

WICKRIO NODE.JS ADDON INTERFACE

Prepared for: Wickr Inc, Engineering Group

Prepared by: Paul Cushman, Project Lead

Date: July 10, 2018

Version: 4.41.15

# USING WICKRIO WITH NODE.JS

## Objective

This document contains information associated with the installation, configuration, and use of the Node.js interface to the WickrIO client architecture. This interface allows Node.js programs access to the WickrIO API via a Node.js addon. The Node.js addon provides the same set of capabilities as the RESTful API interface.

## Assumptions

This document is intended for Systems Administrators and developers that have a working knowledge of Linux server administration, package installation, API usage, and scripting knowledge.

Wickr staff is available to assist in the deployment and configuration, but for security reasons, at no time should Wickr have access to the actual machine where the deploy is taking place. Screen sharing sessions can be used for troubleshooting.

### Document Overview

The following is a list of sections in this document:

* [Solution Overview](#_SOLUTION_OVERVIEW)
* [Software Installation](#_SOFTWARE_INSTALLATION)
* [Configuration](#_CONFIGURATION)
* [Node.js Addon API Description](#_REST_API_DESCRIPTION)
* [Broadcast Feature Examples](#_BROADCAST_FEATURE_EXAMPLES)
* [Troubleshooting](#_TROUBLESHOOTING)

# SOLUTION OVERVIEW

## WickrIO Components

The components of the WickrIO Node.js addon solution include a WickrIO Docker container, and Node.js sample software. You will have to supply an appropriate host machine to run the WickrIO Docker container and any Node.js software you develop. Included are several Node.js sample solutions that use the WickrIO Node.js addon.

### Host Machine

The WickrIO Docker container should run on any capable host machine. At the time this docuent was written testing was only performed using Ubuntu 16.04 host machines. The WickrIO client software will require access to the file system running on the host machine so that persistent data can be stored.

### WickrIO Docker Container

The WickrIO Docker container contains the WickrIO client(s), WickrIO client service and configuration software. The WickrIO clients are Wickr clients that provide software interfaces to the Wickr client capabilities, instead of via a graphical user interface (GUI). The interface to the Wickr features is through the Node.js addon. This interface provides the ability to send and receive messages, as well as create secure rooms and group conversations. Details of the Messaging API is described in this document.

WARNING: The WickrIO clients implemented in the Docker container are Wickr Bot clients. Visualization of the Wickr Bot clients in the Wickr Clients is under design and development.

### Node.js Scripts

The Node.js scripts are scripts or any software that are implemented by users of the WickrIO Node.js addon interface. Sample scripts using the WickrIO Node.js addon are provided at the end of this document.

# SOFTWARE INSTALLATION

## WickrIO Node.js Installation Components

There are basically three components to installing the WickrIO software:

### Host Machine

The host machine must be capable of running Docker containers. You will have to install appropriate Docker software before you can install the WickrIO Docker Container.

TBD: Add details of setting up Docker and an appropriate Docker account

### WickrIO Docker Container

The WickrIO Docker container includes all of the WickrIO software necessary to configure, run and maintain the WickrIO client(s).

## WickrIO Installation Steps

The following is an example of the components that will be included in the WickrIO distribution, version numbers may change:

* Wickr/bot-cloud-alpha:4.41.15.03
* WickrIO\_node.js\_samples.zip

### Host Machine Setup

The host machine used to run the WickrIO Docker container needs to be setup to be running the appropriate Docker software.   
  
The WickrIO Docker container will also require access to the host file system. Persistent data will be stored in a location on the host’s file system.

The host machine should be secure so that it is impossible for people to tamper with the system.

### WickrIO Docker Container

The WickrIO Docker container can be pulled down to the host machine using the following command (note: the version number may change):

docker pull wickr/bot-cloud-alpha:4.41.15.03

The WickrIO Docker contain includes all of the necessary software to run the WickrIO client(s).

### Node.js addon Software

The Node.js addon software contains the Wickr Node.js addon as well as several samples that can be used to access the WickrIO Node.js interface. These samples can be used as a base to develop your own Wickr applications.

Extract the software to a directory on your host machine.

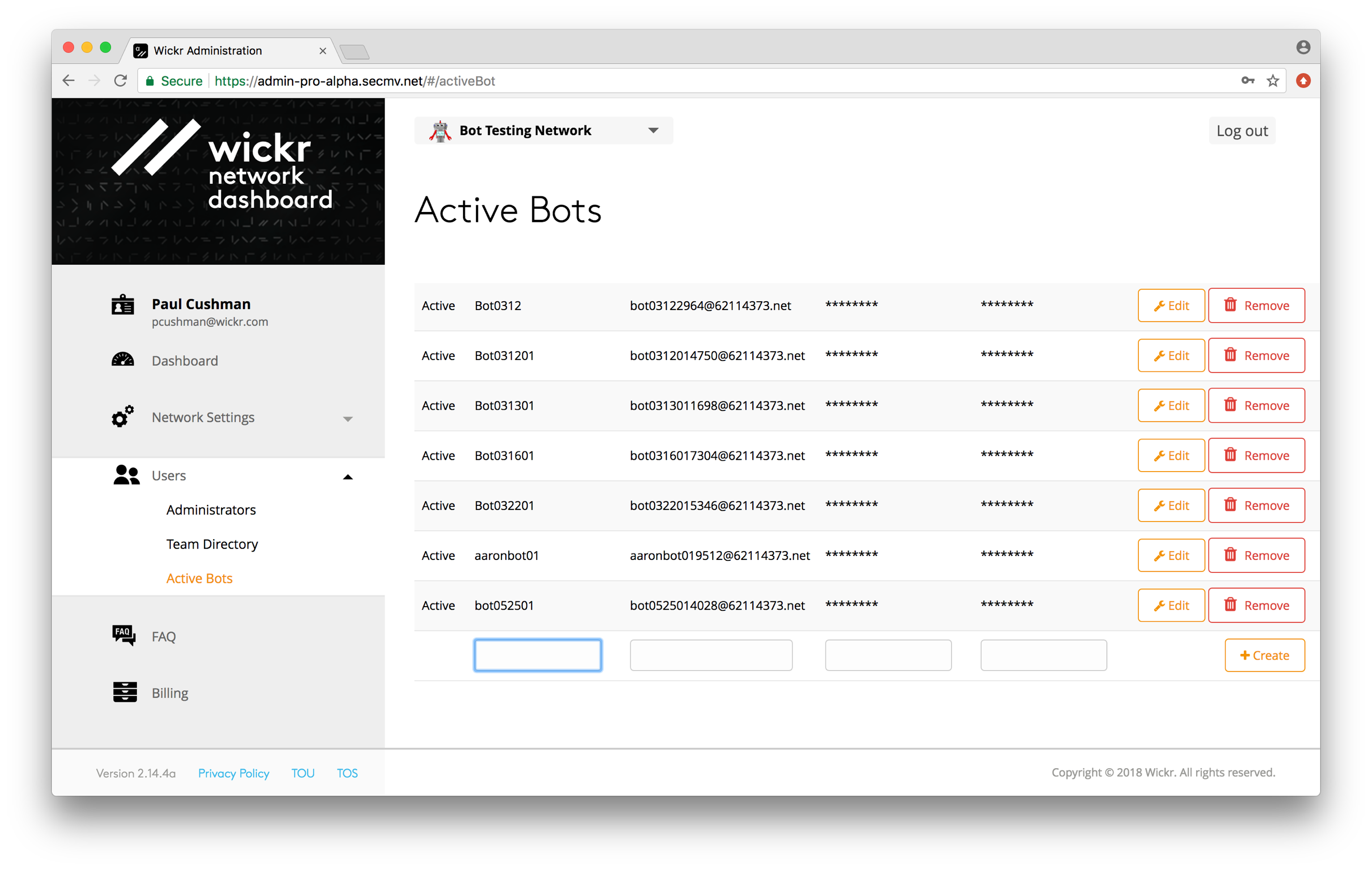
# CONFIGURATION

## WickrIO Configuration Steps

Once the software has been installed there are several things that need to be configured before the WickrIO clients can be used.

### WickrIO Client Creation

The WickrIO Client(s) will need to be created and configured. WickrIO clients are actually Wickr Bot accounts. You will need to create these accounts using the Active Bots screen of the Wickr Admin Console, see below:



When adding a Wickr Bot to the Active Bots screen you will input the Bot display name and the password fields. The console will generate the Bot Username. You will need the Bot username and password fields to configure the Wickr Bot.

## Host Machine Setup

The host machine must have sufficient disk space to support the WickrIO clients running within the WickrIO Docker container.

TBD: Add disk sizing information

The WickrIO clients will need to save persistent data to a location on the host machine. Normally this is located in the /opt/WickrIODebug directory. When running the WickrIO Docker container you will identify where this is located on the host machine, for example:

docker run -v /opt/WickrIODebug:/opt/WickrIODebug -ti wickr/bot-cloud-alpha:4.41.15.03

In this example the host has a directory named /opt/WickrIODebug that is where the WickrIO client(s) running in the Docker container will store persistent data.

## WickrIO Configuration Steps

All the configuration steps for WickrIO are performed from the command interface that is presented when you run the WickrIO Docker container. To start the WickrIO Docker container the following command should be entered, replace the version number with the appropriate version number:

docker run -v /opt/WickrIODebug:/opt/WickrIODebug -ti wickr/bot-cloud-alpha:4.41.15.03

When you run the WickrIO Docker container for the first time there will be no WickrIO clients configured, you will see the following output:

There are no clients currently configured!

Enter command:

When you enter the WickrIO Docker container, after having configured one or more WickrIO clients, the list of currently configured WickrIO clients will be displayed. You will then be prompted to enter a command. You will have the choice of entering the following commands:

* add: add a new client
* delete: deleting a client
* list: see a list of currently created clients
* modify: modify the settings of a client
* pause: pause a running client
* start: start a client
* quit: exits the docker container
* help or ‘?’: display the list of commands

To configure a new WickrIO client run the “add” command. You will be prompted for two fields, the Wickr username and the password. The username is the bot user name generated on the Wickr Admin console, and the password is the password entered on the Wickr Admin console. The “add” command will then provision this WickrIO client and create the appropriate data structures.

Once the client is created you can start the client by running the “start” command with the appropriate index number that you can get from running the “list” command. The “list” command will also display the current status of each WickrIO client.

# NODE.JS ADDON DESCRIPTION

This section will describe the WickrIO Node.js addon and how to use it. There are several samples provided that show how the addon is used.

The WickrIO Node.js addon interface supports a finite set of functions that you can access via your javascript code.

* cmdGetStatistics
* cmdClearStatistics
* cmdGetRooms
* cmdAddRoom
* cmdModifyRoom
* cmdGetRoom
* cmdLeaveRoom
* cmdDeleteRoom
* cmdAddGroupConvo
* cmdDeleteGroupConvo
* cmdGetGroupConvo
* cmdGetGroupConvos
* cmdGetReceivedMessage
* cmdSend1to1Message
* cmdSendRoomMessage

There are several functions that are used to setup and shutdown the addon interface to the WickrIO Client:

* clientInit
* closeClient

The following section will describe each of the APIs.

# WickrIO Node.js addon API Description

## INIT APIs

### clientInit(string clientName)

Before accessing any of the addon interface function you will need to run the clientInit function with the WickriO client name.

### closeClient()

This function will close the currently open client object(s). This should be called when done interacting with the client set in the clientInit() function.

## STATISTICS APIs

### cmdGetStatistics()

This function will retrieve the current statistics on the open client. The statistics are returned in a JSON string. The following is an example of what that returned string would look like:

{

"statistics”: {

"message\_count”: 5,

"pending\_messages”: 0,

"sent”: 7,

"received”: 3,

"sent\_errors”: 1,

"recv\_errors”: 1

}

}

Figure 7: Get statistics response JSON

The following table has a description of each of the statistics returned by this API:

|  |  |
| --- | --- |
| Statistics | Description |
| message\_count | The number of incoming messages that are currently on the WickrIO client. |
| pending\_messages | The number of messages that are to be sent from the specific WickrIO client. |
| sent | The number of messages that have been sent by the WickrIO client. |
| received | The number of messages that the WickrIO client has received. |
| sent\_errors | The number of errors that have occurred while trying to send messages. |
| recv\_errors | The number of errors that occurred while receiving messages. |
| pending\_callback\_messages | The number of messages on the callback message queue. These are messages received by the WickrIO client, that are waiting to be send to a callback process. |
| outbox\_sync | The number of outbox sync messages received. These are messages that were sent by another device for this WickrIO client. |

### cmdClearStatistics()

This API will clear the current statistics that are saved on the client.

## SECURE ROOM APIs

### cmdGetRooms()

This API will return a list of rooms that are known by the WickrIO client. The WickrIO client will respond with a JSON array of secure rooms. The format of the response will look like the following:

{

"rooms”: [

{

"description”: "Room description",

"masters”: [

{ "name" : "username001" }

],

"members”: [

{ "name" : "username001" },

{ "name" : "username002" }

],

"title”: "Room Title",

"ttl”: "7776000",

"bor”: "0",

"vgroupid”: "S00bf0ca3169bb9e7c3eba13b767bd10fcc8f41a3e34e5c54dab8bflkjdfde"

}

]

}

Figure 10: Get secure rooms response JSON

### cmdGetRoom(string vgroupid)

This API will return details of a specific secure room or group conversation. The WickrIO client will respond with a JSON structure containing information for the specified conversation. The format of the response will look like the following:

{

"rooms”: [

{

"description”: "Room description",

"masters”: [

{ "name" : "username001" }

],

"members”: [

{ "name" : "username001" },

{ "name" : "username002" }

],

"title”: "Room Title",

"ttl”: "-1",

"vgroupid”: "S00bf0ca3169bb9e7c3eba13b767bd10fcc8f41a3e34e5c54dab8bflkjdfde"

}

]

}

Figure 11: Get secure room response JSON

### cmdAddRoom(string members[], string moderators[], string title, string desc, string ttl, string bor)

This API will create a new secure room.  The arguments of this request will contain the information associated with the room. The members and moderators arguments are arrays of strings that Identify the members and moderators of the room. The ttl and bor values are optional, but if the bor value is included then the ttl value must also be included.

The response will either be an error with a description of that error or a successful response with the vGroupId of the newly created secure room. The following is an example of a successful response:

{

"vgroupid": "S0b503ae14cc896aad758ce48f63ac5fae0adccd78ef18cde82563c63b2c7761"

}

Figure 9: Create secure room response JSON

### cmdLeaveRoom(string vgroupid)

In order to leave a secure room, you will need to have the vGroupID associated with that room. You can use the get rooms API to get the list of rooms known by the WickrIO client, then determine which room to leave. Also, saving the vGroupID returned from the create room API can be used as well.

The user associated with the WickrIO client will leave the room associated with the specified vGroupID.

### cmdDeleteRoom(string vgroupid)

In order to delete a secure room, you will need to have the vGroupID associated with that room. You can use the get rooms API to get the list of rooms known by the WickrIO client, then determine which room to delete. Also, saving the vGroupID returned from the create room API can be used as well.

### **cmdModifyRoom**(string vgroupid, string members[], string moderators[], string title, string description, string ttl, string bor)

This API is used to modify some of the settings associated with a secure room. The following secure room attributes can be modified using this API:

* TTL
* BOR
* Description
* Title
* Members
* Moderators

## GROUP CONVERSATION APIs

This section describes the APIs associated with group conversations. Using these APIs you can create, get or delete group conversations that the client is a part of.

### cmdAddGroupConvo(string members[], string ttl, string bor)

This API will create a new group conversation.  The response will either be an error with a description of that error or a successful response with the vGroupID of the newly created group conversation. The following is an example of a successful response:

{

"vgroupid": "S0b503ae14cc896aad758ce48f63ac5fae0adccd78ef18cde82563c63b2c7761"

}

Figure 13: Create group conversation response JSON

### cmdGetGroupConvos()

This API will return a list of group conversations that are known by the WickrIO client. The WickrIO client will respond with a JSON array of the group conversations. The format of the response will look like the following:

{

"groupconvos”: [

{

"members”: [

{ "name" : "username001" },

{ "name" : "username002" }

],

"ttl”: "7776000",

"bor”: "0",

"vgroupid”: "S00bf0ca3169bb9e7c3eba13b767bd10fcc8f41a3e34e5c54dab8bflkjdfde"

}

]

}

Figure 14: Get group conversations response JSON

### cmdGetGroupConvo(string vgroupid)

This API will return details of a specific group conversation. The WickrIO will respond with a JSON structure containing information for the specified conversation. The format of the response will look like the following:

{

"rooms”: [

{

"members”: [

{ "name" : "username001" },

{ "name" : "username002" }

],

"vgroupid”: "S00bf0ca3169bb9e7c3eba13b767bd10fcc8f41a3e34e5c54dab8bflkjdfde"

}

]

}

Figure 15: Get group conversation response JSON

### cmdDeleteGroupConvo(string vgroupid)

In order to delete a group conversation, you will need to have the vGroupID associated with that conversation. You can use the get group conversations API to get the list of conversations known by the WickrIO client, then determine which conversation to delete. Also, saving the vGroupID returned from the create group conversation API can be used as well. The group conversation with the same vGroupID will be deleted.

## MESSAGING APIs

This section will describe the messaging APIs. These are APIs that get and send messages.

### cmdGetReceivedMessage()

This API will retrieve the next message waiting to be read. The message will be removed from the client’s database after it has been retrieved.

Currently you will need to explicitly make this call to retrieve a received message. In the future there will be an asynchronous event that will identify when a message was received.

#### Callback Message Format

There are different types of messages that will be delivered to the configured message destination, URL or email callbacks. This section will describe each of these formats. All of the formats are using JSON.

##### One-to-one messages

The following shows a normal one-to-one message format. All text-based messages will have the msgtype of 1000.

{

"id":"3960e020ca4211e799802f2894564caa",

"message":"This is a typical 1:1 message",

"msg\_ts":"1510777143.738976",

"msgtype":1000,

"receiver":"pwcuser001",

"sender":"pwcuser003",

"time":"11/15/17 3:19 PM",

"vgroupid":"fb6e21630c05fde50ae39113c3626018712cf2c374b4a80eba4d28ced9419c07"

}

##### Group and Secure Room Conversation messages

The following shows a normal group and secure room conversation message format. The WickrIO client does not currently track the list of clients associated with group conversations, so the list of destination clients will not be included.

{

"id":"76775de0ca4211e7bddcafd7007db1d1",

"message": "Typical message in a secure room",

"msg\_ts":"1510777246.227505",

"msgtype":1000,

"sender":"pwcuser003",

"time":"11/15/17 3:20 PM",

"vgroupid":"S3042f1bd04491c6f3732a871e27ab516a8d1534cc1e2d25c4e4869ce72e8541"

}

##### File Transfer messages

The following shows the format of a file transfer message. The msgtype for files is 6000. Files received by the WickrIO client will be decrypted and remain on the WickrIO client until removed by your software.

{

"file": {

"filename": "picture.jpeg",

"guid": "AD20D048-9B60-4F32-A691-2D4BE4152E58",

"localfilename": "/opt/WickrIO/clients/compliancebot01/attachments/attachment\_20171116111610865\_picture.jpeg"

},

"id": "91a189c0cae911e79ec4eb19a763225b",

"msg\_ts": "1510849017.756174",

"msgtype": 6000,

"sender": "pwcuser003",

"time": "11/16/17 11:16 AM",

"vgroupid": "S3042f1bd04491c6f3732a871e27ab516a8d1534cc1e2d25c4e4869ce72e8541"

}

As of version 4.35, files sent for screen shots will be identified by a “isscreenshot” key value pair, in the “file” object. This is a Boolean value, where true identifies the file as a screenshot. If the “isscreenshot” key is not found then the file is not a screen shot.

##### Wickr Control messages

Wickr control messages used to setup and configure the conversations will also be sent to the callback destination. These messages are useful to reproduce the conversations, specifically which clients are associated with the specific conversation. Details of these message types will be included later.

### cmdSend1to1Message(string users[], string message, string ttl, string bor)

This API is used to send a message to one or more Wickr clients. The "users" field may contain an array of 1 or more users to send the message to.  The message will be sent to each user on a separate 1-to-1 conversation. So, if the POST message contains 5 users then 5 messages will be sent, using the text from the "message" field.

When sending a message, you can also set the specific burn on read (BOR) value for the message.  The following format shows how to set the BOR value to 10 seconds:

### cmdSendRoomMessage(string vgroupid, string message, string ttl, string bor)

This API is used to send a message to a secure room or group conversation. If you want to send a message to a secure room or a group conversation you will need to get the vGroupID associated with the room. The vGroupID will be returned when you create the room/conversation using the appropriate API. Also, the get rooms API will return a list of known rooms that you can send to, the vGroupID is contained in the response.

### cmdSend1to1Attachment(string users[], bool isURL, string filename, string ttl, string bor)

TBD

### cmdSendRoomAttachment(string vgroupid, bool isURL, string filename, string ttl, string bor)

TBD

# SAMPLES

This section contains some sample javascript source code that implement some basic bots.

## Compliance Bot Sample

This source code implements a very simple compliance bot capability. Basically it will stay on a loop reading frames from the WickrIO addon api.

var addon = require('bindings')('wickrio\_addon');

var fs = require('fs');

module.exports = addon;

console.log(addon.clientInit('aaronbot019512\_62114373.net'));

for (;;) {

var message = addon.cmdGetReceivedMessage();

if(message === "{ }" || message === "" || !message){

continue;

}

else{

console.log(message);

fs.writeFile("receivedMessages.log", message, function(err){

if(err)

return console.log(err);

});

}

}

## Welcome Bot Sample

This source code implements a simple welcome bot, specifically the Welcome Bot part that responds to anyone sending a message to the Welcome Bot client ID:

var addon = require('bindings')('wickrio\_addon');

module.exports = addon;

var responseMessageList = [

"Hey there! Thanks for messaging me! I have a few helpful but random tips I can share in response to your messages, " +

"so please bear with me☺ If you have more questions than I have answers, head to Settings > Support in Wickr Me. " +

"Way to go to protect your privacy!",

"Here is how to find your friends on Wickr Me:\n\n" +

"Go to New Conversations> tap on the contact you want to message if you see them, or start typing their Wickr ID in " +

"the contact field. For group conversations: tap and hold on multiple contacts on Android, or select more than 1 " +

"participants in iOS and desktop.",

"Here is how to set expiration on your messages:\n\n" +

"Expiration is the max time your message will live. Burn-On-Read (BOR) is how long your message will live once the " +

"recipient(s) has seen it. You can change both by tapping on the (i) next to any conversation name, at the top of " +

"your screen.",

"Video Verification:\n" +

"If you'd like to be sure you are talking to the right person, you can send them a verification request. Tap on a user's " +

"avatar>then on the key icon. You can read more on why key verification is cool for your privacy on our blog: " +

"https://medium.com/cryptoblog/key-verification-in-secure-messaging-bd93a1bf3d40",

"How to invite friends to join you on Wickr Me:\n\n" +

"You can invite your friends by going to your Settings > Contacts > then tap Invite in the top right corner of your app. " +

"Chose to invite friends by either sending a text or email from your device. They will need to download the app, and " +

"create a Wickr ID to communicate with you here.\n\n" +

"We never store your device contacts on our servers. All invitations are generated locally on your device, without " +

"sharing any information with us.",

"Sending files on Wickr Me:\n\n" +

"You can now send photos, videos, and other files via Wickr Me, up to 10 MB. This feature supports collaboration " +

"and maximum data hygiene for you and the contacts you TRUST. If you do not trust the person you’re talking to, do " +

"not open files coming from them or send them photos/files you do not want to be saved. Stay safe!",

"Verification\n\n" +

"You’ll notice an orange dot around your contacts’ avatars – that means you have not yet verified them.\n\n" +

"You don’t have to, but in case you want to make sure you are talking to the right person, send them a key video " +

"verification request to establish trust between your Wickr Me accounts.\n\n" +

"Check out our blog on this: https://medium.com/cryptoblog/key-verification-in-secure-messaging-bd93a1bf3d40",

"Passwords\n\n" +

"Important to know: there is no password reset on Wickr Me – we don't know who you are which prevents us from " +

"verifying you to reset your password.\n\n" +

"So please remember your password☺",

"Client Support\n\n" +

"You can use Wickr Me on mobile or desktop to stay in touch with your friends across all your devices.\n\n" +

"Go to www.me-download.wickr.com to download and install on your other devices.",

"Privacy\n\n" +

"We built Wickr Me to provide private communications to everyone.\n" +

"We take your privacy & security very seriously, learn more: www.wickr.com/security.\n\n" +

"Source code https://github.com/WickrInc/wickr-crypto-c. FAQ www.wickr.com/faq"

];

var wickrUsers = [];

console.log(addon.clientInit('aaronbot019512\_62114373.net'));

welcomeBot();

function welcomeBot() {

for (;;) {

var message = addon.cmdGetReceivedMessage();

if (message === "{ }" || message === "" || !message) {

continue;

} else {

var parsedData = JSON.parse(message);

var wickrID = [parsedData.sender];

var location = find(wickrID);

if (location === -1) {

wickrUsers.push({

wickrID: wickrID,

index: 0

});

}

var current = getIndex(wickrID);

if (current > 9) {

location = find(wickrID);

wickrUsers[location].index = 0;

}

current = getIndex(wickrID);

if (current <= 9 && current != -1) {

addon.cmdSend1to1Message(wickrID, responseMessageList[current], '100', '60');

location = find(wickrID);

wickrUsers[location].index = current + 1;

}

}

}

}

function find(wickrID) {

for (var i = 0; i < wickrUsers.length; i++) {

if (wickrUsers[i].wickrID[0].localeCompare(wickrID[0]) === 0)

return i;

}

return -1;

}

function getIndex(wickrID) {

for (var i = 0; i < wickrUsers.length; i++) {

if (wickrUsers[i].wickrID[0] === wickrID[0]) {

return wickrUsers[i].index;

}

}

}

This code will maintain a place holder for each user that sends to the Welcome bot, so that if that user sends another message to the Welcome bot then the next welcome message will be sent back to that user.

# TROUBLESHOOTING

## Troubleshooting WickrIO Components

This section will describe some possible issues you may run into while using the WickrIO client and the associated services.

### WickrIO Client does not start

First you need to determine if the WickrIO client is running or not. To do so you can run the ps command to see that the client process is running. The following should return an entry for each WickrIO client that is running:

ps -aef | grep wickrio\_bot

If nothing is returned from that command, then verify that the background service is running. Using ps, the following command should return an entry for the background service:

ps -aef | grep WickrIOSvr

If nothing is returned then use the WickrIOConsoleCmd to start the background service, this is described in a section above. If you cannot get the background service to start, then contact Wickr support.

If the background service process is running but there is no process running for the WickrIO client, then use the WickrIOConsoleCmd program to help diagnose the problem.

1. At the top level of the WickrIOConsoleCmd program enter the “client” command.
2. Enter the “list” command to see the list of WickrIO clients.
   1. If the WickrIO client’s state is “Paused”, then use the “start” command to start the client.
   2. If the WickrIO client’s state is “Running”, and there was no associated process running, then check the output file for the background services (described later) to see if the background service is having a problem starting the client. If it looks like the service is not trying to start the client, then restarting the background service should fix the problem. To do so, go to the “server” section of the WickrIOConsoleCmd program and stop and start the server. If this does not fix the issue the contact Wickr support for further assistance.
   3. If the WickrIO client’s state is “Down”, this is typically related to the background service not running. As per the previous step, go to the “server” section of the WickrIOConsoleCmd program and stop and start the server. If this does not fix the issue the contact Wickr support for further assistance.

Further diagnosis of problems with the WickrIO client or background service should be done with the help of the Wickr support team.

### Log and Output Files

The WickrIO Client and the background service will generate log and output files that can be used to determine possible issues. These files should be sent to Wickr Support to allow them to diagnose any issues that cannot be easily fixed. Output and log files will only be allowed to reach a certain file size. Once that size is reached a new file will be created. The maximum number of files saved to disk should only be 5.

The WickrIO Client log and output files are located in the following location:

/opt/WickrIO/clients/<client name>/logs

There are several files found in that directory. The file with the “.output” extension contains the most information and is useful in diagnosing issues with the WickrIO client.

The background service will write log and output files to the following location:

/opt/WickrIO/logs

The name of the background service files will start with WickrIOSvr. These files will also be limited in size and number of saved files. The background service output file can be used to diagnose any possible issues with starting a WickrIO client.