Qliq Cloud Messaging API Guide

QliqSOFT provides Cloud API for third party applications to send Secure Messages to Qliq users. Following steps need to be performed prior to sending messages:

1. The Application provider must register for a Group Account. Make sure that "My Organization" is selected.

qliq soft	Use Cases API &	Integration Security & HIPAA	About Login Get Started
	Get St	arted	
Verizen ♥ BE2 PM 100K → 1 Mark Zanders ● Pours Surron	QLIQ REGISTRATION Whether you're registering for an individual Qliq account or	Fields marked with an are re I am registering: Myself	equired f • My Organization
O Turkey, 3352 PM He needs attention right He way. Please contact Dr. Mason. (PI be there in 10 min Varian More Carlos at a PM Ok, I will alert the OR and get everything prepped.	you're the group administrator, you've come to the right place. About Qliq	Organization •	
Flag Request Ack □ Oulck message Image: Provide the state of th	Qliq is the only realtime, end- to-end encrypted, secure communication solution for the entire healthcare team. And	Website *	
	best of all, it's Free!	E-Mail Address	

2. Once the group is activated, the Admin of the group must generate an API key in order to use Cloud Messaging.



Once the key is generated, you can start using the Cloud Messaging API from your Application.

Qliq Cloud Messaging API is a RESTful POST command with parameters like below:

https://capi.qliqsoft.com/secure_messages/send?api_key =785bc5068fb1dd505a6ef7f1fb402329&to=demo2@pilot.com&f rom=demo1@pilot.com&subject=Welcome&text=First%20Messa ge

NOTE: Use %20 or + for space in Query String. Particularly for "subject" or "text" field.

If you are trying to send messages longer than 150 characters, User JSON Payload format to sent the message

Request URL: https://capi.gligsoft.com/secure messages/send

Request: POST

Response: JSON Data

Parameter	M/O	Description
api_key	м	This should match the generated API key from the Admin Portal
to	М	This is the email address of the recipient. If the person with email is already a member of the Qliq group, the message is routed to the Qliq App on the user's device. If you want to send a broadcast message to all members of the Group or Subgroup, you should use the QliqID of the Group or Subgroup in "to" field.
from	0	This is the email address of the member. You must create a member with valid email before using the email in the from field. If the from field is not specified, the Group's Point of Contact is used for the From field. Recipient will see the message from this from user.
subject	0	You can specify optional subject for the text. Qliq uses subject field to group the messages into conversation.
priority	0	Set the priority of the message. The options are "fyi", "asap", "urgent".
text	м	This is the actual text of the message.
conversation_id	0	This can be used to group all the messages inside a single conversation.

Cloud API JSON Body Example:

METHOD	SCHEME :// HOST [":" PORT] [PATH	["?" QUERY]]				
POST -	A https://webprod.qliqsoft.com	/secure_me	ssages/	send	🖪 Send	-
	QUERY PARAMETERS			length: 49 bytes		
HEADERS $^{\textcircled{0}}$] ^A		Form 👻	•	BODY ®		Text 🕶
Content-Type :	application/json	×		<pre>1 { 2 "api_key": "shshfsjdhgfjhsdgfhsdhjfsd", 3 "to": "to@cloudapi.net", 4 "from": "from@cloudapi.pilot", 5 "subject": "Test Message", 6 "text": "I would like to see if this works", 7 }</pre>		
				Text JSON XML HTML Enable body evaluation	🛍 lengt	th: 181 bytes

You can also send an attachment with the request as MIME. You can only send one attachment per request. PDF, WORD, EXCEL, PNG/JPG are allowed to be sent as attachments.

Here is an example JSON payload of sending an attachment using Cloud Messaging API

```
{
 "api key": "<from cloud messaging api tab>",
 "from": "<email of the glig user in the group",
 "to": "<email/qliqID/mobile number of the recipient",
 "subject": "String here",
 "text": "String here",
 "attachment": {
   "content type": "image/png",
   "file name": "qliq-qr.png",
   "base64":
"iVBORw0KGgoAAAANSUhEUgAAAPgAAAD4AQMAAAD7H7IIAAAABIBMVEUAAAD///+I2Z/dAAABCUIEQVRYw+
WZQRLDMAwC+f+n6SGSwWmnD4DkEI82J42FsAz+fxDP8Ty7JkkoWsA3Twd6tIE/kdksnrcqjoO0V5q4UkSiiksHOf
n5VT+x3JTQ3i/9TOWnE3qqfvTPWD56MP8MN8IM57u27bBFggI+8PiiK3UFnFrj8obWL5K5LLEpwxrmAm6+cDoij1
9AC5c1sB9RwWH2eGUCOGLRwbcwNj8AfD6QzFUFuD6qj2w++eEtDi6Z2ZzuEKh2YcYpmksToTg2ks91PhqhkFGq
4LdZlDjcU6NgTjsHmiV+T0pyuc3HfGTCMr73A2sVTUEa+O2LfVKYzml3Q5BMyDJkc1hncl1494dQ3n3/+wGvUwfoE
YT/GgAAAABJRU5ErkJggg=="
 }
}
```

If you would like to send multiple attachments, you can use the conversation_id returned for the first attachment, and use it to send the second attachment.

Sending Broadcast Message to Group

Your application can send a broadcast message to an entire group or a subgroup or external group. When you send a broadcast message, everyone in the group will receive the message. If a recipient replies, the reply message goes to the sender's address only.

In Qliq service each group is identified by unique "QliqID". You can find the QliqID of the group you are interested in from the Admin Portal.

Sending multiple messages for within the same conversation

Qliq provides the ability to group multiple messages between a sender and a receiver into a conversation. Typically the conversation is tied to a subject. In healthcare, it can be used to tied to one episode, one patient and/or one problem.

Each conversation is uniquely identified by conversation_id. To send messages that belong to one conversation, first send the message without a conversation_id parameter, the service returns system generated unique conversation_id which should be used while sending subsequent messages. Whenever you would like to change the conversation, send the message without conversation_id. This will create a brand new conversation.



Below is an example of sending Broadcast message to "OnCall" subgroup:

https://capi.qliqsoft.com/secure_messages/send?api_key =785bc5068fb1dd505a6ef7f1fb402329&to=469841783&from=de mol@pilot.com&subject=Welcome&text=First%20Message

Following HTTP Responses you will receive when sending message:

- 202 The API executed successfully and the message is queued.
- 400 The request was incorrect, please make sure that passed arguments are matching format in method's documentation.

- 401 Unauthorized. You attempt to authenticate with an invalid API key.
- 403 Forbidden. Not allowed to send message.
- 404 Not Found. You attempt to request a resource which doesn't exist. Check your URL
- 500 Internal Server Error. Something unexpected happened on our end. Please try again or contact support.

Testing from Command Line

If you have command line tool such as CURL <u>http://curl.haxx.se/</u> you can run the test the message delivery directly before integrating the logic into your app. Below is an sample of testing from command line.

```
$ curl --data
"api_key=02a7eaf78181d2c9f5f6800cfb6a8799&to=827790129&s
ubject=11111111&text=Hello%20again"
https://capi.gligsoft.com/secure_messages/send
$ {"status":"Message queued for
delivery","conversation_id":"65e314a805c3541ac62284177fa
05065","message_id":"cb644eb71a88db57f639db5490626cc6"}
```

If the request is good, you will see a JSON response to the curl command. You will see conversation_id that can be used to send subsequent message to the same conversation and the message_id that can be used to query the message status.

Below is the example of sending message with attachment. Make sure that the attachment is in the current directory.

```
$ curl -X POST -F
"api_key=02a7eaf78181d2c9f5f6800cfb6a8799" -F
"to=827790129" -F "from=<u>test@test.com</u>" -F
"subject=11111111" -F "text=Hello%20again" -F
attachment=@GNYHA.pdf
https://capi.gliqsoft.com/secure_messages/send
$ {"status":"Message queued for
delivery","conversation_id":"65e314a805c3541ac62284177
fa05065","message_id":"cb644eb71a88db57f639db5490626cc
6"}
```

With the example above, the request packet looks like this:

POST /secure_messages_send HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=38516d25820c4a9aad05f1e42cb442f4
Host: capi.qliqsoft.com

--38516d25820c4a9aad05f1e42cb442f4 Content-Disposition: form-data; name="file"; filename="GNYHA.pdf" Content-Type: application/pdf Content-Encoding: base64

H4sICI0fXVQAA3BhZ2UucGRmAAvOz01VCE7MLchJVQhITE8FAAyOwbUQAAAA --38516d25820c4a9aad05f1e42cb442f4--

Notes:

- 1. Make sure the you use HTTPS and POST method when submitting the RESTful API
- 2. Make sure that the subgroup, group, external group QliqID is correct before adding it to "to" field of the request.
- 3. Make sure that the email address is correct in the to field if you are sending message to specific user.

Retrieving Message Status

The Cloud API provides a RESTful API for querying the status of the message. The message_id is mandatory to query for the status of the message along with the api_key.

Request URL: https://capi.qliqsoft.com/secure_messages/message_status

Request: GET

Response: JSON Data

The response to the GET request is a JSON array. The array contains one or more recipient's message status. If the message is single party message, expect to see one element in the array. If the message is a group or broadcast message, you might see more than one element in the array.

If the message is not delivered to any recipients, expect to see array with no elements (ex: [])

Each element will have the email address of the recipient and the delivery time in Unix Epoch Time <u>https://en.wikipedia.org/wiki/Unix_time</u>. It can contain read time as well in Unix Epoch Time.

The Unix Epoch Time can be converted to local time for display or other purposes.

[{ "to": <email_address1>,

"delivered_at": <unix_epoch_time1>,

"read_at": <unix_epoch_time1>

},

]

If the recipient did not read the message, the read_at will not be present.



Receiving Replies from the recipient

The cloud service provides the ability to receive replies to sent messages. In order to receive the messages, the client must implement a Webservice that can receive the messages. Client must implement following to receive messages

- 1. Generate RSA 2048-bit public/private keypair
- 2. Upload the public key to qliqSOFT service
- 3. Implement Webservice
- 4. Decrypt the attachment with the private key

Let's go through details of each step.

Generate Public/Private keypair

You can use crypto library such as OpenSSL to generate public/private keys. Following is a sample code in Ruby to generate these keys.



Upload the public key to qliqSOFT service

qliqSOFT provides a Webservice to upload the public key.

Request URL: <u>https://capi.qliqsoft.com/secure_messages/set_sender_pubkey?sender=john.doe@qliq.com</u>

Request: PUT

Parameters:

Parameter	М/О	Description
api_key	М	This should match the generated API key from the Admin Portal
sender	М	The Email address of the sender. This is the same ID that is used in "from" when sending message. Sender must be a qliq user.

Attachment:

Public Key File

Following HTTP Responses you will receive when sending message:

- 200 The API executed successfully and the key is applied to the user
- 400 The request was incorrect, please make sure that passed arguments are matching format in method's documentation.
- 401 Unauthorized. You attempt to authenticate with an invalid API key.
- 403 Forbidden. Not allowed to set the Public Key. Normally this happens if the user is with the ID does not exist in the system or already in "Active" or "Accepted" state. You can only set key for user who is "Inactive" or in "Pending" state.
- 500 Internal Server Error. Something unexpected happened on our end. Please try again or contact support.



Implement Webservice to receive message

On your web platform go ahead and implement a webservice that can receive POST requests. Copy the URL for the webservice in qliqSOFT Dashboard.

MobileQliq qliqID: 4506319 615 Frederick St. Bluefield2,	83 Admin TX 247011676	• 9722352606 Edit	Admin guide User guide
MobileQliq	Clo	ud Messaging API	
Cloud Messaging API	API URL	https://webprod.qliqsoft.com/secure_messages /send	Click to download API Guide
	API Key	3e25411a6b8487fe83652e55afdec045	This key is used in sending secure messages
	Reply API URL	Regenerate API Key <u>https://hospital.com/secure_message_replies</u>	The replies to messages are posted to this URL

Receive and Decrypt Replies

When recipient of the message replies, your webservice receives a POST on the URL with the reply as an attachment. The Reply is encrypted with the public key that you have set before. You need to you the associated private key to decrypt the message.

Following parameters are sent along with the attachment:

Parameter	M/O	Description
conversation_id	М	This is the same conversation_id that is sent back when you sent the first message.
message_id	М	This is the ID of the reply

Once you decrypt the reply, you will see the reply in following JSON format:

```
"Message": {
```

"Command": "extended-im",

```
"Subject": "text",
```

"Type": "u2u",

"Data": {

"conversationUuid": <conversation id>,

"convesationSubject": <the subject you specified>,

"messageId": <The message id for this reply>,

"text": <content of the message>,

"createdAt": <UTC time at which this message is created>

"attachments": [{ // Empty of there are no attachments

"encryptionMethod": 1, // AES 256

"fileName": <file name>,,

"key": <AES 256 bit Key>,

"mime": <Type of the attachment>,

"size": <size of the file>,

'url": <URL where the encrypted file is located>,

"thumbnail": <For image/video attachments only>

```
}]
}
```

}

Currently only one attachment per message supported. When you see attachment after decrypting the message, you need to download the encrypted attachment and decrypt it with the AES-256 key present in the message.

Following Algorithm shall be used to decrypt the message.

```
1. Decode Base64 attachment
```

- 2. Create chunks of 256 bytes (The RSA key size)
- 3. Decrypt each chunk with the RSA private key
- 4. Join the decrypted chunks

Following is the sample C++ code:

```
BIO *b64 = BIO new(BIO f base64());
        mem = BIO push(b64, mem);
        std::vector<char> encryptedData;
        char inbuf[512];
        int inlen;
        while ((inlen = BIO read(mem, inbuf, sizeof(inbuf))) > 0)
        {
            for (int i = 0; i < inlen; ++i)
                encryptedData.push back(inbuf[i]);
        }
        BIO free all(mem);
        return decryptWithKey(encryptedData, privKey, ok);
    }
     static std::string decryptWithKey(const std::vector<char>& encryptedData,
                                       EVP PKEY *privKey, bool *ok = 0)
      {
        if (privKey == NULL) {
            PJ LOG(1, (THIS FILE, "Cannot decrypt because privKey is NULL"));
            if (ok) {
                *ok = false;
            }
            return "";
        }
        int len = RSA size(privKey->pkey.rsa);
        std::vector<char> buffer(len, '\0');
        std::string decrypted;
        decrypted.reserve(encryptedData.size());
        if (ok)
            *ok = true;
        int totalBytes = encryptedData.size();
        int pos = 0;
        while (pos < totalBytes)</pre>
        {
            int bytesToDecrypt = std::min(len, totalBytes - pos);
            int decrLen = RSA private decrypt(bytesToDecrypt, (const unsigned char
*)encryptedData.data() + pos, (unsigned char *) buffer.data(), privKey->pkey.rsa,
RSA PKCS1 PADDING);
            if (decrLen == -1)
            {
                logAndClearError("Error in RSA private decrypt:");
                if (ok)
                    *ok = false;
                break;
            }
            decrypted.append((const char *)buffer.data(), decrLen);
            pos += bytesToDecrypt;
```

```
}
return decrypted;
}
```

```
Ruby Example Code
```

```
def decrypt_msg private_key, msg
data_chunks = []
key_size = 256
begin
Base64.decode64(msg).bytes.each_slice(key_size) do |slice|
data_chunks << private_key.try(:private_decrypt, slice.pack("C*"))
end
clear_msg = data_chunks.join("")
rescue
clear_msg = failed_to_decrypt_msg
end
clear_msg
end</pre>
```

Update Reply as Read

When the message reply is posted to the web URL you have provided, the message is marked as "delivered". You need to explicitly call "update_message_status" to mark the message as "read".

Request URL: https://capi.qliqsoft.com/secure_messages/update_message_status

Request: PUT

Parameters

Parameter	M/O	Description
api_key	м	This should match the generated API key from the Admin Portal
sender	М	The Email address of the sender. This is the same ID that is used in "from" when sending message for the conversation
conversation_id	м	The id of this conversation
message_id	м	The id of the reply that is posted to your URL
status	М	"opened" to set the status as read "acked" to set the status as acknowledged if the original reply has "Request Acknowledgement" flag set

```
$ curl -X PUT -F
"api_key=02a7eaf78181d2c9f5f6800cfb6a8799" -F
"sender=<u>test@test.com</u>" -F
"conversation_id=jsdfjsdfsjshfsdf" -F
"message_id=skdjfskjfksdhfksjd" -F "status=opened"
<u>https://capi.gligsoft.com/secure_messages/update_mess</u>
age_status
```

\$ OK

Annex A. Sending SMS to Patients/Partners

Cloud Messaging API can be used to send SMS messages to Patients. Patients don't need to download Qliq App to get these messages. Moreover Patients can reply to these messages and the the replies are POSTed back to a webhook or the replies can be queried. QliqSOFT considers Patient's Phone Number as PHI and does not store in the Qliq DB. Only Last 4 digits of the Phone Number are stored for troubleshooting purposes.

Sending SMS

Request URL: https://capi.qliqsoft.com/secure_messages/send

Request: POST

Response: JSON Data

Request Params:

Parameter	M/O	Description
api_key	М	This should match the generated API key from the Admin Portal
to	М	Mobile Number of the Patient/Recipient.
from	0	This is the email address of the member. You must create a member with valid email before using the email in the from field. If the from field is not specified, the Group's Point of Contact is used for the From field. Recipient will see the message from this from user.
subject	0	You can specify optional subject for the text. Qliq uses subject field to group the messages into conversation.
text	М	This is the actual text of the message.
no_phi	М	API user should not send PHI through SMS since it is not HIPAA compliant. Set no_phi to true
reply_posting_url		The webhook implemented by the client to receive replies

Response:

Parameter	M/O	Description
conversation_id	М	Unique for the recipient. There is one conversation id per recipient
message_id	м	Unique ID for each message sent.
statatus	м	Status of the message

Example Request:

🙁 🖨 💷 chrome://	poster - Poster - Mozilla Firefox
Request	
URL:	https://webprod.qliqsoft.com/secure_messages/send
User Auth:	
Timeout (s):	30
Actions	
Content to Se	end Headers Parameters
File:	Browse
Content Typ	e: application/json
Content Opt	ions: Base64 Encode Body from Parameters
{	: "kshdfkshfkhasdfsdfjhdf", 3121234", ender@client.com", "Cloud API", rst Message", true

{

"api_key": "1239c32379ec0725dbb8bc1b87ab9094e",

"from": "notifier@cardinal.com",

"to": "4693110979",

"text": "Your forgot to take a pill",

"subject": "From Cardinal Health",

"no_phi": true

}

Example Response:

{ "status": "Queued for Delivery",

"Conversation_id": "1ksdfjkhskfhsdsfsdfsdhfjks",

"Message_id": "kjshfkskfhsdfsdkjfsfs"

}

Receiving Replies

Client can receive replies in two ways.

- 1. When a reply from the recipient arrives, the reply is POSTed to the Webhook provided by the Client in "reply_posting_url"
- 2. If the Client has a problem with webhook, Client can use the "get_replies" API

Request URL: https://capi.qliqsoft.com/secure_messages/get_replies

Request: GET

Response: JSON Data

Request Params:

Parameter	M/O	Description
api_key	Μ	This should match the generated API key from the Admin Portal
from	0	Mobile Number of the Patient/Recipient. If the Mobile Number is not given all the replies from all recipients are returned
since	0	This is a UTC time. All the replies arrived after "since" are returned. If th "since" is not given all the replies regardless of when they arrived are returned

Response:

Array of JSON objects are returned. If there are no replies, empty array is returned.

Parameter	М/О	Description
conversation_id	Μ	Unique for the recipient. There is one conversation id per recipient
message_id	М	Unique ID for each message sent.
reply	М	Reply Message
received_at	М	Time at which the Reply was received

Example Request:

17					
[8]					
Content to Send Headers Parameters					

Example Response:

😣 🗐 🗐 🛛 Response

GET on http://localhost:3000/secure_messages/get_replies?api_key=89120eb17e186a64b70d41317fe1e697& since=2018-1-17

Status: 200 OK

[{"conversation_id":"6592daad9de5bd323bf3d371e38a5708229c23139e1ca99b2563ea989a2e51ef","message_id":"4f23a82f5c2 8ee1288b78f3f3ada2555","sender_info":"XXXXXX0979","reply":"Test","received_at":"2018-01-17T18:30:09-11:00"}]

Annex B. Sending Secure Message to Non-Qliq User

Cloud Messaging API can be used to send Secure Message to Non-Qliq User. For example, you need to send a patient referral or clinical summary to a provider who is not currently a Qliq User, the API can help systems to generate such messages.

Sending Non-Qliq User

Request URL: https://capi.qliqsoft.com/secure_messages/send

Request: POST

Response: JSON Data

Request Params:

Parameter	M/O	Description
api_key	М	This should match the generated API key from the Admin Portal
to	М	Mobile Number of the Patient/Recipient.
from	0	This is the email address of the member. You must create a member with valid email before using the email in the from field. If the from field is not specified, the Group's Point of Contact is used for the From field. Recipient will see the message from this from user.
subject	0	You can specify optional subject for the text. Qliq uses subject field to group the messages into conversation.
text	М	This is the actual text of the message.
send_non_qliq_us er	М	API User must set this value to "true". Otherwise, the message will not be delivered.
expiry_in_minutes	0	The Secure Message Link generated by this operation has limited life time to protect against sharing. The durection normally 1 day (1440 minutes). API user can customize the expiry with this field. The maximum time the Secure Message Link will be active is 7 days.

Response:

Parameter	М/О	Description
conversation_id	Μ	Unique for the recipient. There is one conversation id per recipient
message_id	М	Unique ID for each message sent.

	statatus	М	Status of the message
--	----------	---	-----------------------

Example Request:

{

"api_key": "1239c32379ec0725dbb8bc1b87ab9094e",

"from": "notifier@cardinal.com",

"to": "4693110979",

"text": "Your forgot to take a pill",

"subject": "From Cardinal Health",

"send_non_qliq_user": true,

"expiry_in_minutes": 60

}

Example Response:

{ "status": "Queued for Delivery",

"Conversation_id": "1ksdfjkhskfhsdsfsdfsdhfjks",

"Message_id": "kjshfkskfhsdfsdkjfsfs"

}