

WICKRIO NODE.JS ADDON API DEFINITION

Prepared for: Wickr Inc, Engineering Group

Prepared by: Paul Cushman, Project Lead

Date: September 24, 2018

Version: 4.54.16

# NODE.JS ADDON DESCRIPTION

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

If you extract the Wickr Node.js addon sample distribution you will see a README file and several directories: addon, exports, and samples. The addon and exports directories are used to build the Wickr Node.js addon, which will be used by each of the samples found in the samples directory. The README file contains specific Instructions to build each of these, as well as any software requirements needed to do so. The samples show an Instance of the welcome bot and the compliance bot, as well as two other programs to test the different parts of the WickrIO Node.js addon Interface.

The WickrIO Node.js addon interface supports a finite set of functions that you can access via your javascript code. These APIs are consistent with the APIs provided by the HTTP REST API.

* 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

Interaction with the WickrIO Node.js addon basically looks like the following sequence of events:

1. Initialize the WickrIO Node.js addon interface. This is done by calling the clientInit() function, and supplying the name of the WickrIO client that is going to be used.
2. Interact with the WickrIO client by calling the appropriate command functions (start with the “cmd” string).
3. When your program is complete then call the closeClient() function to stop processing.

## 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.

### cmdStartAsyncRecvMessages(function callback)

This API is used to start the asynchronous reception of incoming Wickr messages. The input argument is a javascript function callback. This function will be called for each message that is received. The format of the messages is the same format as described in the polling message receive function section.

### cmdStopAsyncRecvMessages()

This API is called when you want to stop the asynchronous reception of incoming messages.

TBD

### 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[], string filename, string displayName, string ttl, string bor)

This API is used to send a file to a set of 1to1 conversations. Each user in the users array will be associated with a 1to1 conversation. The filename refers to a file on the system that is reachable by the running bot client.

### cmdSendRoomAttachment(string vgroupid, string filename, string displayName, string ttl, string bor)

This API is used to send a file to a Secure Room or Group conversation. The filename refers to a file on the system that is reachable by the running bot client.

# NODE.JS SAMPLES

There are several samples available that show how to use the WickIO Node.js interface. Included in these samples are programs to perform file\_bot, welcome\_bot and compliance\_bot functionality. Please get the samples package that includes documentation on how to use the samples.

# CUSTOM INTEGRATIONS

The WickrIO system allows the user to create their own custom integrations. A custom integration is software that uses the WickrIO Node.js API, with the goal of being installed and used by one or more WickrIO clients. The custom integration will be treated like one of the WickrIO integrations supplied with the WickrIO installation.

Custom integrations are configured and maintained within the WickrIO system using the set of “integration” commands. These commands allow you to, add, delete, upgrade and list custom integrations. To access the integration commands, enter the “integration” command at the top-level command prompt.

The software used for a custom integration will end up residing with the WickrIO client’s specific directory. This will allow you to configure any custom integration to the specific WickrIO client. The custom integration software must also provide several executable scripts so that the WickrIO service software can install, configure, start and stop the custom integration. A version file can also be used to maintain a version number for the custom integration. Format of the version file is X.Y.Z, where X, Y and Z are 1 to 3-digit number values (i.e. 1.2.3).

## Integration Commands

The integration commands are used to create and maintain the custom integrations. Any WickrIO client that you add/modify can use the supplied integrations or custom integrations. This section will describe the integration commands. The integrations commands can be accessed via the top level using the “integration” command.

### Add Custom Integration: add

The add command will create a new custom integration. You will be prompted for the name of the custom integration, the location of the custom integrations software.tar.gz file, and optionally a version number. The name must be unique across all integrations.

If you supply a version number, it will be used to identify when future versions of the custom integration are installed and identify if existing clients using the custom integration are using an old version.

See the Software.tar.gz Contents section for a list of files that MUST be included in the software.tar.gz file.

### Delete Custom Integration: delete <index>

The delete command is used to delete a custom integration. The index identifies which custom integration is to be deleted from the list of custom integrations. You should remove/update any existing WickrIO clients that are using the custom integration that is to be deleted.

### List Current Custom Integrations: list

The list command will display a list of the custom integrations that are currently installed/configured on the WickrIO system.

### Update Custom Integration: update <index>

The update command is used to update the custom integration software. Like the add command, the update command will prompt for the software.tar.gz file location, and a version number. Any client that is using the custom integration will not be updated automatically. If you supply version numbers and they are different than the existing installation, you will see an indication of this when listing the clients. For example:

Enter command:

list

Current list of clients:

client[0] pwcbot0028083@85022943.net, State=Paused

client[1] testbot019045@62114373.net, State=Paused, Integration=test\_bot (Needs Upgrade!)

Enter command:

You can see that the second client in the list needs an upgrade of the test\_bot custom integration.

## Custom Integration Software Contents (software.tar.gz)

The software.tar.gz file associated with the custom integration will include all the software associated with the custom integration and several executable scripts. Each script will be run from the directory where the script is located. The following are descriptions of each of these scripts, their purpose and any arguments that are passed to the scripts:

### Installation Script: install.sh

This script is used to install any software that is required to execute the custom integration. If all the necessary software is included in the software.tar.gz file, then this script will do nothing. This script is required.

### Configuration Script: configure.sh

Used to configure the custom integration software, specific to the WickrIO client. This script is used to configure any client specific information. Examples may include configuration of user credentials necessary to log into a system, not including Wickr client credentials. This script is required.

Future versions of WickrIO clients will provide a way to save any sensitive information in an encrypted server.

### Start Script: start.sh

This script is used to start the specific instance of the custom integration. Specific instance being the instance associate with a specific WickrIO client. This is a required script.

### Stop Script: stop.sh

This script is used to stop the specific instance of the custom integration. This is a required script.

### Upgrade Script: upgrade.sh

This script is used to upgrade the custom integration software. The goal of this script is to perform and specific operations that need to be done when upgrading the custom integration software. This would consist of saving any configuration information from the current installation, associated with a specific WickrIO client. The script is passed two arguments, the first one is the directory location of the existing installation, the second argument is the directory of the new software installation. If there is an upgrade.sh script supplied, it is up to the script to make the appropriate changes with the result being that the existing installation directory will have been updated and ready to move on to the next step. NOTE: when upgrading custom integration software, the upgrade.sh script in the NEW software.tar.gz will be run, not the upgrade.sh from the previous version.

This is an optional script. If the upgrade.sh is NOT part of the software.tar.gz then the default operation will be to remove the old software installation, specific to a WickrIO client, and replace it with the new version of the software.

After the upgrade.sh script has completed, or the default operation of replacing the software, then the install.sh and then the configure.sh scripts will be called.

# 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

The console of the WickrIO Docker provides commands that you can use to help diagnose the client issue.

1. 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 this does not fix the issue the contact Wickr support for further assistance.

Further diagnosis of problems with the WickrIO client or other Wickr software 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, depending on how you setup the shared files:

/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.