Zipwhip API Developer Reference
Message Send & Receive

Version 1.0 , Dec. '15

CONTENTS

Document Summary

This document describes the two primary functions of the Zipwhip API: message/send and message/receive. The message/send description also includes how to send a message with an attached image (MMS), including how to retrieve the message to be attached.

1.1 Intended Audience

The intended audience for this document is:

- Software Developers who are responsible for using Zipwhip's text-messaging functionality to add text-messaging functionality to their product, system, or network.

1.2 Introduction

The Zipwhip API message/send and message/receive functions described in this document are the same functions that are used in Zipwhip's Web, Desktop, and Mobile Applications. This means that you can add any/all of the features used in our Web, Desktop, or Mobile Apps to your organization's product, system, or network.

Zipwhip Authentication

Authentication is a primary element of using the Zipwhip API. Authentication is more than obtaining a session key.

The first step in authentication is to set-up a Zipwhip account. To set-up an account, contact sales@zipwhip.com.
If you already have an account but cannot obtain session keys, then contact api@zipwhip.com for support.

When you set-up an account, you can then obtain one or more session keys, depending on how you plan to configure your sessions.

2.1 Obtain A session key

Zipwhip's API uses a session key to perform all actions. This session key is a GUID representation of an authenticated user. You can use either a single-user or multi-user session key.

Zipwhip sessionKeys do not expire. The session keys can be obtained once and then used for all future actions. The keys are specific to each phone number; they are not universal for all phone numbers under your control.

To obtain a single-user session key, you perform a user/login web-call for the account:

```
$ curl -X POST https://api.zipwhip.com/user/login \
    -d username=[phone number] \
    -d password=[account password]
{
  "success":true,
  "response":"3d0f1dde-aaff-4ce8-b61a-af212a860abc:123456789"
}
```

In this case, the response is the session key for the account.

To obtain a single-user session key, you perform a user/login web-call for the account:

```
curl -X POST https://api.zipwhip.com/user/login \
  --data-urlencode username=test@8445551212 \
  --data-urlencode password=greatpassword
```

In this case, the response is the session key for the account.
After you obtain either type of session key, you can use the key to send/receive a text message.

Send A Text Message

There are three Send-Message scenarios:

- Send a text message to a single contact
- Send a text message to multiple contacts
- Send a scheduled text message

3.1 Parameters

The Zipwhip API uses input and output parameters to define message elements.

3.1.1 Input Parameters

The following tables include the input parameters for message/send, message/send to multiple recipients, and message/send a scheduled message.

Zipwhip API Input Parameters: Message/Send To Single Recipient

Parameter Name	Value	Type	Required
`session`	The GUID representation of a logged-in user.	String	True
`contacts`	The string representation of a phone number. • US Domestic numbers are full, 10-digit numbers. • International numbers must be in E.164 format.	String	True
`body`	The message to be sent. The message body can be a maximum of 250 bytes.	String	True

Zipwhip API Input Parameters: Message/Send To Multiple Recipients

Parameter Name	Value	Type	Required
`session`	The GUID representation of a logged-in user.	String	True
`contacts`	The string representation of a phone number. • US Domestic numbers are full, 10-digit numbers. • International numbers must be in E.164 format. Note: You can have 15 individual contacts as parameters. For example: Message/send?session={}&contacts=asdf&contacts=asdfasdf	String	True
`body`	The message to be sent. The message body can be a maximum of 250 bytes.	String	True

Zipwhip API Input Parameters: Message/Send Scheduled

Parameter Name	Value	Type	Required
`session`	The GUID representation of a logged-in user.	String	True
`contacts`	The string representation of a phone number. • US Domestic numbers are full, 10-digit numbers. • International numbers must be in E.164 format.	String	True
`body`	The message to be sent. The message body can be a maximum of 250 bytes.	String	True
`scheduledDate`	Time/Date of desired delivery; must be in milliseconds and converted to Pacific Time.	String	False

3.1.2 Output Parameters

This section includes output parameters for both message/send success and failure.

Message/send success response


```
{
    "response"  {
        "fingerprint": "4233621183",
        "root"  "6648805131753308l6",
        "tokens" [
            {
                "contact": 3460624702,
                "device": 304397802,
                "fingerprint": "4233621183",
                "message": "6648805131753308l6"
            }
        ]
    },
    "success": true
}
```


Technically, you can ignore the fingerprint and root elements of the response object. You can use the elements in the tokens array instead of the fingerprint and root elements. In all cases, there is a single token in the token array.

Zipwhip API Output Parameters: Message/Send Success

Parameter Name	Value	Type	Required
`name`	The ContactId of the message recipient.	String	True
`device`	UserId of the message sender.	String	True
`fingerprint`	Is the ConversationId, which is an ID assigned based off of recipients of the message.	String	True
`message`	The ID of the message that was just sent.	String	True
`success`	The success/failure of the web request	String	True

Message/send failure response

The following is an example of a failure response to a message/send:

```
{
    "response": {
        "code": -703,
        "desc": "Bad arguments"
    },
    "success": false
}
```

Zipwhip API Output Parameters: Message/Send Failure

Parameter Name	Value	Type	Required
code	The Zipwhip error code.	String	True
desc	Description of the error.	String	True
success	The success/failure of the web request. True= success; False = failure.	String	False

Message/send success response

The following is an example of a failure response to a message/send:

```
{
        "response":
        {
                "id":664883126574915584,
                "status":"queued"
        },
        "success":true
}
```

Zipwhip API Output Parameters: Message/Send Success

Parameter Name	Value	Type	Required
code	The Zipwhip error code.	String	True
status	The status of the message delivery.	String	True
success	The success/failure of the web request. True= success; False = failure.	String	True

Message/send failure response

The following is an example of a failure response to a message/send:

```
{
       "success":false,
       "errorDesc": "Invalid recipients. Invalid phone numbers: +1206856.
                    Phone numbers should be in e164 format."
}
```

Zipwhip API Output Parameters: Message/Send Failure

Parameter Name	Value	Type	Required
`success`	The success/failure of the web request. True= success; False = failure.	String	False
`errorDesc`	Description of the error that caused failure.	String	True

Failure Response: File too big

The following is an example of a message/send failure because the size of file being sent exceeds the maximum allowed:

```
<html>
      <head><title>413 Request Entity Too Large</title></head>
      <body bgcolor="white">
            <center><h1>413 Request Entity Too Large</h1></center>
            <hr><center>nginx</center>
      </body>
</html>
```

3.2 Status Code

Zipwhip uses status codes to indicate the success, failure, or progress of an event. The success codes are assigned when message delivery succeeds to both the recipient and database. Failure codes are assigned when the message cannot be sent because of an internal or external problem. Progress messages are assigned when the message is in-process or queued. If you use Zipwhip's API Web Hooks, then there are specific progress messages that give you the ability to track the message through each step of the delivery process.

3.2.1 Status Codes: Message Not Sent

If an error occurs when the text message is sent, then an error message appears. The current list of error codes is included in the table below.

Code	Description	State	Retry
-1234	Recipient sent STOP message Resolution = Recipient must send UNSTOP message.	Failure	No
-855	Internal Error	Failure	Yes
-801	Spam Protection Resolution = Contact api@zipwhip.com to have spam flag removed.	Failure	No
-704	Internal Error	Failure	No
-700	Internal Error	Failure	No
-394	Invalid Parameter Length Resolution = May no longer be valid. May be replaced by -703/Bad Argument, which is caused by incorrect/invalid input parameter.	Failure	No
-124	Unable to determine carrier	Failure	No
-1	Unknown Error	Unknown	Yes

3.2.2 Status Codes: Message Progress

When you use Web Hooks, the message-progress Status Codes give you the ability to track the progress of events as they occur. These specific codes are labeled as transient. Consequently, they are unlikely to be visible in any message/list API calls.

Code	Description	State	Final State
0	Delivered – No Confirmation	Success	Yes
1	In Progress	Transient	No
2	Queued	Transient	No
3	Queued	Transient	No
4	Delivered- Confirmed	Success	Yes
5	Error- Confirmed	Failure	Yes
6	Delivered – Confirmed by Delivery Receipt	Success	Yes
7	Error – Confirmed by Delivery Receipt	Failure	Yes

3.3 Send To One Individual Contact

The first and most basic scenario is sending a single text message to a single, individual contact. In this scenario, one unscheduled message is sent to one recipient. The recipient can be either an existing contact or a contact not yet included in the Contact List. The only requirement for the contact is that their telephone number must be valid and text-enabled.

```
$ curl https://api.zipwhip.com/message/send \
    -d session=3d0f1dde-aaff-4ce8-b61a-af212a860abc:123456789
    -d --data-urlencode contacts='+18559479447'
    -d --data-urlencode body='Hello World, from Zipwhip!'
{
  "response"
    "fingerprint": "4236521183",
    "root": "3275590933633723008",
    "tokens": [
      {
        "contact": 1989548603,
        "device": 309626613,
        "fingerprint": "42336654183",
        "message": "3275456789633723008"
      }
    ]
  },
  "success": true
}
```

3.4 Send To Multiple Individual Contacts

The second scenario is sending a single text message to multiple, individual contacts. In this scenario, one unscheduled message is sent to multiple contacts. However, the contacts/recipients are individuals and not a Contact Group.

```
$ curl -X POST https://api.zipwhip.com/message/send
-d session=8762f385-22a0-4675-9aaf-949602f3edde:304397802
--data-urlencode  contacts='+12068597896'
--data-urlencode contacts='+14257772300' -d body=asdf
{
    "success": true,
    "response": {
        "fingerprint": "227798380",
        "root": "666345808406650880",
        "tokens": [
            {
                "message": "666345808406650881",
                "fingerprint": "4233621183",
                "device": 304397802,
                "contact": 3460624702
            },
            {
                "message": "666345808406650882",
                "fingerprint": "3235647327",
                "device": 304397802,
                "contact": 3461841602
            }
        ]
```

3.5 Send A Scheduled Message

The third scenario is setting-up and sending a single, scheduled text message that is sent to an individual contact.

```
$ curl -X POST https://api.zipwhip.com/message/send \
    -d session=3d0f1dde-aaff-4ce8-b61a-af212a860abc:123456789
    -d --data-urlencode contacts='+18559479447'
    -d --data-urlencode body='Hello World, from Zipwhip!'
    -d scheduledDate=14477075270000

{
  "response"  {
    "fingerprint": "4236521183",
    "root": "327559093363723008",
    "tokens": [
      {
        "contact"  : 1989548603,
        "device"  : 309626613,
        "fingerprint": "42336654183",
        "message": "327545678963723008"
      }
    ]
  },
  "success"  : true
}
```

Receive a Text Message

Zipwhip uses Web Hooks to push state-changes to your account. Web Hooks allow Server-to-Server communication of new events, without the requirement of a persistent connection. Web Hooks give you the ability to receive and track real-time events within Zipwhip.

4.1 Retry Logic

The Zipwhip API uses Web Hooks to manage and execute the text-message retry processes. There are two processes: a retry process for single or intermittent delivery failure and a retry process when all messages fail to be delivered for one hour.

4.1.1 Single Message/Intermittent Message Delivery Failure

In the case of delivery failure for a single message or intermittent messages, Zipwhip defines the message delivery as failed when one of the following conditions is met:

- If you receive a status code that is a not a 200-level code, then message delivery has failed.
- If your server takes longer than seven (7) seconds to respond to a delivery, then message delivery has failed.

When message delivery fails, the message is sent to the retry queue. Zipwhip uses an exponential back-off algorithm with a ten (10) minute ceiling to try to resend the message. The first retry occurs when the message has been in the queue for ten (10) seconds. If message delivery does not succeed within the 10-minute ceiling, then Zipwhip can continue to retry delivery for a maximum of 72 hours.

4.1.2 All Message Delivery Failure

If all messages fail to be delivered for one (1) hour, then Zipwhip assumes that the host is down and stops trying to send messages. Zipwhip then waits ten (10) minutes and then tries to send a single, relevant message. If the single message is delivered successfully, then Zipwhip queues the backlog of unsent messages and starts sending. If the single message is not delivered successfully, then Zipwhip continues sending a single message every 10 minutes. Zipwhip can continue to retry delivery for a maximum of 72 hours.

4.2 Security

It is important that you set-up Web Hooks in a manner that maintains privacy.

- We recommend that the destination address is HTTPS.
- Zipwhip supports TLS V1.0 and greater. Zipwhip does not support SSL V3.0 and older.

4.3 Access Control List, ACL

If you have strict firewall rules, or would prefer to lock down your Web Hook end-points to our IP space, then use the ranges provided below.

69.46.44.0/24
208.69.95.64/26

4.4 Events

As the implementer, you can choose which events you would like to listen for.

- Delete
- Progress
- Read
- Receive
- Send

4.4.1 Example Payload

```
POST /message/receive HTTP/1.1
Host: http://www.yoururl.com/zipwhip/api/receive
Content-Length: 581
Content-Type: application/json; charset=UTF-8

{
  "body":"Thanks for texting, this is an auto reply!",
  "bodySize":42,
  "visible":true,
  "hasAttachment":false,
  "dateRead":null,
  "bcc":null,
  "finalDestination":"4257772300",
  "messageType":"MO",
  "deleted":false,
  "statusCode":4,
  "id":634151298329219072,
  "scheduledDate":null,
  "fingerprint":"132131532",
  "messageTransport":9,
  "contactId":3382213402,
  "address":"ptn:/4257772222",
  "read"
  "dateCreated":"2015-08-19T16:53:45-07:00",
  "dateDeleted":null,
  "dateDelivered":null,
  "cc":null,
  "finalSource":"4257772222",
  "deviceId":299538202
}
```

4.5 Install Web Hooks

Install a Web Hook onto the text-enabled phone number. When an event occurs the details of the message will be delivered to the URL specified. During Web Hooks installation, the URL parameter should be unique for each event. The payload of Web Hooks does not identify the event being sent.

4.5.1 Parameters

Parameter	Description	Required
`session`	Session is the response of user/login.	True
`type`	For now, the available options is 'message'	True
`event`	Events listed above. i.e. send, receive, etc.	True
`url`	The destination when the event fires.	True
`method`	HTTP Method used for Web Hook. i.e. POST, PUT, GET	False

4.5.2 Example Request

```
$ curl -X POST https://api.zipwhip.com/webhook/add \
       -d session=[sessionKey] \
       -d type=message \
       -d event=receive \
       -d url=https://test.zipwhip.com/message/receive \
       -d method=POST
```

4.5.3 Example Response


```
  {
    "success": true,
    "response"
          {
            "webhookId": 687557714,
            "type": "message",
            "event": "receive",
            "url": "https://test.zipwhip.com/message/receive",
            "method": "POST"
          }
      ]
  }
```

4.6 List Web Hooks

List the currently installed Web Hooks for the account.

4.6.1 Parameters

Parameter	Description	Required
`session`	Session is the response of user/login.	True

4.6.2 Example Request

```
$ curl -G https://api.zipwhip.com/webhook/list \
       -d session=[sessionKey]
```

4.6.3 Example Response


```
{
  "success": true,
  "response":[
        {
          "webhookId": 687557714,
          "type": "message",
          "event": "receive",
          "url": "https://test.zipwhip.com/message/receive",
          "method": "PUT"
        }
    ]
}
```

4.7 Update Web Hooks

Update an existing Web Hook configuration. Change the URL or update the Method used.

4.7.1 Parameters

Parameter	Description	Required
`session`	Session is the response of user/login.	True
`webhookId`	The Id of an installed Web Hook.	True
`url`	The destination when the event fires.	False
`method`	HTTP Method used for Web Hook. i.e. POST, PUT, GET	False

4.7.2 Example Request

```
$ curl -X POST https://api.zipwhip.com/webhook/update \
       -d session=[sessionKey] \
       -d method=PUT
```

4.7.3 Example Response

```
{
  "success": true,
  "response":[
        {
          "webhookId": 687557714,
          "type": "message",
          "event": "receive",
          "url": "https://test.zipwhip.com/message/receive",
          "method": "PUT"
        }
    ]
}
```

4.8 Delete Web Hooks

Update an existing Web Hook configuration. Change the URL or update the Method used.

4.8.1 Parameters

Parameter	Description	Required
`session`	Session is the response of user/login.	True
`webhookId`	The Id of an installed Web Hook.	True

4.8.2 Example Request

```
$ curl -X POST https://api.zipwhip.com/webhook/delete \
       -d session=[sessionKey] \
       -d webhookId=687557714
```

4.8.3 Example Response

```
{
    "success": true,
    "response": []
}
```

4.9 Single Host vs Multiple Hosts

The primary difference between single-host and multiple-host implementations is that multiple-host implementations require different send, receive, and retry queues. Specifically, there are different retry queues for each host. If there are two hosts and one host fails, then Zipwhip routes traffic to the operating host. After rerouting the traffic, Zipwhip continues to attempt to send messages to the failed host, using an exponential back-off logarithm similar to that described in the Retry Logic section.

MMS Implementation

The process Zipwhip uses for sending a text message with an image attached is designed to minimize the message payload until the image is actually sent. This means that the first message sent to the host doesn't include the image itself. Instead, it includes a value (true/false) that indicates that Zipwhip does, or does not, want to attach an image to a message.

5.1 MMS Retrieve Process

This section describes the steps necessary to retrieve MMS content from Zipwhip. When you look at the details of the message, you'll see that there is a has Attachment field. This field is a boolean value. If this value is "true", then execute a message Attachment/list web call.

Parameter	Description	Type	Required
`session`	The GUID representation of a logged-in user.	String	True
`messageId`	A unique string used to identify a given message object.	String	True

5.1.1 Example Request

```
$ curl -G https://api.zipwhip.com/messageAttachment/list \
       -d session=[sessionKey] \
       -d messageId=[id of message]
```

5.1.2 Example Response

The key data points are the storageKey values. These values are used in the next web calls.


```
{
    "total":2,
    "response": [
        {
            "fileName":"IMG_1827.jpg",
            "dateCreated":"2015-03-26T13:15:17-07:00",
            "fileSizeBytes :40101,
            "mimeType":"image/jpeg",
            "storageKey":"be9396da-a6aa-442c-9406-asdfasdfasdf"
        },
        {
            "fileName":"123_1.smil",
            "dateCreated":"2015-03-26T13:15:17-07:00",
            "fileSizeBytes":300,
            "mimeType":"application/smil",
            "storageKey":"269023ec-67a0-4481-a820-asdfasdfasdf"
        }
    ],
}
```

5.2 Retrieve Hosted Content

HostedContent is Zipwhip's brand name for our MMS storage server. All items are stored with a key and permissions.

5.2.1 Zipwhip API Parameters: MMS Retrieve Hosted Content

Parameter	Description	Type	Required
`session`	The GUID representation of a logged-in user.	String	True
`storageKey`	A unique string used to identify a given media file.	String	True

5.2.2 Example Request

```
$ curl -G https://api.zipwhip.com/hostedContent/get \
       -d session=[sessionKey] \
       -d storageKey=[storageKey]
```

5.2.3 Example Response

The response payload includes the media object.

5.3 MMS Send

Zipwhip now offers MMS on landline and toll free numbers. It is required at this time to have the feature enabled on the individual line.

Notes:

- There can be only one recipient of an MMS message.
- You can send only the following MIME types:
  - image/bmp
  - image/gif
  - image/jpeg
  - image/png
  - text/plain
- The message body is limited to 700 bytes. Images greater than 700 bytes should be sent as an attachment.
- The maximum payload is 600kb. There is no transcoding functionality.
- A session is obtained from a user/login response. Sessions do not expire. They can be stored and used for all future requests.

Parameters

Parameter	Description	Required
`session`	Session is the response of user/login.	True
`to`	The recipient's phone number in E.164 format.	True
`body`	Text body of the message. Max 700 Bytes.	False

5.3.1 Example Request (1 image & body)

```
$ curl -X POST \
       -F "image=@Filename.png" \
       https://api.zipwhip.com/messaging/send"?"session=[sessionKey]"&"
to=+12065551212"&"body=Hello
```

5.3.2 Example Request (2 images & body)

```
$ curl -X POST \
       -F"image1=@Filename01.png" \
       -F"image2=@Filename02.png" \
       https://api.zipwhip.com/messaging/send"?"session=[sessionKey]"&"
to=+12065551212"&"body=Hello
```

5.3.3 Example Request (Large text attachment = body greater than 700 bytes)

```
$ curl -X POST \
       -F"file=@textFile.txt" \
       https://api.zipwhip.com/messaging/send"?"session=[sessionKey]"&"
to=+12065551212
```