ensure that the message index is initialized and has already retrieved the stored messages using the Get Message List method of the In-App-Messaging API. After this is done, your app must keep in sync with any updates to the user’s message store. One way to do this is to poll for any deltas that have occurred to the message store after the last message was retrieved. The Get Delta method uses the state parameter from the previous Get Message List, Get Delta, or Get Message Index Info method as an indicator of when the app was last updated. The index is effectively a time stamp of the last time that the app was updated with the message store.

The app should periodically perform the following procedures to ensure that it is kept up to date with the message store.

Note: It is important that the index cache for each user be kept active so that it is up to date with the message store. To do this, the Get Message Index Info method must be invoked within a 24 hour period. For this reason, the app must ensure that the Get Message Index Info method is scheduled to be invoked every 24 hours so that the index cache is kept up to date. For more information, see the In-App Messaging Index Management document.

2.1 Message Polling Call Flow

The following diagram shows the call flow between the Developer App Server and the API Gateway. Notice that the diagram shows the preferred method of retrieving all messages and an optional way lo loop over individual messages.


In-App Messaging - Message Polling
DHS
AT&T API Platform
Prerequisite: Index is INITIALIZED and existing messages retrieved
(1) GET /myMessages/v2/delta?state={state}
(2) HTTP 200 (state and messageId(s))
(3) cache the new **state** value
alt
[get all messageIds in bulk (preferred method)]
(4) GET /myMessages/v2/messages?messageIds={messageId1},{messageId2}&limit={limit}&offset=0
(5) HTTP 200 messageList={...}
(6) process messages
[looping over individual messageIds]
loop
[loop over messageIds returned in the Get Delta response]
(7) GET /myMessages/v2/messages/{messageId}
(8) HTTP 200 message={...}
(9) process message
DHS
AT&T API Platform

2.2 Retrieving Messages

This process shows how to retrieve all messages from the user's method store. This is the preferred method for reading all the messages in the user's message store.

1. Get any updates that have occurred since the last operation. Use the state parameter from the response of a previous call to the Get Message List, Get Message Index Info, or Get Delta method to tell the API platform at what point the app has been in sync with the message store.

```
GET /myMessages/v2/delta?state=1416337804222 HTTP/1.1
Host: api.att.com
content -type: application/json
Accept: application/json
authorization: Bearer abcxyz123456
```

Example 2-1: Get updates

2. If the request was successful, the body of the response contains a new state parameter value as well as messageId parameter values for any messages that have been added, updated, or deleted.

```
HTTP /1.1 200 OK
Content -Type: application/json
Content -Length: 561

{
   "deltaResponse ": {
      "state ": "1416338570349"
      "delta ": [
         {
            "type": "TEXT",
            "adds": [
               {
                  "messageId ": "r888",
                  "isUnread ": true,
                  "isFavorite ": false
               }
            ],
            “deletes”: [],
            “updates”: []
         },
         {
            “type”: “MMS”,
            “adds”: [],
            “deletes”: [],
            “updates”: []
         }
      ]
   }
}
```

Example 2-2: Get updates

3. The app must cache the new state parameter value in order to perform any other Get Delta requests.

   Note: There are two ways to retrieve the actual message content for each of the message identifiers returned in the response. The preferred way is to retrieve all of the messages at once, since this minimizes the number of requests to the API Gateway. However, if the number of message identifiers is large, it is recommended that you split them up into 500 message identifiers per request.

4. Use the Get Message List method to retrieve all message identifiers that are needed to stay in sync with the message store.

   Note: The limit and offset parameters are required to submit the request. However, the actual values are ignored when a value is specified for the messageId parameter.

```
GET /myMessages/v2/messages?messageIds=r888 ,x99&limit=10&offset=0
HTTP/1.1
Host: api.att.com
content -type: application/json
Accept: application/json
authorization: Bearer abcxyz123456
```

Example 2-3: Retrieve all message identifiers

5. If successful, the body of the response contains the content of the messages that were requested. If all of the specified message identifiers are not found, then a response containing an error message is returned. If one or more, but not all, of the specified message identifiers are not found, then the message identifiers that were found are listed in the failedMessages parameter in the successful response.

```
HTTP /1.1 200 OK
Content -Type: application/json
Content -Length: 786

{
   "messageList": {
      "messages": [
         {
            “messageId”: “r888”,
            "from": {
               "value": “14255551212”
             },
             “recipients": "[
                {
                  "value": “12065551212”
                }
             ],
            "timeStamp": "2014 -11 -18 T19 :05:17",
            "text": "hello there",
            "isFavorite": false,
            "isUnread": false,
            "isIncoming": false,
            "type": "TEXT",
            "typeMetaData": {}}
         }
      ],
      "offset": 0,
      "limit": 10,
      "total": 1,
      "state": "1416338570349" ,
      "cacheStatus": "INITIALIZED",
      "failedMessages": [
         “x99”
      ]
   }
}
```

Example 2-4: Content of the messages that were requested

6. Process the messages that are returned.
7. Use the Get Message method to retrieve the content for each message identifier that is needed to stay in sync with the message store.

```
GET /myMessages/v2/messages/r888 HTTP/1.1
Host: api.att.com
content-type: application/json
Accept: application/json
authorization: Bearer abcxyz123456
```

Example 2-5: Retrieve the content for each message identifier

8. If successful, the body of the response contains the content of the message that was requested. If the specified message identifier was not found, then a response containing an error message is returned.

```
HTTP /1.1 200 OK
Content-Type: application/json
Content-Length: 418

{
   "message": {
      “messageId”: “r888”,
      "from": {
         "value": “14255551212”
      },
      “recipients": "[
         {
            "value": “12065551212”
         }
      ],
      "timeStamp": "2014 -11 -18 T19 :05:17",
      "text": "hello there",
      "isFavorite": false,
      "isUnread": false,
      "isIncoming": false,
      "type": "TEXT",
      "typeMetaData": {}}
   }
}
```

Example 2-6: Content of the message that was requested

9. Process the message that is returned, and then loop the process to retrieve the content for other message identifiers.