Court Notify Service Provider Guide

**Last Updated: 1/16/2024**

Court Notify (CourtNotify) is a software as a service providing a solution for public organizations to communicate with public audiences by sending emails and/or text messages and handling opt-in/out. It’s an easy and quick pluggable solution for your multiple departments. It provides Web APIs for programmable service uses and a website for online usage.

*Developed by Los Angeles Superior Court’s Court Notify team.*  
*Team Leader: Sky Kim* [*<yjkim@lacourt.org*](mailto:%3cyjkim@lacourt.org)*>*  
*System Admin Contact: Hector Pila* [*<hpila@lacourt.org*](mailto:%3cDvoong@lacourt.org) *>*

Table of Contents

[Why Should I use CourtNotify? 1](#_Toc1814202082)

[Required for Communication with the Public 2](#_Toc1682658232)

[Email 2](#_Toc1114112099)

[Text Message 3](#_Toc834193889)

[Cost and Time Saving 3](#_Toc1518986200)

[Pluggable Solution 3](#_Toc501830595)

[Handles Opt-in/out and more 3](#_Toc666530597)

[Quick Start 4](#_Toc1816408249)

[For Non-Developers, follow these steps: 5](#_Toc1454869393)

[For Developers, follow these steps: 5](#_Toc1137154296)

[Sign Up for an Account 5](#_Toc1310721358)

[Submit API Access Request 7](#_Toc424313559)

[Create/Request Subscription 9](#_Toc1135947908)

[Call our API 11](#_Toc1966948625)

[General Concept 11](#_Toc227330702)

[Overall Process Diagram 12](#_Toc386057121)

[Message Grouping Structure 12](#_Toc1237528409)

[Service Provider Account 13](#_Toc191784714)

[Grouping Messages 13](#_Toc599072454)

[Service 13](#_Toc231266585)

[Message Group 13](#_Toc301704910)

[Subscription 13](#_Toc1164431700)

[Service Details 14](#_Toc1062106235)

[Custom ‘From Email’ 15](#_Toc1317241694)

[‘From Email’ Format 15](#_Toc625714333)

[Domain Registration 15](#_Toc488789796)

[Sending emails with a Verified Domain 15](#_Toc796035616)

[Reply-To Email 16](#_Toc1927911049)

[Text Message Replies 18](#_Toc2019426578)

[General 18](#_Toc2119912392)

[Common Scenarios 18](#_Toc894271986)

[Rare Multiple Scenarios 19](#_Toc2027214342)

[Data Retention 23](#_Toc753614627)

[Developer API Guide 24](#_Toc1788779231)

[Web API 24](#_Toc2073564688)

[Common 24](#_Toc2007837147)

[Message API 27](#_Toc1660205632)

[ShortURL API 56](#_Toc1822538003)

[VerificationCode API 56](#_Toc281743410)

[HelloAuthorizedWorld API 56](#_Toc475554687)

[HelloWorld API 56](#_Toc1696081361)

[API Client 56](#_Toc2037160824)

[API Client Library for .NET NuGet package 56](#_Toc1510278480)

[Call Limit Policy 56](#_Toc1770168022)

[Error Handling 56](#_Toc763718690)

[Warning Messages 56](#_Toc567205958)

[Specifying TimeOffset 56](#_Toc1033535881)

[Value Tables 56](#_Toc1223997530)

[Message Type 56](#_Toc735871092)

[MessageQueue Status 56](#_Toc2110912338)

[Phone Invalid Reason 56](#_Toc344706888)

[Email Invalid Reason 56](#_Toc432065615)

[API Errors 56](#_Toc661606728)

[MessageQueue Errors 56](#_Toc36989497)

[HTTP Requests Status Codes 56](#_Toc111017189)

[Data Type Common Rule 56](#_Toc1043957837)

[Date Time 56](#_Toc759713888)

[Phone Number 56](#_Toc2089597973)

[Email Attachments 56](#_Toc1877585177)

[CallBack URL 56](#_Toc628323110)

[Using Website instead of API calls 56](#_Toc720151111)

[Service Provider 56](#_Toc238218299)

[Services 56](#_Toc1547071165)

[Message Groups 56](#_Toc2052472954)

[Message Log 56](#_Toc858446110)

[Send Message 56](#_Toc1393737482)

[Testing 56](#_Toc1392364410)

[Browser Support 56](#_Toc719948858)

[Release Note 56](#_Toc1301984456)

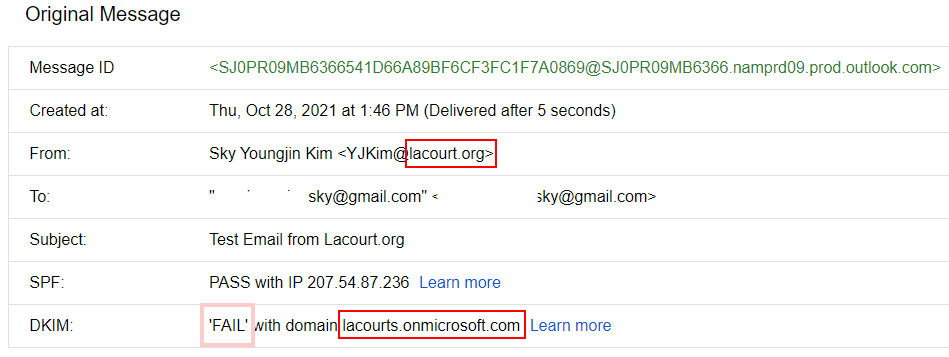
# Why Should I use CourtNotify?

Sending notifications to the public seems like a simple task, but it isn’t. There are many details required for programming and regulations you must take care of. You can save cost and time by taking advantage of the trials and errors the CourtNotify team went through by simply using CourtNotify.

## Required for Communication with the Public

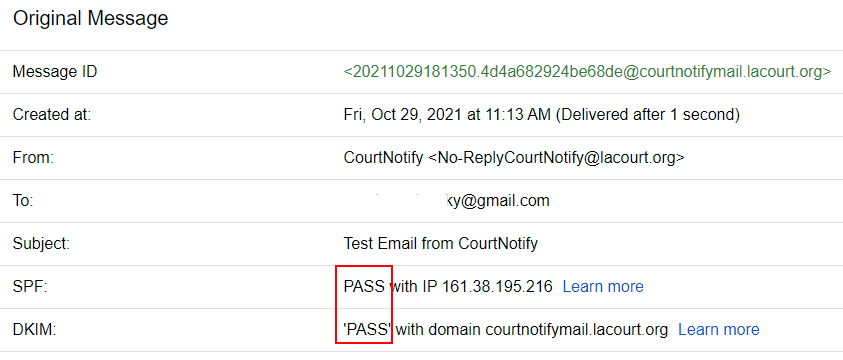
### Email

When you send emails to public recipients, you must not use Outlook. Outlook is not supposed to be used for public broadcasting and some or many of the emails sent won’t be delivered properly because of the increasing security used by email services, [DMARC (SFP and DKIM)](https://en.wikipedia.org/wiki/DMARC).



If you send an email from Outlook to a public recipient, it fails DKIM authentication as shown above because ‘From’ email domain “lacourt.org” is different from the sending service domain “onmicrosoft.com”. An email service may consider this email as a phishing email and put it into the SPAM/JUNK folder or block it completely.

CourtNotify is designed to avoid this issue and allows you to [register your domain](#Custom_FromEmail) and send emails with your domain. The email sent by CourtNotify below passes all security requirements.



CourtNotify follows the best practice suggested by [Mail-tester](https://www.mail-tester.com/) to maximize the delivery rate to the recipient’s email inbox.

Graphical user interface, text, application, email

Description automatically generated

### Text Message

When you broadcast Text Messages to the public, you need to use a pre-approved [‘Short Code’](https://support.twilio.com/hc/en-us/articles/223134587-Why-use-a-short-code-instead-of-a-long-code-) to follow the best practice by avoiding the risk of being marked as spam out of the [available options](https://support.twilio.com/hc/en-us/articles/360038173654-Comparison-of-SMS-messaging-in-the-US-and-Canada-for-long-codes-short-codes-and-toll-free-phone-numbers).

## Cost and Time Saving

It costs and takes time to prepare an environment to send emails and text messages by following the best practice.

Acquiring Short Code for the best practice of sending text messages takes a few months to register and costs $11k a year on top of the usage cost. CourtNotify holds a [dedicated Short Code](https://www.slicktext.com/sms-short-code-service.php) and will acquire more as volume goes up. Preparing an environment to send emails securely takes time to follow [the email best practices](https://documentation.mailgun.com/en/latest/best_practices.html#email-best-practices). CourtNotify has built the environment following the best practice and will put effort to keep its good status to deliver the emails to the recipients properly.

CourtNotify is ready for you to use. You don’t need to spend time and money to build the same solution.

## Pluggable Solution

You can send emails and text messages using an Email Service (Mailgun.com) and Text Message Service (Twilio.com) directly. Although it takes time and effort, it might be reasonable for you to build your solution if you need it for only ONE service such as “Appointment Reminder Notification”. However, as soon as another service or department wants to send emails and/or text messages to the public, you MUST duplicate all the setup and programming. Or you have to build a solution like CourtNotify, a pluggable solution.

CourtNotify is a pluggable solution that you can add multiple services for multiple departments easily. And you can send messages immediately about the new services.

## Handles Opt-in/out and more

CourtNotify has the common features required for communication with the public. Opt-In/Out is one of them and is handled by CourtNotify.

CourtNotify supports more common features such as email and phone number validation and a safe Short-Url.

# Quick Start

Please follow this Quick Start guide to initiate the process of accessing the Court Notify API.

## For Non-Developers, follow these steps:

* [Signing Up for an Account](#_Sign_Up_for)
* [Submit API Access Request](#_Submit_API_Access)

## For Developers, follow these steps:

* [Signing Up for an Account](#_Sign_Up_for)
* [Submit API Access Request](#_Submit_API_Access)
* [Create/Request Subscription](#_Create/Request_Subscription)
* [Calling our API](#_Call_our_API)

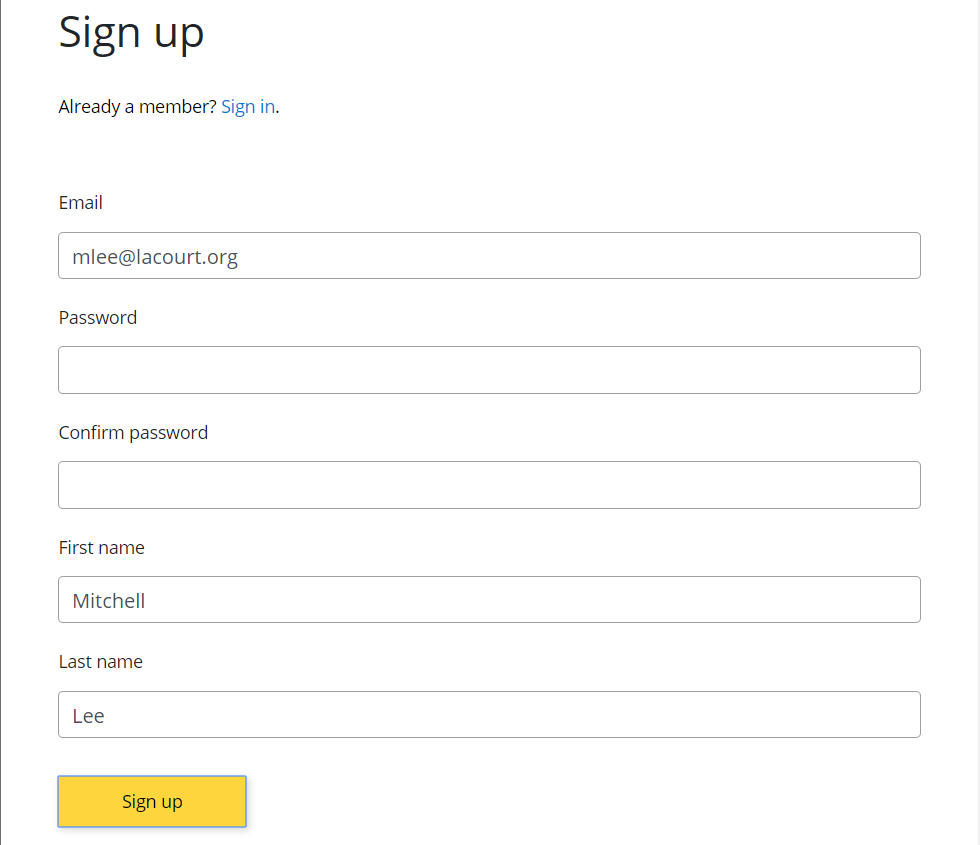
### Sign Up for an Account

1. Service Provider (SP) go to ‘Service Provider Website’ URL:  
   <https://courtnotifysp.lacourt.org>

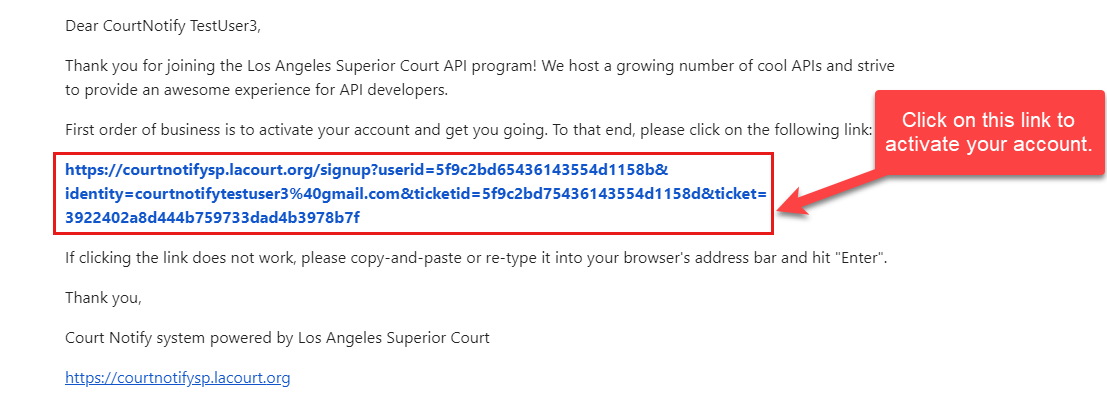
Graphical user interface, website

Description automatically generated

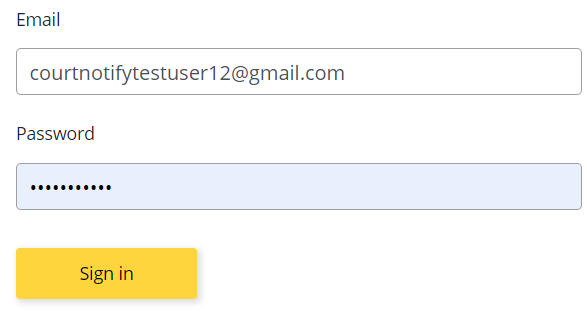
1. Click on the ‘**Sign up**’ button to create a user account for the Service Provider website.
   1. Email must be able to receive incoming email messages.



1. SP will receive an email from the Service Provider Website and will need to “**click”** on the link to activate the account.



1. Click on the ‘Sign in’ tab located on the top right corner of the website and enter the Email and Password to Sign In.

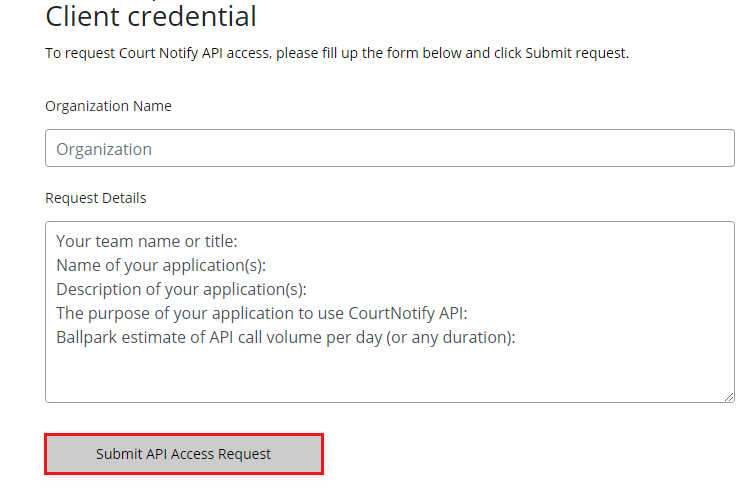


### Submit API Access Request

1. Click on ‘**Profile’** and enter your organization name under ‘Client credential’ (e.g. Los Angeles Superior Court, Orange County Superior Court) and fill out Request Details. Once finished, please click the ‘**Submit API Access Request**’.

Graphical user interface, application

Description automatically generated



1. Admin approves the Client credential request and the user will also receive an email confirmation of the approved request.
2. SP will be able to see the Client ID.



1. (FOR Non-Developers ONLY)
   1. If you are using the website only for ‘[Sending Message’](#ServiceProviderWebsite_Send_Message) and not using our API, please just add‘[Services](#_Services)’.
   2. Now you can Send Messages on the website without having to proceed forward with the remaining steps.
2. (For Developers)
   1. Please continue onto ‘Create/Request Subscription’ below.

### Create/Request Subscription

1. 'Add Secret' on the ‘Profile’ page.

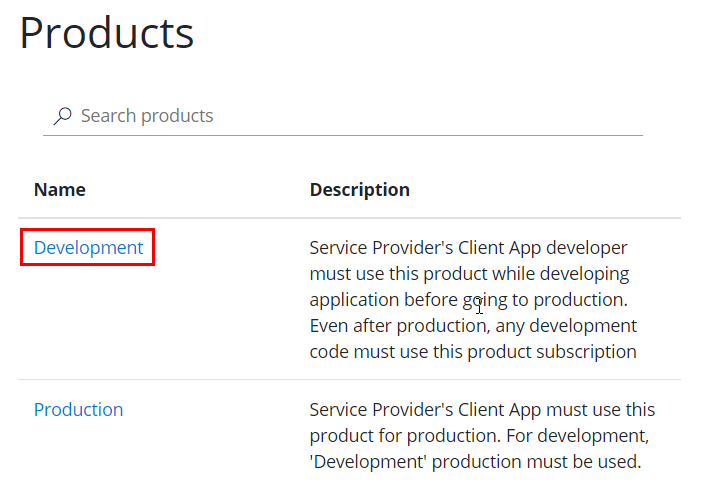


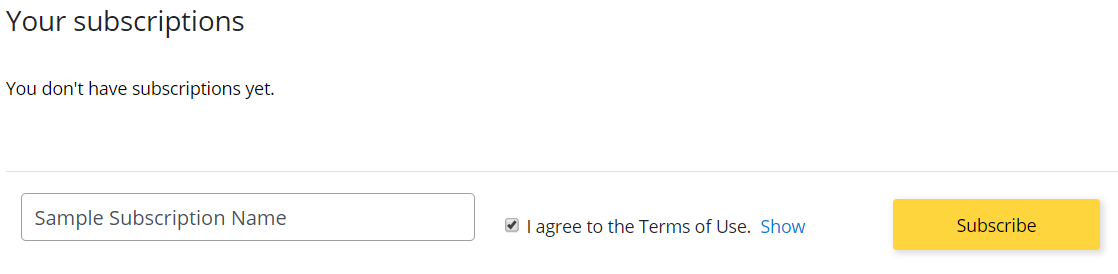
1. SP selects ‘**Products**’ and submits a subscription request for the ‘Development’ product.

=>Development has a limitation on the number of calls you can make. This is best to use to test your application before utilizing the ‘Production’ product. For more information regarding ‘Products’, please refer to the [**Limit Calls**](#Call_Limit_Policy) section in this guide.

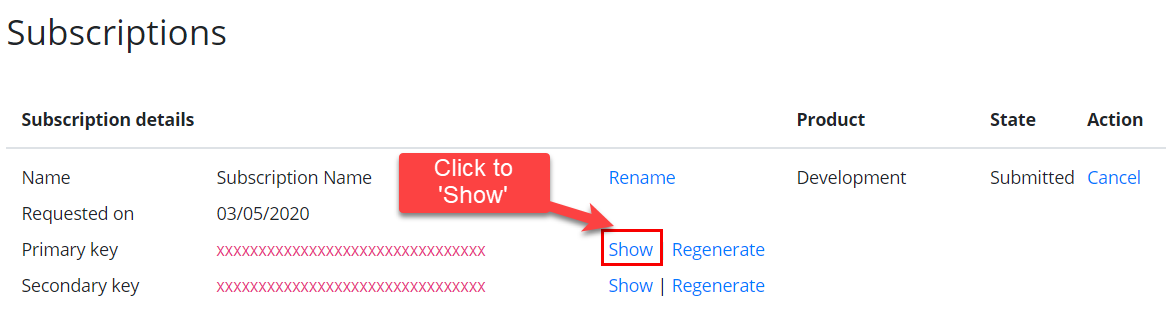
Graphical user interface, application

Description automatically generated





1. A subscription key is generated but it is not usable until Admin **approves** of the subscription.



1. Once approved by the admin, the SP’s subscription key for the product is now active.
2. SP **add** [**Services**](#Services)**.**

### Call our API

1. SP now has a client ID, secret, subscription key, and service id.

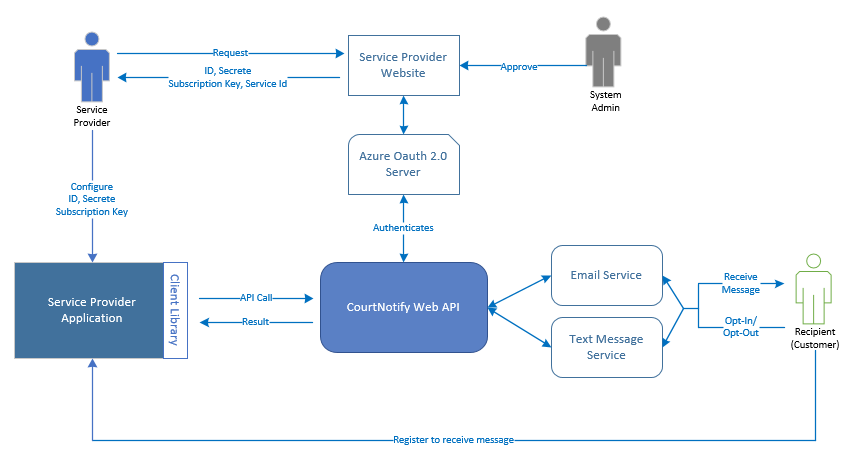
You can **call Web API for Development** with the information.

1. After testing in Development, SP can now request for ‘Production’ product.
2. Admin approves of the subscription.
3. SP **calls Web API for Production**.  
   You must use the ‘Production’ subscription key for production.
4. You can use our [API Client Library for .NET](#API_Client_Library_for_NET) to send out calls.

# General Concept

This includes the general system overview with Process Diagram, the structure of Message Grouping, and an explanation of the Service Provider account including the grouping of Messages.

## Overall Process Diagram



* Service Provider request Client Id, Secrete, Subscription Key, and create a Service from Service Provider Website.
* System Admin approves of the request.
* Service Provider can configure their information to call our Court Notify Web API.
* The recipient can register to receive messages from the Service Provider Application.
* The recipient will receive messages and has the ability to Opt-in/Opt-Out of the service.

## Message Grouping Structure

Firstly, the following three terms need to be understood and considered with their respective associations:

**Service Provider Account** (e.g. Your team)

|\_\_\_\_\_ **Services** (e.g. Application)

|\_\_\_\_\_ **Message Group** (e.g. Case Number)

You need to create a Service Provider Account to request API access. Then, you need to set up a Service to specify your application that calls API. If you want to group messages further, you can specify Message Group on each message with your ad-hoc unique identifier. The following details apply:

## Service Provider Account

The service provider is the client who has an account to call API.

* One login account registered to Service Provider Portal.
* One account per team is recommended.
* One client id is assigned per account. Client Id is used to call API.

## Grouping Messages

The system allows to group messages in two levels: Service and Message Group.  
One Account has one or more services. Each service can optionally have multiple Message Groups.

* Recipients can opt-in/out per Message Group if it exists.  
  If there is no Message Group, they can opt-in/out per service.
* The system allows a recipient to opt-out of all messages.

### Service

One team might have multiple applications that need to call API. Each application can be a service.

* At least one service is required to call API.
* This is a logical grouping of messages.  
  We recommend one service for each application.
* Services must specify whether they will Use a Message Group or not. This is enforced to prevent the recipient from Opt-Out of the entire Service as a whole.

### Message Group

When you want to group messages per case or so, this ad-hoc grouping can be used.

* Optional.
* Message Groups can be specified when calling each message.
* It’s ad-hoc grouping by any string as long as it’s unique. You can specify Id with the service abbreviation prefix to make it unique.
  + E.g.) If you have a Service named Event, you can specify your Message group as “EV-ba434-gdfg45ga34g”

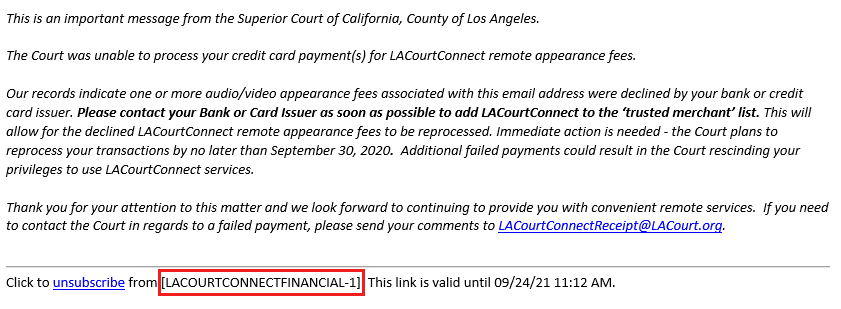
### Subscription

The term ‘Subscription’ represents the Service and Message Group that the recipient agreed to receive a text or email message. Recipients don’t need to understand what Services or Message Groups mean to Opt-In/Opt-Out because they will only see the ‘Subscription’.

#### Example of a Subscription using a Service

The Service being used is called “LACOURTCONNECTFINANCIAL-1” which sends notifications of financial payments to recipients using LA Court Connect. The image below, with the red box, highlights the name we display to our recipient in the email.

Caption: Email Message



#### Example of a Subscription using a Message Group

This text message below is using a Message Group called “Divorce/Dissolution of Marriage Orientation (Part 1 of 3) With Minor Children event at 2:00 PM on 7/6/2020.” This Message Group is under a Service called “Event”. By using the Subscription Name as “Divorce/Dissolution..”, it allows the recipient to Opt-In/Opt-Out of that Message Group only, instead of the entire Service by mistake.

Caption: Text Message



# Service Details

This section explains the rules while sending emails, sending text messages and how long messages will be kept.

## Custom ‘From Email’

Register your domain for custom branding using your domain in ‘From Email’ with this format “{FriendlyFromName} <{EmailAccount}@{RegisteredDomain} such as “OC Court Appointment <appointment-noreply@occourt.org>". By registering your domain, you can specify your ‘From Email’ instead of using our Sender Email: “Court Notify <No-ReplyCourtNotify@lacourt.org”.

### ‘From Email’ Format

* {FriendlyFromName}: This is the friendly name that recipients will see first instead of the Email Account.
* {EmailAccount}: The email address name that is used to send the email from.
* {RegisteredDomain}: The domain that is registered by the Service Provider to specify their branded email.

### Domain Registration

To register your domain, you will need to specify the domain that you want to use and email it to the [System Admin](#Support_Contact). An example of a domain is “lacourt.org”, “occourt.org”, etc.

Email Template to send to System Admin:

|  |
| --- |
| Email Subject: Domain registration on Court Notify for {InsertOrganizationName}  Email Body:  Hello,  I would like to submit a request to use this domain: {InsertYourDomain}  Reason: I need to register my domain because.…  Emails Used: The emails I will be using from my Organization are {InsertEmailsThatWillBeUsed}. |

### Sending emails with a Verified Domain

Currently, you can only use the API to specify ‘FromEmail’ and ‘FriendlyFromName’ to send out your emails with your domain. Please see [SendEmailMessage > Parameters](#SendEmailMessage_Parameters) to use ‘FromEmail’ and ‘FriendlyFromName’.

This is the hierarchy that ‘FromEmail’ and ‘FriendlyFromName’ will follow. First, at the Message level, Court Notify will use the specified ‘FromEmail’ and ‘FriendlyFromName’. If that isn’t provided, then it moves up to the Service level, and then the Organization level. Ultimately, the default will be used if the domain hasn’t been registered yet.

Here is an example below.

* Default
  + From Email: No-ReplyCourtNotify@lacourt.org
  + Friendly Name: Court Notify
* Organization
  + From Email: CaliforniaCourtNotification@CACourt.org
  + Friendly Name: California Court
* Service
  + From Email: CaliforniaCourtAppointment@CACourt.org
  + Friendly Name: California Court Appointment
* Message
  + From Email: CaliforniaCourt@CACourt.org
  + Friendly Name: California Court

Service

Message

Organization

Default

## Reply-To Email

You can specify your Reply-To email on the message level, then it will be used on the Service level, and followed by the Organization level last. Please see [SendEmailMessage > Parameters](#SendEmailMessage_Parameters) to use ‘ReplyToEmail and ‘FriendlyReplyToName’.

You need to contact the [System Admin](#Support_Contact) to add Reply-To email and Friendly Reply To Name on the Organization level.

Here is an example below of specifying Reply-To email and Friendly Reply to Name:

* Organization
  + Reply-To Email: NO-REPLYOCC@occ.org
  + Friendly Name: NO-REPLY
* Service
  + Reply-To Email: NO-REPLYAppointmentOCC@occ.org
  + Friendly Name: NO-REPLYAppointment
* Message
  + Reply-To Email: NO-REPLYAppointmentSELF-HELPOCC@occ.org
  + Friendly Name: NO-REPLY SELF-HELP

Organization

Service

Message

## Text Message Replies

### General

Court Notify will handle text message replies from recipients such as “Stop”, “Start”, and “Help”. When a recipient replies with these commands, Court Notify will reply to each response based on each scenario. The system also supports industry-standard synonyms for each command, for example, CANCEL is equivalent to STOP.

\*Please note there may be delays on some carriers for response times.

Key Terms

* Recipient: Anyone who has a subscription.
* [Subscription](#Subscription): This can be either a subscription [service or a message group](#Grouping_Messages).

### Common Scenarios

Most recipients would have only one concurrent subscription and the conversation would be simple.

#### The recipient replied “STOP”

STOP

You stopped msgs from [Court appointment at 12/1/19 3:30 PM].

36412

>



The recipient wanted to Opt-Out of receiving messages from the Subscription name “Court appointment at 12/1/19 3:30 PM”.

#### The recipient replied “START”

START

You now receive msgs from [Court appointment at 12/1/19 3:30 PM].

36412

>



The recipient wanted to Opt-In to start receiving messages from the Subscription name “Court appointment at 12/1/19 3:30 PM”.

#### The recipient replied “HELP”

HELP

You receive msgs from [Court appointment at 12/1/19 3:30 PM]. Reply ‘STOP’ to stop receiving msgs. Msg&data rates may apply. For detail, visit https://s.lacourt.org/H

36412

>



The recipient needed Help and the System replied with the active Subscription that the recipient receives messages from and a link to see more commands from the System.

### Rare Multiple Scenarios

In rare cases, a recipient might have multiple concurrent subscriptions. This could occur mostly to an attorney user who represents multiple recipients. In this case, the system helps the customer choose the right subscription to Opt-In/Opt-Out.

#### The recipient replied “STOP”

You stopped msgs from [Court appointment at 12/1/19 3:30 PM].

STOP ONE

STOP

Reply 'STOP ONE' to stop [Court appointment at 12/1/19 3:30 PM]. Reply ‘LIST’ to see all 3 subscriptions, or 'STOP ALL' to stop all. Msg&data rates may apply.

36412

>



The recipient has multiple Subscriptions and texts “STOP”, but our System sent the recipient multiple options to handle the subscription. The recipient chose to “STOP ONE” to Opt-Out of their current subscription.

You stopped msgs from [Traffic Payment Plan for Ticket #DF00012].

STOP 2

These are your subscription(s):

1. [Court appointment at 12/1/19 3:30 PM]

2. [Traffic Payment Plan for Ticket #DF00012]

3. [Jury Duty] (Stopped)

You have 1 stopped subscription(s).

Reply 'START #' or 'STOP #' to start/stop receiving msgs from the subscription by the number '#' (1-3). Reply ‘START All’ or ‘STOP ALL’ to start/stop msgs from all subscriptions. Msg&data rates may apply.

LIST

STOP

Reply 'STOP ONE' to stop [Court appointment at 12/1/19 3:30 PM]. Reply ‘LIST’ to see all 3 subscriptions, or 'STOP ALL' to stop all. Msg&data rates may apply.

36412

>



The recipient has multiple Subscriptions and texts “STOP”. The system presented the recipient with multiple options to choose from and the recipient used “LIST”. The “LIST” command is best used when the recipient has multiple Subscriptions. “LIST” shows the recipient all of their Subscriptions and also shows if the Subscription is (Stopped) meaning Opt-Out. The recipient can “STOP #” from the list of Subscriptions shown. This recipient decided to “STOP 2” to Opt-Out of the “Traffic Payment Plan for Ticket #DF00012”.

STOP ALL

You stopped msgs from all 3 subscriptions.

STOP

Reply 'STOP ONE' to stop [Court appointment at 12/1/19 3:30 PM]. Reply ‘LIST’ to see all 3 subscriptions, or 'STOP ALL' to stop all. Msg&data rates may apply.

36412

>



The recipient has multiple Subscriptions and texts “STOP”. The system presented the recipient with multiple options to choose from and the recipient used “STOP ALL” to Opt-Out of all the Subscriptions.

#### The recipient replied “START”

You now receive msgs from all 3 subscriptions.

START ALL

These are your subscription(s):

1. [Court appointment at 12/1/19 3:30 PM] (stopped)

2. [Traffic Payment Plan for Ticket #DF00012] (stopped)

3. [Jury Duty] (Stopped)

You have 3 stopped subscription(s).

Reply 'START #' to start recieving msgs from the subscription by the number #, or 'START ALL' to start all 3 subscriptions. Msg&data rates may apply.

LIST

START

Reply 'START ONE' to receive msgs from [Court appointment at 12/1/19 3:30 PM]. Reply ‘LIST’ to see all 3 subscriptions. Msg&data rates may apply.

36412

>



The recipient texted “START” and was presented with multiple options from our System. The recipient texted “LIST” to see all their Subscriptions that were (Stopped) meaning Opt-Out. The recipient wanted to receive messages from them again, so the recipient texted “START ALL” to Opt-In for all of their Subscriptions.

## Data Retention

Your messages will be stored for a certain amount of time. Once the time has been reached, it will be removed from the database.

* Message Groups: When Message Groups are expired, they will be removed.
* Message Logs: After 90 days, the messages will be removed.

For details, see [‘Message Groups’](#ServiceProviderWebsite_MessageGroupExpir) and [‘Message Log’](#ServiceProviderWebsite_MessageLogExpire) for more information on how long data will be available.

# Developer API Guide

This section will teach you what methods we have to use to get started on using Court Notify.

## Web API

APIs contain operations for notification and further communication with the public using text messages and email.

* It is a standard REST API with OAuth 2.0 protocol.
* It uses Microsoft Azure API Management technology.

Here we use the Postman tool to demonstrate API calls.   
(Download the Postman tool from [https://www.getpostman.com](https://www.getpostman.com/))

### Common

This section describes the general concepts and requirements that apply to all API calls.

#### Authentication

To call an API operation, you need to pass an access token and set the subscription key in the header.

##### Get Access Token

Navigate to Authorization on Postman, click on the drop-down for Type and select ‘OAuth 2.0’. Then click on the ‘Get New Access Token’ button.

Graphical user interface, application

Description automatically generated

You can get the Client ID and secret from the [Service Provider Website](https://courtnotifysp.lacourt.org).  
For other values, you enter the exact value shown below.

Graphical user interface, application

Description automatically generated

* Grant Type: Client Credentials
* Access Token URL: https://login.microsoftonline.com/ dc94c3c7-bb48-40ff-9305-e473fd6f6a16 /oauth2/v2.0/token
* Client Id: {Add your Client Id}
* Client Secret: {Add your Client Secret}
* Scope: api://PubShareAPIM-Public/.default

##### Set Headers

You can get a subscription key from the [Service Provider Website](https://courtnotifysp.lacourt.org).

Graphical user interface, application

Description automatically generated

* Content-Type:

Application/JSON

* Ocp-Apim-Subscription-Key:

{Key Generated}

#### Common Returns

All API operation calls return the following basic information. The different return values of each API are described in its API section.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| APIResult.successful | Bool | True if the call is successful. | True |
| APIResult.error | String | When an API call fails, this contains detailed information about why it failed such as whether the passed parameter value is invalid, or the recipient mobile number is invalid.  Error contains - Code (string): Unique error id.  - Description (string): Description of the error. - Message (string): This contains further ad-hoc detail of the error specific to this call.  See the [list of API errors.](#API_Errors) | "error":  {  "code": "100",  "description": "Required value is missing.",  "message": "Required field 'MessageGroupTitle' is missing.",  "httpStatusCode": 0  } |
| APIResult.value | Object | This property contains the actual value of the requested data by the caller.  The value type varies for each API. Please see each API section for the details. |  |
| APIResult.Warning | String | This is a warning message that is displayed to the client. | Duplicated ‘MobileNumbers’ {12223334444}” |

#### Version

This allows new API improvements while supporting the existing API implementation simultaneously. When the call is used without a Version, the call will be using the latest version.

To change the version in APIs, look for [version] and then replace it with the version you want from the [History](#Version_History).

##### Example

To use the new version, change [version] to “V1” or the latest version, as shown below:

Endpoint Template: [**https://CourtNotifyAPI.lacourt.org/msg/[version]/SendTextMessage**](https://CourtNotifyAPI.lacourt.org/msg/%5bversion%5d/SendTextMessage)

Actual URL: [**https://CourtNotifyAPI.lacourt.org/msg/V1/SendTextMessage**](https://CourtNotifyAPI.lacourt.org/msg/V1/SendTextMessage)

This will allow you to use Version 1 on our API.

##### No Version Specified

When you don’t specify the version, it uses the latest API version.

\*\*Note that if the API new version changes the method signature, your existing code may break if it is not updated. We recommend you use a specific version.

Example: Use the original URLs without any version.

Using this URL: <https://CourtNotifyAPI.lacourt.org/msg/SendTextMessage> will use the latest version.

#### Change History for each Version

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 08/31/2022 | V2 | Updates for V2:   * [‘SendTextMessage’](#SendTextMessage) and [‘SendEmailMessage’](#SendEmailMessage) version change. * [‘SendEmailMessageInBulkV2‘](#SendEmailMessageInBulkV2) has two new parameter objects ‘EmailMessageRequests’ and ‘EmailAttachments’. * [’SendTextMessageInBulkV2](#SendTextMessageInBulkV2)’ has one new parameter object ‘TextMessageRequests’. |
| 08/02/2022 | V1 | Introducing to API version. |

### Message API

This API provides a way of handling communication with recipients: send a text or email message, check the status of the message and retrieve the message log, etc.

Versioning information can be found under [Version](#Version).

Available Operations:

* [SendTextMessage](#_SendTextMessage_1)
* [SendTextMessageInBulk](#_SendTextMessageInBulk)
* [SendTextMessageInBulkV2](#SendTextMessageInBulkV2)
* [SendEmailMessage](#SendEmailMessage)
* [SendEmailMessageInBulk](#_SendEmailMessageInBulk)
* [SendEmailMessageInBulk V2](#SendEmailMessageInBulkV2)
* [GetMessageQueue](#_GetMessageQueue_1)
* [SearchMessageQueue](#_SearchMessageQueue)
* [GetPhoneOptInOut](#_GetPhoneOptInOut)
* [GetEmailOptInOut](#_GetEmailOptInOut)
* [UpdateMessageGroupStatus](#_UpdateMessageGroupStatus)
* [GetPhone](#_GetPhone)
* [GetEmailMessageStatus](#_GetEmailMessageStatus)
* [UpdateMessageGroupExpiredOn](#_UpdateMessageGroupExpiredOn)
* [UpdateMessageGroupTitle](#_UpdateMessageGroupTitle)
* [GetBulkMessageQueue](#_GetBulkMessageQueue)
* [CheckInvalidMobileNumber](#_CheckInvalidMobileNumber)
* CheckInvalidEmailAddress

#### SendTextMessage

Sends a text message to the mobile number of a recipient.

This operation has a limit on the number of times you can call it. Refer to [Call Limit Policy](#Call_Limit_Policy) for more details.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | [**https://CourtNotifyAPI.lacourt.org/msg/[version**](https://CourtNotifyAPI.lacourt.org/msg/%5bversion)**]/SendTextMessage** |

##### Post Body

* Name: “textMessageRequest”
* Type: TextMessageRequest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | **Example** |
| <<The below are the properties of TextMessageRequest>> | | | | |
| ServiceId | | Integer | The registered service under which this message is sent. Service is the default group of messages, called a subscription that the recipient can Opt-in or out of.   The Service Provider can manage the services on the Service Provider website, and get this id. | 1 |
| MobileNumbers | | List | This is the recipient’s mobile number. Mobile numbers can have multiple phone numbers in a list. | [“2133107000”,”12099657121” ] |
| TextMessage | | String (Max: 4000) | Text message to send.  The text message is split into segments for every 160 characters and the text message cost counts per segment.   NOTE that the message must contain “Msg&data rates may apply.” at the end of the message when you put a message expecting the recipient’s reply. | Your jury duty summons meeting is scheduled for 05/30/2021. Please text the letter 'C' to confirm your receipt of this message. This reply is mandatory. Msg&data rates may apply. |
| MessageGroupId (optional) | | String (Max: 50) | Unique identifier in the string to group messages under service to give recipients to Opt-In or Out. If MessageGroup is not specified, ‘Service’ is used to group messages to Opt-in or Out.  A case number is a common example of this.  This id must be unique in the system, we recommend you append the prefix of your service with a few characters, such as “JURY”, “TPP”, “AT”. | JURY-CTKS1245134 |
| MessageGroupTitle (optional) | | String (Max: 200) | Recipient-friendly text for the MessageGroupId.  This text displays to the recipient. | Case John Smith vs. People of LA |
| MessageGroupExpireOn (optional) | | DateTime | The expiration date of this Message Group (specified by MessageGroupId) is used to decide whether to show this MessageGroup to the recipient. Once expired, this message group does not display to the recipient when they want to see the list of subscriptions they are opt-in or out of.  By default 2 months away from the current time is set as the expiration date. | 2020-12-30T10:30:00 |
| CustomerOptInAgreedOn (optional) | | DateTime | The time the recipient agreed to receive a message via the Service Provider’s system, which is opt-in. When the recipient opt-out from the subscription before, this time informs the system that the recipient opt-in again.  When the recipient opt-out and this time is earlier than the opt-out time, the system throws the error with code 200 without sending the message. | 2020-05-10T01:00:00 |
| CallBackURL (optional) | | String (Max: 1000) | As soon as the status is updated, the system calls this URL to update the status to the Service Provider’s system.  strings: receipt number, status, error message, and respond value.  If you want to put security to verify the caller and to avoid the malicious call to the URL, we recommend you to put a verification code into the URL that you can verify: use our ‘[VerificationCode API’](#VerificationCode_API) to generate and verify the code.  Use this [example](#CallBackURL) for assistance with creating a callbackurl. | https://crmportalqa.lacourt.org/Appointment/MessageStatus?reminderId=0a37d3f7-9986-eb11-8137-005056a30fb9 |
| RequestResponseInText (optional) | | Object | When a response from the recipient for this message is required, you can specify the list of options from which a recipient can choose. Note that the text message must contain the response options and ask them to reply.  For example, “You have an upcoming appointment tomorrow. Please reply ‘Yes’ if you come, ‘No’ if you cannot come. Msg&data rates may apply.”  When the recipient replies, the system saves the response and time and updates the service provider via CallbackURL. |  |
|  | <<The below are the properties of RequestResponseInText.>> | | | |
| RequireResponse | Bool | If it’s “True”, it informs the system clearly that the response is expected and the system handles the response. | True |
| ResponseValidUntil | DateTime | This is the date and time the response period is valid. When a response is received after this time, the system displays an error to the recipient without saving the response. | 2021-05-15T10:00:00 |
| RespondOptionList | List | List of multiple response options with RespondText and RespondValue properties. You can include multiple text aliases for the same value.  - ResponText (string): The user-friendly text that a recipient can reply with. Please keep it short for them to type easily. - RespondValue (string): The values that are stored in DB when a recipient selects this option. | {"RespondText": "Yes", "RespondValue": "1"},  {"RespondText": "No", "RespondValue": "2"},  {"RespondText": "Y", "RespondValue": "1"},  {"RespondText": "N", "RespondValue": "2"},,  {"RespondText": "OK", "RespondValue": "1"} |

**Parameters in Body Example**

{

"ServiceId": 54,

"TextMessage":"Your jury duty summon meeting is scheduled on 05/30/2021. Please text letter 'C' to confirm with your recipient of this message. This reply is mandatory. Msg&data rates may apply.",

"MobileNumbers": ["12133107000","12099657121"],

"MessageGroupId": "CTKS1245134",

"MessageGroupTitle": "Case John Smith vs. People of LA",

"RequestResponseInText": {

"RequireResponse": "true",

"ResponseValidUntil": "2021-05-15T10:00:00",

"RespondOptionList": [

{"RespondText": "Confirm", "RespondValue": "c"},

{"RespondText": "cnfm", "RespondValue": "c"},

{"RespondText": "C", "RespondValue": "c"},

{"RespondText": "Deny", "RespondValue": "d"}

]

}

}

##### RETURNS

Returns immediate status and message information in a list if there are multiple recipients sent including ‘ReceiptNumber’ which is a unique value to track the message later.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| ReceiptNumber | String | Unique id assigned to this message. | a-7457 |
| Status | String | The current status of the message. The initial status is “Pending” or “Sending”. The further status will be updated via CallbackURL or you can call ‘[GetMessageQueue](#_GetMessageQueue_1)’ manually to check the status by passing MessageQueueId.  See [the list of Statuses](#MessageQueueStatus). | Sending |
| CreatedOn | DateTime | The time when the record was created. | 2020-03-23T13:21:22.3181776+00:00 |
| ErrorCode | String | Code of an error why this message failed to be sent. Empty if it doesn’t fail.  See [the list of message queue errors](#MessageQueueErrors). | 100 |
| ErrorMessage | String | The further detail of the error is an ad-hoc description specific to this message. | The ‘To’ number 123-000-9999 is not a valid phone number. |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the return for any reason.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"customerOpInAgreedOn": null,

"mobileNumber": "12099657121",

"textMessage": "Your jury duty summon meeting is scheduled on 05/30/2021. Please text letter 'C' to confirm with your recipient of this message. This reply is mandatory. Msg&data rates may apply.",

"requestResponseInText": {

"respondOptionList": [

{

"respondText": "Confirm",

"respondValue": "c"

},

{

"respondText": "cnfm",

"respondValue": "c"

},

{

"respondText": "C",

"respondValue": "c"

},

{

"respondText": "Deny",

"respondValue": "d"

}

],

"requireResponse": true,

"responseValidUntil": "2021-05-15T10:00:00"

},

"messageQueueId": 665757,

"clientId": "70f85e15-cd27-4aa8-85b4-0a7808f72362",

"serviceId": 54,

"messageGroupId": "CTKS1245134",

"messageType": "TextMessage",

"status": "Sending",

"createdOn": "2020-10-21T11:31:38.1143928+00:00",

"errorCode": null,

"errorMessage": null,

"callbackURL": null

}

}

###### OPT-OUT

When calling SendTextMessage and a recipient has Opt-Out of a Service or Message Group, a response will return stating that the recipient has Opt-Out.

**Response Example**

{

"successful": false,

"warning": null,

"error": {

"code": "110",

"description": "Opt Out",

"message": "Recipient opt-out for this message group Id.",

"httpStatusCode": 0

},

"value": null

}

#### SendTextMessageInBulk

Sends multiple text messages at once. The messages are queued and the system sends each message by calling ‘[SendTextMessage’](#_SendTextMessage_1). You must use this operation instead of ‘SendTextMessage’ for efficient and reliable service if you send multiple messages in a short time that might exceed [the call limit](#Call_Limit_Policy).

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/SendTextMessageInBulk** |

##### Post Body

List of parameters that are used in and the same as the ‘SendTextMessage’ operation. Please see ‘[SendTextMessage Parameters](#_PARAMETERS)’ for the details of the parameter.

**Parameters in Body Example**

[

{

"ServiceId": 32,

"TextMessage":"LA Court - You have an upcoming Court appointment at 10:00 AM on 10/19/2020.",

"MobileNumbers": ["12099657121"],

"MessageGroupId":"AP\_d86bf5e9-34fsdfdf2-ea11-8110",

"MessageGroupTitle":"Online English - Appointment Online - Get Assistance from Home #1 appointment at 10/19/20 01:30 PM"

},

{

"ServiceId": 32,

"TextMessage":"LA Court - You have an upcoming payment for a traffic ticket due by 10/19/2020.",

"MobileNumbers": ["12099657121"],

"MessageGroupId":"TP\_d86bf5e42349-34f2-ea11-8110",

"MessageGroupTitle":"Traffic Payment Plan #Abc342"

},

{

"ServiceId": 32,

"TextMessage":"LA Court - You have an upcoming Court appointment at 10:00 AM on 10/30/2020.",

"MobileNumber": ["12099657121"],

"MessageGroupId":"AP\_d86bf5e9-34f2-ea11-8110",

"MessageGroupTitle":" English - Appointment at 10/19/20 01:30 PM"

}

]

##### RETURNS

Returns the Id of Bulk MessageQueue. Individual MessageQueueId is generated when the system sends the queued message and can be read by calling ‘[SearchMessageQueue’](#_GetMessageQueueLog_1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| BulkReceiptNumber | String | Unique id for the Bulk Message Queue record. | b-a-7480 |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": b-a-392

}

}

#### SendTextMessageInBulk V2

Sends multiple text messages at once using ‘TextMessageRequests’. The messages are queued and the system sends each message by calling ‘[SendTextMessage’](#_SendTextMessage_1). You must use this operation instead of ‘SendTextMessage’ for efficient and reliable service if you send multiple messages in a short time that might exceed [the call limit](#Call_Limit_Policy).

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/v2/SendTextMessageInBulk** |

##### Post Body

* Name: “bulkTextMessageRequest”
* Type: BulkTextlMessageRequest

|  |  |  |
| --- | --- | --- |
| **Property Name** | **Type** | **Description** |
| <<The below are the properties of BulkTextMessageRequest >> | | |
| ‘[TextMessage](#_PARAMETERS)Request’ | List<TextMessageRequest> | List of parameters that are used in and the same as the parameter used in ‘SendTextMessage’ operation. Please see ‘[SendTextMessage](#_PARAMETERS) Parameters’ for the details of the parameter. |

**Parameters in Body Example**

{

"TextMessageRequests": [

{

"ServiceId": 32,

"TextMessage": "LA Court - You have an upcoming Court appointment at 10:00 AM on 10/19/2020.",

"MobileNumbers": [

"12099657121"

],

"MessageGroupId": "AP-Ada434d-6sd232=-asdad2342",

"MEssageGroupTitle": "ProcessLog04262022",

"RequestResponseInText": {

"RequireResponse": "true",

"ResponseValidUntil": "2022-07-15T10:00:00",

"RespondOptionList": [

{

"RespondText": "dasd",

"RespondValue": "dasd"

},

{

"RespondText": "cnfm",

"RespondValue": "c"

},

{

"RespondText": "C",

"RespondValue": "c"

},

{

"RespondText": "Deny",

"RespondValue": "d"

}

]

}

},

{

"ServiceId": 32,

"TextMessage": "LA Court - You have an upcoming Court appointment at 11:00 AM on 10/19/2020.",

"MobileNumbers": [

"12099657121"

],

"MessageGroupId": "AP-Ada434d-6sd232=-asdad2342",

"MEssageGroupTitle": "ProcessLog04262022"

}

]

}

##### RETURNS

Returns the Id of Bulk MessageQueue. Individual MessageQueueId is generated when the system sends the queued message and can be read by calling ‘[SearchMessageQueue’](#_GetMessageQueueLog_1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| BulkReceiptNumber | String | Unique id for the Bulk Message Queue record. | b-a-7480 |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": b-a-392

}

}

#### SendEmailMessage

Sends an email message to a recipient.

This operation has a limit on the number of times you can call it. Refer to [Call Limit Policy](#Call_Limit_Policy) for more details.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/SendEmailMessage** |

##### Post Body

* Name: “emailMessageRequest”
* Type: EmailMessageRequest

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | | **Example** |
| <<The below are the properties of EmailMessageRequest>> | | | | | |
| ServiceId | | Integer | The registered service under which this message is sent. Service is the default group of messages, called a subscription that the recipient can Opt-in or out of.   The Service Provider can manage the services on the Service Provider website, and get this id. | | 1 |
| ReplyToEmail  (optional) | | String  (Max: 200) | If the Service Provider wants to specify a certain email address to reply to instead of the From Email address, the Service Provider can use this property.  See [How to use Reply-To email](#ReplyToEmail) for more information. | | CourtNotifyComplaints@lacourt.org |
| FriendlyReplyToName (optional) | | String  (Max: 200) | Friendly reply to name for recipients to reply to. | | Court Appointments Help |
| ToEmails | | List | This is the recipient’s email address. ToEmails can have multiple email addresses in a list. | | [“Mlee@lacourt.org”,  “DVoong @lacourt.org”] |
| EmailSubject | | String (Max: 1000) | Email subject. | | Court Notify Test |
| FromEmail (optional) | | String  (Max: 200) | Service Provider can specify their “FromEmail” like “[occnotify@occourt.org](mailto:occnotify@occourt.org)” by registering their domain with Court Notify. Please see the “[Custom From Email](#Custom_FromEmail)” section. | | [OCCNotify@Occourt.org](mailto:OCCNotify@Occourt.org) |
| FriendlyFromName (optional) | | String (Max: 200) | This is the friendly-from name that shows the recipient what the Service Provider service wants to be identified as. | | Orange County Notify |
| EmailBody | | String | Email body message. It supports HTML format. | | Welcome to Court Notify. <br /> This is a testing email message. <p><b>You have an upcoming appointment.</b></p> |
| MessageGroupId  (optional) | | String (Max: 50) | Unique identifier in the string to group messages under service to give recipients to Opt-In or Out. If MessageGroup is not specified, ‘Service’ is used to group messages to Opt-in or Out.  A case number is a common example of this.  This id must be unique in the system, we recommend you append the prefix of your service with a few characters, such as “JURY”, “TPP”, and “AT”. | | JURY-CTKS1245134 |
| MessageGroupTitle (optional) | | String  (Max: 200) | Recipient-friendly text for the MessageGroupId.  This text displays to the recipient. | | Case John Smith vs. People of LA |
| MessageGroupExpireOn (optional) | | DateTime | The expiration date of this Message Group (specified by MessageGroupId) is used to decide whether to show this MessageGroup to the recipient. Once expired, this message group does not display to a recipient when they want to see the list of subscriptions they are opt-in or out of.  By default, 2 months away from the current time is set as the expiration date. | | 2020-12-30T10:30:00 |
| CustomerOptInAgreedOn (optional) | | DateTime | The time the recipient agreed to receive a message via the Service Provider’s system, which is opt-in. When the recipient opt-out from the subscription before, this time informs the system that the recipient opt-in again.  When the recipient opt-out and this time is earlier than the opt-out time, the system throws the error with code 200 without sending the message. | | 2020-05-30T12:00:00.0000000-07:00 |
| CallBackURL (optional) | | String (Max: 1000) | As soon as the status is updated, the system calls this URL to update the status to the Service Provider’s system.  Status information is passed via query strings: receipt number, status, error message, and respond value.  If you want to put security to verify the caller and to avoid the malicious call to the URL, we recommend you to put a verification code into the URL that you can verify: use our ‘[VerificationCode API](#VerificationCode_API)’ to generate and verify the code.  Use this [example](#CallBackURL) for assistance with creating a callback URL. | | https://crmportalqa.lacourt.org/Appointment/MessageStatus?reminderId=0a37d3f7-9986-eb11-8137-005056a30fb9 |
| HideUnsubscribe (optional) | | Bool | The system includes an ‘Unsubscribe’ message and a link to the page where the recipient can opt-out by default. This value allows the service provider to specify NOT to include it for any special reason bypassing “True”. | | True |
| RequestResponseInEmail (optional) | | Object | When a response from the recipient for this message is required, you can specify the list of options that a recipient can choose in this object.  When the recipient replies, the system saves the response and time and updates the service provider via CallbackURL. | |  |
|  | <<The below are the properties of RequestResponseInEmail.>> | | | | |
| RequireResponse | Bool | If it’s “True”, it informs the system clearly that the response is expected and the system handles the response. | | True |
| ResponseValidUntil | DateTime | This is the date and time the response period is valid. When a response is received after this time, the system displays an error to the Service Provider without saving the response. | | 2021-05-15T10:00:00 |
| RespondOptionList | List | List of multiple response options with RespondLinkLabel and RespondValue properties.  - RespondLinkLabel (string): The user-friendly statement that displays to the recipient. This can be long text if needed. - RespondValue (string): The values that are stored in DB when a recipient selects this option. | | {"RespondLinkLabel": "I hereby confirm a receipt of this message",  "RespondValue": "C"} |
| RequestingMessage (optional) | String (Max: 4000) | This message displays first to the recipient before the response option starts. It can be an instruction or acknowledgment or disclaimer message regarding the response. | | Please select one of the following options for your case by 10/20/2020. The failure of this response will result. |
| ResponsePosition (optional) | Byte | This specifies the location of the response options in the email body.   0: Bottom of the email body.  1: Top of the email body.  2: Custom location; when you want to put the response options in the middle of the email body. Put the string specified in the next parameter,  CustomPositionWithResponsePositionKeyWord, anywhere in the email body text. | | 2 |
| CustomResponsePositionWithResponseKeyWord (optional) | String (Max: 20) | The placeholder in the email message body will be replaced by the generated response options.  Use a specific syntax that does not conflict with any text in the email body. The recommended format is “[@PlaceHolderName].” | | [@ResponseOptionGoesHere] |
| EmailAttachments (optional) | | List | Users can attach files to send with the emails. For more information, look at [Email Attachments](#Email_Attachments). |  | |
|  | <<The below are the properties of EmailAttachments.>> | | | | |
| FullName | String  (Max: 500) | Name of the attachment file with the extension. | DataAnalysisOfUsers.csv | |
| Data | String | This is the Base64String file data. | R0lGODlhAQABAHcAACH5BAEAAAAALAAAAAABAAEAh/8A/////wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA… | |
| MetaData | Object | The object with the metadata needed describes what type of file extension is being used. |  | |
|  | <<The below are the properties of MetaData.>> | | | | |
| ContentType | String | List the MIME types here. For more information, you can click on this link for [Common Mime types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types). | text/csv | |
| ContentEncoding | String | The string character encoding to use for string/character-based MIME types. | us-ASCII | |

**Parameters in Body Example**

{

"ServiceId": 1,

"ToEmail": "mlee@lacourt.org",

"EmailSubject": "Your upcoming court appearance appointment",

"EmailBody": "<p>You have upcoming court appearance on 05/30/20 @ 10:00 am onsite at Stanley Mosk..</p> <div>ResponsePosition</div> <div><p>Duration: 1 Hours</p><p>Location: Stanley Mosk Courthouse</p><p>Room: 100</p><p>Address: 111 N. St, Los Angeles, CA 90012</p><p>Location Map: https://goo.gl/maps/8LExbhkEypTGkqsNA</p></div>",

"RequestResponseInEmail": {

"RequireResponse": "True",

"ResponseValidUntil": "2022-05-30T13:21:00.0000000-07:00",

"RespondOptionList": [

{

"RespondLinkLabel": "I hereby confirm a receipt of this messsage",

"RespondValue": "Confirm"

},

{

"RespondLinkLabel": "I hereby deny a receipt of this messsage",

"RespondValue": "Deny"

}

],

"ResponsePosition": 0,

"RequestingMessage": "Please select the button below to confirm you have received this email.",

"CustomResponsePositionWithResponseKeyWord": "[@ResponseOptionGoesHere]"

},

"EmailAttachments": [

{

"FullName": "image.png",

"Data": "R0lGODlhAQABAHcAACH5BAEAAAAALAAAAAABAAEAh/8A/////wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgEAAMEBAA7",

"MetaData": {

"ContentType": "image/png",

"ContentEncoding": "us-ascii"

}}

]

}

##### RETURNS

Returns message queue model data related to email.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| ReceiptNumber | String | Unique id assigned to this message. | a-7457 |
| Status | String | The current status of the message. The initial status is “Pending” or “Sending”. The further status will be updated via CallbackURL or you can call ‘[GetMessageQueue’](#_GetMessageQueue_1) manually to check the status by passing MessageQueueId.  See the [list of Status](#_MessageQueue_Status). | Sent |
| CreatedOn | DateTime | The time when the record was created. | 2020-01-31T15:43:14.8408704+00:00 |
| ErrorCode | String | Unique id of error that this message has not been sent.  See [the list of message queue errors](#_MessageQueue_Error). | 107 |
| ErrorMessage | String | The further detail of the error is an ad-hoc description specific to this message.  When this email is marked “invalid (ErrorCode =107 ),” this message displays the detail reason why marked invalid from the recipient email service such as ‘Gmail’ or ‘Yahoo.’ | The required field ‘ResponseValidUntil’ is missing. |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the Return for any reason.

**Response Example**

{

"successful": true,

"error": null,

"value": {

"customerOpInAgreedOn": null,

"fromEmail": "No-ReplyCourtNotify@lacourt.org",

"toEmail": "mlee@lacourt.org",

"emailSubject": "Your upcoming court appearance appointment",

"emailBody": "<p>You have upcoming court appearance on 05/30/20 @ 10:00 am onsite at Stanley Mosk..</p> <div><div>\r\n Please select the button below to confirm you have received this email.<br>\r\n <table cellspacing='10'><tr style='background-color:#0077cc'><td style='padding5px;padding-left:20px;padding-right:20px;'><a style='color:#fff;cursor: pointer;text-decoration: none;' href='https://courtnotify.lacourt.org/Response?v=e2986390-65a4-4045-90eb-7cb891428637&r=CONFIRM'>I hereby confirm a receipt of this messsage</a></td></tr></table >\r\n </div></div> <div><p>Duration: 1 Hours</p><p>Location: Stanley Mosk Courthouse</p><p>Room: 100</p><p>Address: 111 N. St, Los Angeles, CA 90012</p><p>Location Map: https://goo.gl/maps/8LExbhkEypTGkqsNA</p></div><hr />Click to <a href=\"https://courtnotify.lacourt.org/Email/Unsubscribe?v=8fb81d57-b719-45f6-b6a7-fa69dd1d6d07\">unsubscribe</a> from Admin Test Service. This link is valid until 5/27/2021 11:11:21 AM.",

"hideUnsubscribe": false,

"requestResponseInEmail": {

"respondOptionList": [

{

"respondLinkLabel": "I hereby confirm a receipt of this messsage",

"respondValue": "Confirm"

}

],

"requestingMessage": "Please select the button below to confirm you have received this email.",

"responsePosition": 2,

"customResponsePositionWithResponseKeyWord": "ResponsePosition",

"requireResponse": true,

"responseValidUntil": "2020-05-30T20:21:00+00:00"

},

"emailAttachments":[

{

"fullName": "image.png",

"metaData": {

"contentType": "image/png",

"contentEncoding": "us-ascii"

}}],

"messageQueueId": 92767,

"clientId": "fab2897a-e2ad-4fb3-b76b-d0888a00a392",

"serviceId": 1,

"messageGroupId": null,

"messageType": "Email",

"status": "Sent",

"createdOn": "2020-05-27T11:11:21.8924843+00:00",

"errorCode": null,

"errorMessage": null,

"callbackURL": null

}

}

###### OPT-OUT

When calling SendEmailMessage and a recipient has Opt-Out of a Service or Message Group, a response will return stating that the recipient has Opt-Out.

**Response Example**

{

"successful": false,

"warning": null,

"error": {

"code": "110",

"description": "Opt Out",

"message": "Recipient opt-out for this message group Id.",

"httpStatusCode": 0

},

"value": null

}

#### SendEmailMessageInBulk

Sends multiple email messages at once. The messages are queued and the system sends each message by calling ‘[SendEmailMessage](#_SendEmailMessage_1)’. You must use this operation instead of ‘SendEmailMessage’ for efficient and reliable service if you send multiple messages in a short time that might exceed [the call limit](#Call_Limit_Policy).

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/SendEmailMessageInBulk** |

##### Post Body

List of parameters that are used in and the same as the parameter used in ‘SendEmailMessage’ operation. Please see ‘[SendEmailMessage Parameters](#_PARAMETERS_1)’ for the details of the parameter.

**Parameters in Body Example**

[

{

"ServiceId": 32,

"ToEmails": ["mlee@lacourt.org"],

"EmailSubject": "Appointment Reminder for upcoming Court appointment at 10/19/20 03:15 PM",

"EmailBody": "Court Appointment"

},

{

"ServiceId": 32,

"ToEmails": ["mlee@lacourt.org"],

"EmailSubject": "Appointment Reminder for upcoming Court appointment at 10/19/20 03:15 PM",

"EmailBody": "Court Appointment"

},

{

"ServiceId": 32,

"ToEmails": ["mlee@lacourt.org"],

"EmailSubject": "Appointment Reminder for upcoming Court appointment at 10/19/20 03:15 PM",

"EmailBody": "Court Appointment"

}

]

##### RETURNS

Returns the Id of Bulk MessageQueue. Individual MessageQueueId is generated when the system sends the queued message and can be read by calling ‘[SearchMessageQueue](#_GetMessageQueueLog_1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| BulkReceiptNumber | String | Unique id for the Bulk Message Queue record. | b-a-7481 |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": “b-a-115”

}

}

#### SendEmailMessageInBulk V2

Sends multiple email messages at once using ‘EmailMessageRequests’ and ‘EmailAttachments’. The messages are queued and the system sends each message by calling ‘[SendEmailMessage](#_SendEmailMessage_1)’. You must use this operation instead of ‘SendEmailMessage’ for efficient and reliable service if you send multiple messages in a short time that might exceed [the call limit](#Call_Limit_Policy).

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/v2/SendEmailMessageInBulk** |

##### Post Body

* Name: “bulkEmailMessageRequest”
* Type: BulkEmaillMessageRequest

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Property Name** | | | **Type** | **Description** | **Example** |
| <<The below are the properties of BulkEmailtMessageRequest >> | | | | |  |
| ‘[emailMessageRequest](#_PARAMETERS_1)’ | | | List<EmailMessageRequest> | List of parameters that are used in and the same as the parameter used in ‘SendEmailMessage’ operation. Please see ‘[SendEmailMessage Parameters](#_PARAMETERS_1)’ for the details of the parameter. |  |
| ‘emailMessageAttachment’ | | | List< EmailMessageAttachment> |  |  |
|  | <<The below are the properties of EmailMessageAttachment >> | | | |  |
|  | FullName | | string | File name with extension | Test1.txt |
|  | Data | | string | Base64-encoded string of the file's contents. | TVVMTiB0byBkYScgJiN4MjEzMzsgJiN4MjYzRDsgJiN4MjYzRTsgJi…. |
|  | DataBinary (Optional) | | byte[] | GZipped, binary of the file's contents. |  |
|  | StorageType (Optional) | | string | Type type of data being stored: 'Base64' or 'GZip'. If 'GZip', 'DataBinary' must be set. If 'Base64', 'Data' must be set. | Base64 |
|  | MetaData | | EmailMessageAttachmentMetaData | Extra info about the file that's required, such as ContentType (MIME type) and Encoding (character-encoding type). |  |
| <<The below are the properties of EmailMessageAttachmentMetaData >> | | | | |  |
|  |  | ContentType | string | The MIME type of the file. E.g., application/pdf, text/plain, text/csv, image/png, etc. | text/plain |
|  |  | ContentEncoding | string | The string character encoding to use for string/character-based MIME types. Use the web name; accceptable encodings are as follows: utf-7, utf-8, utf-16 (for Unicode), utf-32, us-ascii (for ASCII/binary content). | utf-8 |

**Parameters in Body Example**

{

"EmailMessageRequests": [

{

"ServiceId": 32,

"ToEmails": [

"courtnotifytestuser1@gmail.com"

],

"EmailSubject": "Court Appointment on 06/30/2022 at 10:00 AM",

"FriendlyFromName": "Deploy D",

"EmailBody": "<p>Good Morning Applicant</p><p>You hae a Court Appointment on 06/30/2022 at 10:00 AM. Please confirm your appointment below. Thank you</p>",

"MessageGroupId": "DeployD06232022",

"MessageGrouptitle": "TestingDeployD07112022",

"HideUnsubscribe": 0,

"RequestResponseInEmail": {

"RequireResponse": "True",

"ResponseValidUntil": "2023-05-30T13:21:00.0000000-07:00",

"RespondOptionList": [

{

"RespondLinkLabel": "I hereby confirm a receipt of this messsage",

"RespondValue": "Confirm"

},

{

"RespondLinkLabel": "I hereby deny a receipt of this messsage",

"RespondValue": "Deny"

}

],

"ResponsePosition": 0,

"RequestingMessage": "Please select the button below to confirm you have received this email.",

"CustomResponsePositionWithResponseKeyWord": "[@]"

}

}

],

"EmailAttachments": [

{

"FullName": "Test1.txt",

"Data": "TVVMTiB0byBkYScgJiN4MjEzMzsgJiN4MjYzRDsgJiN4MjYzRTsgJiN4MjExNTsgJiN4MjdCOTsgJiN4Mjc5QTsgJiN4MjYxRDsNCg0KTVVMTiB0byBkYScg4oSzIOKYvSDimL4g4oSVIOKeuSDinpog4pidDQoNCiYjeDI3Qjk7ICYjeDI3OUE7ICYjeDI2M0Q7ICYjeDI2M0U7ICYjeDI2MUQ7DQoNCg0KJDEyLDI2MS41NyBYIDAuMzUgPSA0MjkxLjU0OTUNCg0KT0xEIFR3aWxpbyBEZXYgQWNjb3VudCBTSUQ6IEFDYjdjOWM0ZjNjODBiOWIwNmI1NjM5MDliZjIxOTkwMmENCkFDMThkMWJkN2UyNDdmYjc2YjE5NDZlOGExMWM5NWFjZjkNCg0K",

"MetaData": {

"ContentType": "text/plain",

"ContentEncoding": "utf-8"

}

}

]

}

##### RETURNS

Returns the Id of Bulk MessageQueue. Individual MessageQueueId is generated when the system sends the queued message and can be read by calling ‘[SearchMessageQueue](#_GetMessageQueueLog_1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| BulkReceiptNumber | String | Unique id for the Bulk Message Queue record. | b-a-7481 |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": “b-a-115”

}

}

#### GetMessageQueue

Get a specific MessageQueue object to check the status of the original message queued.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT** **TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/GetMessageQueue[?receiptNumber**](https://CourtNotifyAPI.lacourt.org/msg/GetMessageQueue%5b?receiptNumber)**]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Example** |
| ReceiptNumber | String | Unique id assigned to this message. | a-7457 |

##### RETURNS

Returns data from MessageQueue, EmailMessage, and TextMessage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | **Example** |
| <<>> | | | | |
| MessageType | | String | The type of message whether it would be a TextMessage or Email. See [the list of message types](#Message_Type). | Email |
| MessageTypeEnum | | Integer | The Id that has the type of message name. | 2 |
| Status | | String | The current status of the message. The system sends text messages async. The initial status is “Pending” or “Sending”. The further status will be updated via CallbackURL.  See the [list of Status](#MessageQueueStatus). | Sent |
| StatusEnum | | Integer | The numeric value of Status.  See the [list of Status](#MessageQueueStatus). | 3 |
| ErrorCode | | String | Unique id of error that this message has not been sent.  See [the list of message queue errors](#MessageQueueErrors). | 100 |
| ErrorMessage | | String | A generic error message stating that identifies what was wrong. | Invalid Phone Number |
| Note | | String | Any note that pertains to the Message Queue record. | This record will be removed later. |
| CallBackURL | | String | As soon as the status is updated, the system calls this URL to update the status to the Service Provider’s system.   Status information is passed via query strings: message Id, status, and error message.  If you want to put security to verify the caller and to avoid the malicious call to the URL, we recommend you to put a verification code into the URL that you can verify: use our [‘VerificationCode API’](#VerificationCode_API) to generate and verify the code. | <https://jury.lacourt.org/JuryNotificationApp1/HandleTextMsg?v=abce-dsec-123a-3gse>&MessageId=1&status=3 |
| CallBackCalledOn | | DateTime | This is the date and time the CallBackURL was called. | 2020-10-14 15:53:10.883 |
| TextMessageInfo | | Object | Contains information related to Text Messages. See [Parameters for SendTextMessage](#SendTextMessage_Parameters). |  |
| EmailMessageInfo | | Object | Contains information related to Email Messages. See [Parameters for SendEmailMessage](#SendEmailMessage_Parameters). |  |
| ServiceInfo | | Object | The ServiceInfo. Is an object that has attributes related to the Services associated with the Message Queue Id searched. |  |
|  | <<The below are the properties of ServiceInfo.>> | | | |
| ClientAppId | Integer | The ID of the Owner of the Service. | 2 |
| Abbr | String | Unique abbreviation of the Service title. | AP |
| Title | String | The name of the Service. | Appointment |
| Description | String | Description of what the service is. | This service is used to create appointments for recipients. |
| DisclaimerForPublic | String | The generic message that can be used to show to the public when a recipient Opt-In for a message. | Welcome to Court Notify, You will be receiving messages from different applications. Msg&data rates may apply. |
| Status | Bool | Indicate if this service is active or not.  1: Active Service 0: Inactive Service | 1 |
| CreatedOn | DateTime | The date and time the Service was created. | 2020-10-14 15:53:10.883 |
| UpdatedOn | DateTime | The date and time the Service was updated on. | 2020-10-16 15:53:10.883 |
| VerificationCodes | String | This is the Verification Code that can be used to associate with the Service. | 3f6a4f54-ce4d-46d3-a9f6-0ec081a852d3 |
| MessageGroupInfo | | Object | Contains information related to Message Group Info. |  |
|  | <<The below are the properties of MessageInfo.>> | | | |
| GroupId | String | ID to sub-group messages under the service. A case number is a common example of this. | CTKS1245134. |
| Title | String | Recipient-friendly text for MessageGroupId.  This must be used when sending with a MessageGroupId. Our Text Messaging Services uses the title to display to the recipient when they receive text messages. | Case John Smith vs. People of LA |
| ClientAppId | Integer | The ID of the Owner of the Service. | 2 |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| CreatedOn | DateTime | The date and time the Service was created. | 2020-10-14 15:53:10.883 |
| Status | Bool | Indicate if this MessageGroup is active or not.  1: Active MessageGroup 0: Inactive MessageGroup | 1 |
| ActivatedOn | DateTime | The date and time the Message Group was activated. | 2020-10-14 15:53:10.883 |
| DeactivatedOn | DateTime | The date and time the Message Group was deactivated. | 2020-10-15 15:53:10.883 |
| ExpireOn | DateTime | The date and time the Message Group will be expired and once expired it will change the status to inactive.  If an SP sends a new message within an expired MessageGroup, ExpireOn will be updated with an additional month. | 2020-10-14 15:53:10.883 |

**Response Example**

"successful": true,

"warning": null,

"error": null,

"value": {

"messageTypeEnum": 2,

"statusEnum": 3,

"statusUpdatedOn": "2020-10-19T14:16:30.52+00:00",

"sentOn": null,

"callbackCalledOn": null,

"note": null,

"responseValue": null,

"respondOn": null,

"textMessageInfo": null,

"emailMessageInfo": {

"fromEmail": "No-ReplyCourtNotify@lacourt.org",

"recipientEmail": "mlee@lacourt.org",

"emailSubject": "Appointment Reminder for upcoming Court appointment at 10/19/20 03:15 PM",

"emailBody": "Court Appointment is scheduled on 10/19/2020 at 3:15 PM.<br /><br />Please visit the following web page to view the detail or cancel the appointment:<br

"hideUnsubscribe": false,

"requestResponseInEmail": null

},

"serviceInfo": {

"id": 3,

"clientAppId": 2,

"abbr": "AP",

"title": "Appointment",

"description": "Users can schedule appointments before visting the court",

"disclaimerForPublic": null,

"status": true,

"createdOn": "2020-02-18T21:03:22.767",

"updatedOn": "2020-02-18T21:03:22.77",

"verificationCodes": null,

"servicePhones": null,

"serviceEmails": null,

"messageGroups": null,

"messageQueues": null

},

"messageGroupInfo": {

"id": 62398,

"groupId": "AP\_3966e434-5012-eb11-811f-005056a3a54a",

"title": "Court appointment at 10/19/20 03:15 PM",

"clientAppId": 2,

"clientAppId": 2,

"serviceId": 3,

"createdOn": "2020-10-19T21:15:35.15",

"status": true,

"activatedOn": "2020-10-19T21:15:35.15",

"deactivatedOn": null,

"expireOn": "2020-10-19T22:30:00",

"clientApp": null,

"service": null,

"messageGroupPhones": null,

"messageGroupEmails": null,

"messageQueues": null

},

"messageQueueId": 655177,

"clientId": null,

"serviceId": 3,

"messageGroupId": "AP\_3966e434-5012-eb11-811f-005056a3a54a",

"messageType": "Email",

"status": "Sent",

"createdOn": "2020-10-19T14:16:28.987+00:00",

"errorCode": null,

"errorMessage": null,

"callbackURL": null

}

}

#### SearchMessageQueue

Retrieve or search the list of messages queued by specifying the criteria.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/SearchMessageQueue** |

##### Post Body

* Name: “messageQueueRequest”
* Type: MessageQueueRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| <<The below are the properties of MessageQueueRequest >> | | | |
| FromDate | DateTime | Start date of a date range to search against the time when a message is queued. | 2020-01-01 |
| ToDate | DateTime | The end date of a date range to search against the time when a message is queued. | 2020-02-28 |
| ServiceId (optional) | Integer | Filter by a service. No filtering is not specified. | 1 |
| MessageType (optional) | Integer | Filter by MessageType to filter Email or Text messages or both if the value is not specified.   See [the list of message types](#Message_Type). | 2 |
| Status (optional) | Integer | Filter by a specific status of the Message Queue.  See the [list of Status](#MessageQueueStatus). | 3 |
| MessageGroupId (optional) | String | Filter by a Message Group  No filtering is not specified. | TPP-CAD245BDI |

**Parameters in Body Example**

{

"FromDate":"2020-01-20",

"ToDate": "2020-01-30",

"ServiceId": 1,

"MessageType": 2,

"Status": 3

}

##### RETURNS

Returns the list of records of [MessageQueue which is defined in the ‘GetMessageQueue’ operation](#_RETURNS_2).

**Response Example**

{

    "successful": true,

    "error": null,

    "value": [

        {

            "messageQueueId": 65,

            "clientAppId": 0,

            "serviceId": 1,

            "messageType": 2,

            "messageTypeEnum": 2,

            "messageGroupId": 0,

            "status": 3,

            "statusEnum": 3,

            "statusUpdatedOn": "2020-01-29T09:58:20.21+00:00",

            "sentOn": "2020-01-29T09:58:20.217+00:00",

            "errorMessage": null,

            "errorCode": null,

            "callbackURL": null,

            "callbackCalledOn": null,

            "createdOn": "2020-01-29T09:58:17.96+00:00",

            "note": null,

            "textMessageInfo": null,

            "emailMessageInfo": {

                "emailMessageId": 1,

                "receiptNumber": “a-3265”,

                "emailId": 1,

                "FromEmailUsed": "mgonzalez@lacourt.org",

                "emailSubject": "Test subject from VS APIClientCoreTestsDev",

                "emailBody": "Test body from VS APIClientCoreTestsDev",

                "messageQueueInfo": null

            },

            "serviceInfo": null,

            "messageGroupInfo": null

        },

        {

            "messageQueueId": 109,

            "clientAppId": 0,

            "serviceId": 1,

            "messageType": 2,

            "messageTypeEnum": 2,

            "messageGroupId": 0,

            "status": 3,

            "statusEnum": 3,

            "statusUpdatedOn": "2020-01-29T11:54:52.867+00:00",

            "sentOn": "2020-01-29T11:54:52.88+00:00",

            "errorMessage": null,

            "errorCode": null,

                "emailMessageId": 25,

                "messageQueueId": 109,

                "emailId": 2,

                "emailSender": "no-replycourtnotify@lacourt.org",

                "emailSubject": "<h1>This my header</h1>",

                "emailBody": "<h1>This is a testing</h1>",

                "messageQueueInfo": null

            },

            "serviceInfo": null,

            "messageGroupInfo": null

        }

    ]

}

            "createdOn": "2020-01-29T11:54:52.443+00:00",

            "note": null,

            "textMessageInfo": null,

            "emailMessageInfo": {

                "emailMessageId": 25,

                "messageQueueId": 109,

                "emailId": 2,

                "FromEmailUsed": "no-replycourtnotify@lacourt.org",

                "emailSubject": "<h1>This my header</h1>",

                "emailBody": "<h1>This is a testing</h1>",

                "messageQueueInfo": null

            },

            "serviceInfo": null,

            "messageGroupInfo": null

        }

    ]

}

#### GetPhoneOptInOut

Retrieves the complete information of a mobile number whether the number is valid. This operation should not be called multiple times to retrieve multiple mobile numbers. Recommended use it for a website to check the status of one mobile number.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/GetPhoneOptInOut[?MobileNumber**](https://CourtNotifyAPI.lacourt.org/msg/GetPhoneOptInOut%5b?MobileNumber)**]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| MobileNumber | String | Recipient phone number | 12138300144 |

##### RETURNS

Returns value from Phone, Services, and MessageGroups.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | **Example** |
| Id | | Integer | Identifier of a Phone record. | 1 |
| Invalid | | Bool | 1: Phone number is invalid 0: Phone number is valid Null | 1 |
| InvalidCheckedOn | | DateTime | The date and time that the Phone number became invalid. | 2020-10-14 15:53:10.883 |
| InvalidReason | | Integer | The different reasons why this phone number is considered invalid.  Please see the list of [Phone Invalid Reasons](#Phone_Invalid_Reason). | 3 |
| ServiceInfo | | Object | The ServiceInfo. Is an object that has attributes related to the Services associated with the Message Queue Id searched. |  |
|  | <<The below are the properties of ServiceInfo.>> | | | |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| Status | Bool | 0: Inactive (False)  1: Active (True) | True |
| IsOptIn | Bool | 0: Opt-Out (False) 1: Opt-In (True) | True |
| OptInType | Integer | 1: Client App Text Message – Service Providers who manage their services. 2: Recipient Text Message – This is related to the recipient’ actions. 3: Website | 1 |
| OptInOn | DateTime | The date and time the Service was Opt-In to. | 2020-01-30T13:29:06.75+00:00 |
| OptOutOn | DateTime | The date and time the Service was Opt-Out on. | 2020-02-30T13:29:06.75+00:00 |
| MessageGroupInfo | | Object | Contains information related to Message Group Info. |  |
|  | <<The below are the properties of MessageInfo.>> | | | |
| GroupId | String | Id to sub-group messages under the service. A case number is a common example of this. | CTKS1245134 |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| Status | Bool | 0: Inactive (False)  1: Active (True) | True |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| ActivatedOn | DateTime | The date and time the Message Group was activated. | 2020-10-14 15:53:10.883 |
| DeactivatedOn | DateTime | The date and time the Message Group was deactivated. | 2020-10-15 15:53:10.883 |
| ExpireOn | DateTime | The date and time the Message Group will be expired and once expired it will change the status to inactive. | 2020-10-14 15:53:10.883 |
| IsOptIn | Bool | 0: Opt-Out (False) 1: Opt-In (True) | True |
| OptInType | Integer | 1: Client App Text Message – Service Providers who manage their services. 2: Recipient Text Message – This is related to the recipients’ actions. 3: Website | 1 |
| OptInOn | DateTime | The date and time the Service was Opt-In to. | 2020-01-30T13:29:06.75+00:00 |
| OptOutOn | DateTime | The date and time the Service was Opt-Out on. | 2020-02-30T13:29:06.75+00:00 |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"errorCode": null,

"errorMessage": null,

"phoneInfo": {

"id": 10,

"mobileNumber": "12099657121",

"invalid": false,

"invalidCheckedOn": null,

"invalidReason": null,

},

"emailInfo": null,

"services": [

{

"serviceId": 1,

"status": true,

"isOptIn": true,

"optInType": "ClientAppTextMessage",

"optInOn": "2020-01-30T13:29:06.75+00:00",

"optOutType": "",

"optOutOn": null

}

],

"messageGroups": [

{

"groupId": "NewGroupId",

"serviceId": 1,

"status": false,

"activatedOn": "2020-01-30T10:50:20.483+00:00",

"deactivatedOn": "2020-01-30T11:54:51.513+00:00",

"isOptIn": true,

"optInType": "ClientAppTextMessage",

"optInOn": "2020-01-30T13:29:06.8+00:00",

"optOutType": "",

"optOutOn": null

},

{

"groupId": "Testing\_Group\_2",

"serviceId": 1,

"status": true,

"activatedOn": "2020-01-30T13:29:17.327+00:00",

"deactivatedOn": null,

"isOptIn": true,

"optInType": "ClientAppTextMessage",

"optInOn": "2020-01-30T13:29:17.343+00:00",

"optOutType": "",

"optOutOn": null

}

]

}

}

#### GetEmailOptInOut

Retrieves information regarding if the email is invalid, services, and message groups that are associated with the email. This operation should not be called multiple times to retrieve multiple email addresses. Recommended use it for a website to check the status of one email address.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/GetEmailOptInOut[?emailAddress]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| EmailAddress | String | Recipient Email Address | mlee@lacourt.org |

##### RETURNS

Return Email with related Services and Message Groups.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | **Example** |
| Id | | Integer | Identifier of the Email Id. | 1 |
| Invalid | | Bool | 1: Email address is invalid 0: Email address is valid Null | 1 |
| InvalidCheckedOn | | DateTime | The date and time that the Email became invalid. | 2020-10-14 15:53:10.883 |
| ServiceInfo | | Object | The ServiceInfo. Is an object that has attributes related to the Services associated with the Message Queue Id searched. |  |
|  | <<The below are the properties of ServiceInfo.>> | | | |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| Status | Bool | 0: Inactive (False)  1: Active (True) | True |
| IsOptIn | Bool | 0: Opt-Out (False) 1: Opt-In (True) | True |
| OptInType | Integer | 1: Client App Text Message – Service Providers who manage their services. 2: Recipient Text Message – This is related to the recipients' actions. 3: Website | 1 |
| OptInOn | DateTime | The date and time the Service was Opt-In to. | 2020-01-30T13:29:06.75+00:00 |
| OptOutOn | DateTime | The date and time the Service was Opt-Out on. | 2020-02-30T13:29:06.75+00:00 |
| MessageGroupInfo | | Object | Contains information related to Message Group Info. |  |
|  | <<The below are the properties of MessageInfo.>> | | | |
| GroupId | String | Id to sub-group messages under the service. A case number is a common example of this. | CTKS1245134 |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| Status | Bool | 0: Inactive (False)  1: Active (True) | True |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| ActivatedOn | DateTime | The date and time the Message Group was activated. | 2020-10-14 15:53:10.883 |
| DeactivatedOn | DateTime | The date and time the Message Group was deactivated. | 2020-10-15 15:53:10.883 |
| ExpireOn | DateTime | The date and time the Message Group will be expired and once expired it will change the status to inactive. | 2020-10-14 15:53:10.883 |
| IsOptIn | Bool | 0: Opt-Out (False) 1: Opt-In (True) | True |
| OptInType | Integer | 1: Client App Text Message – Service Providers who manage their services. 2: Customer Text Message – This is related to the customers' actions. 3: Website | 1 |
| OptInOn | DateTime | The date and time the Service was Opt-In to. | 2020-01-30T13:29:06.75+00:00 |
| OptOutOn | DateTime | The date and time the Service was Opt-Out on. | 2020-02-30T13:29:06.75+00:00 |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the Return for any reason.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"errorCode": null,

"errorMessage": null,

"phoneInfo": null,

"emailInfo": {

"id": 2,

"emailAddress": "mlee@lacourt.org",

"invalid": false,

"invalidCheckedOn": "2929-01-24T16:00:00+00:00",

"invalid”: null

},

"services": [

{

"serviceId": 1,

"status": true,

"isOptIn": true,

"optInType": "ClientAppEmail",

"optInOn": "2020-01-30T14:18:16.09+00:00",

"optOutType": "",

"optOutOn": null

},

{

"serviceId": 8,

"status": true,

"isOptIn": true,

"optInType": "ClientAppEmail",

"optInOn": "2020-01-30T10:51:59.45+00:00",

"optOutType": "",

"optOutOn": null

}

],

"messageGroups": []

}

}

#### UpdateMessageGroupStatus

Activate or deactivate a message group.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/UpdateMessageGroupStatus[?messageGroupId**](https://CourtNotifyAPI.lacourt.org/msg/UpdateMessageGroupStatus%5b?messageGroupId)**]&[status=false]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| MessageGroupId | String | Unique group Id | NewGroupId |
| Status | Bool | You can activate or deactivate a message group  True: activate  False: deactivate | false |

##### RETURNS

Returns confirmation that the message group id has been successfully deactivated or activated.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": null

}

#### GetPhone

Checks if the mobile number provided is valid. It provides phone, service, and message group information relating to the mobile number.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/GetPhone?[mobileNumber]&[serviceId]&[messageGroupId]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| MobileNumber | String | Recipient’s Mobile Number. | 12099657121 |
| ServiceId | Int | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| MessageGroupId | String | Id to sub-group messages under the service. A case number is a common example of this. | CTKS12451 |

##### RETURNS

Returns phone, service, and message group information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| isNumberFound | Bool | 1: True if it exists in our Database  0: False if it does not exist in our Database | 1 |
| Invalid | Bool | 1: Email address is invalid 0: Email address is valid Null | 1 |
| InvalidCheckedOn | DateTime | The date and time that the Email became invalid. | 2020-10-14 15:53:10.883 |
| ServiceIsRemoved | Bool | 1: True if the Service was removed and not accessible anymore. 0: False the Service is still active. | False |
| ServiceIsRemovedOn | DateTime | The date and time that the Service was removed. | 2020-10-14 15:53:10.883 |
| MessageGroupIsRemoved | Bool | 1: True if the MessageGroup was removed and not accessible anymore. 0: False MessageGroup is still active. | False |
| MessageGroupRemovedOn | DateTime | The date and time that the Email was removed. | 2020-10-14 15:53:10.883 |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the Return for any reason.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"isNumberFound": true,

"invalid": false,

"invalidCheckedOn": null,

"isOptIn": true,

"optInOn": "2020-02-18T16:20:47.027+00:00",

"optInType": "ClientAppTextMessage",

"optOutOn": null,

"optOutType": null,

"serviceId": 1,

"serviceTitle": "Admin Test Service",

"serviceIsOptIn": true,

"serviceOptInOn": "2020-02-19T13:47:57.11+00:00",

"serviceOptInType": "ClientAppTextMessage",

"serviceOptOutOn": null,

"serviceOptOutType": null,

"serviceIsRemoved": false,

"serviceRemovedOn": null,

"messageGroupGroupId": "CTKS12451",

"messageGroupIsOptIn": true,

"messageGroupOptInOn": "2020-03-23T13:21:22.797+00:00",

"messageGroupOptInType": "ClientAppTextMessage",

"messageGroupOptOutOn": null,

"messageGroupOptOutType": null,

"messageGroupIsRemoved": false,

"messageGroupRemovedOn": null

}

}

#### GetEmailMessageStatus

Checks the email status to see if the message has been delivered.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/GetEmailMessageStatus?[receiptNumber]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| ReceiptNumber | String | Unique id assigned to this message. | a-7457 |

##### RETURNS

Returns status of the email message queue that was sent.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| status | Integer | The current status of the message. The system sends text messages async. The initial status is “Pending” or “Sending”. The further status will be updated via CallbackURL.  See [the list of statuses.](#MessageQueueStatus) | Sent |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

“status”: “Sent”

}

}

#### UpdateMessageGroupExpiredOn

If a message group has expired, Service Provider can use this API to update the Expiration DateTime. This operation should not be used to create a NEW Message Group.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/UpdateMessageGroupExpiredOn?[messageGroupId]?[ExpireOn]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| MessageGroupId | String | Id to group messages such as case number. Not filtered if not specified. | AP\_32f9056a30fb9 |
| ExpireOn | DateTime | Date time for the Message Group to expire. | 2020-02-01 |

##### RETURNS

Returns as true if all validations are met.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": true

}

#### UpdateMessageGroupTitle

Service Providers can update the title of their message groups by calling this API.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/msg/[version]/UpdateMessageGroupTitle** |

##### Post Body

* Name: “updateMessageGroupTitleRequest”
* Type: UpdateMessageGroupTitleRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| <<The below are the properties of UpdateMessageGroupTitleRequest >> | | | |
| MessageGroupId | String | Id to group messages such as case number. No filtering is not specified. | CAD245BDI |
| MessageGroupTitle | String | Customer-friendly text for MessageGroupId.  This must be used when sending with a MessageGroupId. Our Text Messaging Services uses the title to display to the customer when they receive text messages. | Probate Test appointment at 09/17/20 4 PM |

**Parameters in Body Example**

{

"MessageGroupId": " CAD245BDI ",

"MessageGroupTitle": "Probate Test appointment at 09/17/20 4 PM"

}

##### RETURNS

Returns as true if all validations are met.

**Response Example**

{

    "successful": true,

"warning": null,

    "error": null,

    "value": true

}

#### GetBulkMessageQueue

Retrieve detailed information related to the bulk message queue id and the statuses of the messages that were sent.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/GetBulkMessageQueue[?bulkreceiptNumber**](https://CourtNotifyAPI.lacourt.org/msg/GetBulkMessageQueue%5b?bulkreceiptNumber)**]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| BulkReceiptNumber | String | Unique id assigned to this message. | b-a-7457 |

##### RETURNS

Returns data from BulkMessageQueue, and MessageQueue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Name** | | **Type** | **Description** | **Example** |
| TotalMessageCount | | Integer | The total number of messages that were requested to be sent out. | 2 |
| Status | | Integer | 0: Pending  1: In Progress  2: Complete | Complete |
| CreatedOn | | DateTime | The time when the record was created. | 2020-03-23T13:21:22.3181776+00:00 |
| MessageQueueInfo | | Object | The MessageQueueInfo. Is an object that has attributes related to the Message Queue record. |  |
|  | <<The below are the properties of MessageQueueInfo.>> | | | |
| ReceiptNumber | String | Unique id assigned to this message. | a-7457 |
| MessageType | String | The type of message whether it would be a TextMessage or Email. See [the list of message types](#Message_Type). | Email |
| Status | String | The current status of the message. The system sends text messages async. The initial status is “Pending” or “Sending”. The further status will be updated via CallbackURL.  See the [list of Status](#MessageQueueStatus). | Sent |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": "b-a-405",

"totalMessageCount": 2,

"status": "Complete",

"createdOn": "2021-04-20T17:48:16.41+00:00",

"messageQueueInfo": [

{

"receiptNumber": "a-7956",

"messageType": "Email",

"status": "Delivered"

},

{

"receiptNumber": "a-7957",

"messageType": "Email",

"status": "Sent"

}

]

}

}

#### CheckInvalidMobileNumber

Retrieve detailed information related to the bulk message queue id and the statuses of the messages that were sent.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/CheckInvalidMobileNumber[?mobileNumber**](https://CourtNotifyAPI.lacourt.org/msg/CheckInvalidMobileNumber%5b?mobileNumber)**]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| MobileNumber | String | Recipient’s Mobile Number. | 12099657121 |

##### RETURNS

Returns information regarding invalid phone.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| Id | Integer | Identifier of a Phone record. | 1 |
| Invalid | Bool | 1: Phone number is invalid 0: Phone number is valid Null | 1 |
| InvalidCheckedOn | DateTime | The date and time that the Phone number became invalid. | 2020-10-14 15:53:10.883 |
| InvalidCode | Integer | The code from our [Phone Invalid Reasons](#Phone_Invalid_Reason) describes why the phone is invalid. | 10 |
| InvalidReason | String | This is the description of why the phone number is considered invalid. | The carrier failed to deliver to this phone number with an unknown error. |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"invalid": true,

"invalidCheckedOn": "2021-07-09T22:56:06.377",

"invalidCode": 10,

"invalidReason": "The carrier failed to deliever to this phone number with an unknown error."

}

}

#### CheckInvalidEmailAddress

Retrieve detailed information related to the bulk message queue id and the statuses of the messages that were sent.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/msg/[version]/CheckInvalidEmailAddress[?emailaddress**](https://CourtNotifyAPI.lacourt.org/msg/CheckInvalidEmailAddress%5b?emailaddress)**]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| EmailAddress | String | Recipient Email Address | mlee@lacourt.org |

##### RETURNS

Returns data from BulkMessageQueue, and MessageQueue.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| Id | Integer | Identifier of the Email Id. | 1 |
| Invalid | Bool | 1: Email address is invalid 0: Email address is valid Null | 1 |
| InvalidCheckedOn | DateTime | The date and time that the Email became invalid. | 2020-10-14 15:53:10.883 |
| InvalidCode | Integer | The code from our [Email Invalid Reasons](#Email_Invalid_Reason) describes why the email is invalid. | 2 |
| InvalidReason | String | This is the description of why the email is considered invalid. | The recipient's mailbox does not exist. |

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"bulkReceiptNumber": "b-a-405",

"totalMessageCount": 2,

"status": "Complete",

"createdOn": "2021-04-20T17:48:16.41+00:00",

"messageQueueInfo": [

{

"receiptNumber": "a-7956",

"messageType": "Email",

"status": "Delivered"

},

{

"receiptNumber": "a-7957",

"messageType": "Email",

"status": "Sent"

}

]

}

}

### ShortURL API

It’s good practice to keep text messages short. URL must be shortened to be embedded in the text message to keep it short. This API provides utility to a short URL and the system provides a redirection service.

\*\*Notice: Before using, if your domain is not part of the list below, please let the [System Admin](#Support_Contact) know.

Allowed Domains: "lacourt.org, lasuperiorcourt.org, azurewebsites.net"

Versioning information can be found under [Version](#Version).

* [GetShortUrl](#_GetShortUrl)

#### GetShortUrl

Converts a Long URL into a short URL.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **GET** | **https://CourtNotifyAPI.lacourt.org/surl/[version]/GetShortURL[?longURL]** |

##### PARAMETERS

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| longURL | String | Normal URL | <http://www.lacourt.org/> |

##### RETURNS

Returns with the ShortURL.

**Response Example**

{

"url": "https://s.lacourt.org/QmTcJ31",

"successful": true,

"errorMessage": null

}

### VerificationCode API

This API generates a new verification code for Mobile numbers and Emails.   
This API can be used to put the minimum security on your CallbackURL. You call ‘GetCode’ to generate a code and put the code into your CallbackURL while sending a message. When you receive a call on CallbackURL for the status update of the message, you can check if the URL contains the same code. Then, you can confirm this call is a legitimate call from the Court Notify system for the message.

Versioning information can be found under [Version](#Version).

* [CreateVerificationCode](#_CreateVerificationCode)
* [GetVerificationCode](#_GetVerificationCode)

#### CreateVerificationCode

This operation is used to create new Verification Codes.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | **https://CourtNotifyAPI.lacourt.org/code/[version]/CreateVerificationCode** |

##### Post Body

* Name: “verificationCodeRequest”
* Type: VerificationCodeRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| <<The below are the properties of VerificationCodeRequest >> | | | |
| AuthFor | String | Any value this Verification Code is used for such as a mobile number, the phone number is entered. | 12099657121 |
| AuthForType | String | This could be any value that the Service Provider can use it for such as “MobileNumber” or “Email” | MobileNumber |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| ValidDurationInMins | Integer | This is the time in minutes used to calculate the Expiration time. | 480 |
| AccessedIp (optional) | String | The IP address was used when the Verification Code was accessed on. | 127.0.0.1 |

**Parameter in Body Example**

{

"AuthFor":"12099657121",

"AuthForType": "MobileNumber",

"ServiceId": 5,

"ValidDurationInMins": “10”

}

##### RETURNS

Returns information regarding the Verification Code.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| AccessedOn | DateTime | The date and time that the code was accessed. | 2020-02-19T15:44:41.2344416+00:00 |
| CreatedOn | DateTime | This is the date and time the code was created. | 2020-02-18T16:23:00.693+00:00 |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the Return for any reason.

**Response Example**

{

    "successful": true,

    "warning": null,

    "error": null,

    "value": {

        "serviceInfo": null,

        "code": "3c387c73-4ccd-479e-b5c4-820b283b2f51",

        "expireOn": "2020-02-19T16:17:44.3998787+00:00",

        "accessedOn": "2020-02-19T16:07:44.3998947+00:00",

        "accessedIP": null,

        "createdOn": "2020-02-19T16:07:44.3998816+00:00",

        "authFor": "12099657121",

        "authForType": "MobileNumber",

        "serviceId": 5

    }

}

#### GetVerificationCode

This operation is used to retrieve the information related to the verification code.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT TEMPLATE** |
| **POST** | [**https://CourtNotifyAPI.lacourt.org/code/[version]/VerifyCode**](https://CourtNotifyAPI.lacourt.org/code/%5bversion%5d/VerifyCode) |

##### Post Body

* Name: “getVerificationCodeRequest”
* Type: GetVerificationCodeRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| <<The below are the properties of GetVerificationCodeRequest >> | | | |
| Code | String | This is the Verification Code that will be generated. | 3f6a4f54-ce4d-46d3-a9f6-0ec081a852d3 |
| ServiceId | Integer | Registered services. A service is a group of messages. At least one service is required to register per service provider. You can manage services in Service Provider Portal, and get this id. | 1 |
| AccessedIP | String | The IP address was used when the Verification Code was accessed on. | 127.0.0.1 |

**Parameter in Body Example**

{

"ServiceId": 5,

"Code": “1B5EF718-C428-4D23-B4B8-D6314C8F623B”

}

##### RETURNS

Returns information regarding the Verification Code.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Example** |
| ExpireOn | DateTime | The date and time that the code expires. | 2020-04-18T17:23:00.693+00:00 |
| AccessedOn | DateTime | The date and time that the code was accessed. | 2020-02-19T15:44:41.2344416+00:00 |
| CreatedOn | DateTime | This is the date and time the code was created. | 2020-02-18T16:23:00.693+00:00 |
| AuthFor | String | This is the value that was used. Example “2” is the ID from ‘MessageQueueId’ or you can use a mobile number value. | 12099657121 |
| AuthForType | String | This could be any value that the Service Provider can use for its usage. | Mobile Phone |

\*\* This list excludes the properties which are the parameter properties passed by the caller. Return sometimes includes the parameters passed for the caller to verify or use the parameter values with the Return for any reason.

**Response Example**

{

"successful": true,

"warning": null,

"error": null,

"value": {

"serviceInfo": null,

"code": "1b5ef718-c428-4d23-b4b8-d6314c8f623b",

"expireOn": "2020-04-18T17:23:00.693+00:00",

"accessedOn": "2020-02-19T16:22:34.9962871+00:00",

"accessedIP": null,

"createdOn": "2020-02-18T16:23:00.693+00:00",

"authFor": "2",

"authForType": "MessageQueueId",

"serviceId": 5

}

}

### HelloAuthorizedWorld API

Call this API for the first-time API connection test with the access token.

* [Hello](#_Hello)

#### Hello

This operation is used for connection with authentication information.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/HelloAuthorizeWorld/Hello**](https://CourtNotifyAPI.lacourt.org/HelloAuthorizeWorld/Hello) |

##### PARAMETERS

None

##### RETURNS

Returns a string message with the Client ID.

**Response Example**

"Hello authorized user! Message at 1/31/2020 5:10:48 PM. Your client id is fab2897a-e2ad-4fb3-b76b-d0888a00a392."

### HelloWorld API

Connection test API for the first time call without an access token.

* [Hello](#_Hello_1)

#### Hello

This operation is used to connect without authentication information.

|  |  |
| --- | --- |
| **VERB** | **ENDPOINT** |
| **GET** | [**https://CourtNotifyAPI.lacourt.org/HelloWorld/Hello**](https://CourtNotifyAPI.lacourt.org/HelloWorld/Hello) |

##### PARAMETERS

None

##### RETURNS

Returns a welcome message.

**Response Example**

"Hello! You called this web api successfully at 1/31/2020 5:11:45 PM."

## API Client

CourtNotify provides information on Web API and our client library for .NET.

### API Client Library for .NET NuGet package

CourtNotify team provides a ready-to-go NuGet Package for .NET developers. Client Library is available for .NET Standard and .NET Framework.  
  
The library handles OAuth 2.0 access token protocol and implements the best practice to consume Rest API. It converts API results to objects; all object class definitions are included in the library.

[Once initialized](#_Authentication), you won’t have to reinitialize again as long as your application is still running.

#### Install

There are two options Service Providers can use to download our NuGet packages.

##### Option 1: Install through Visual Studio

1. Open Visual Studio
2. Go to Manage NuGet Package
3. Search for “courtnotify”

Graphical user interface, application

Description automatically generated

1. Install the package that you want
   1. CourtNotify.ClientLibrary: This is the .NET Standard 2.0.
   2. CourtNotify.ClientLibrary.NETFramework: This is the .NET Framework.

##### Option 2: Download through the NuGet website

1. Go to <https://www.nuget.org/>
2. On the Search box, type in “courtnotify”

Diagram, engineering drawing

Description automatically generated

1. There are two packages:
   1. CourtNotify.ClientLibrary: This is the .NET Standard 2.0.
   2. CourtNotify.ClientLibrary.NETFramework: This is the .NET Framework.

#### NuGet Change History for each Version

##### Standard (CourtNotify.ClientLibrary)

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 08/04/2022 | 1.1.0 | Removed RestSharp dependency due to a vulnerability and replaced it with Microsoft libraries. |
| 07/22/2022 | 1.0.0 (Deprecated) | Initial version. |

##### Framework (CourtNotify.ClientLibrary.NETFramework)

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 08/12/2022 | 1.2.0 | This version 1.2.0 fixes a deadlock issue from the previous version 1.1.0. |
| 08/04/2022 | 1.1.0 | Removed RestSharp dependency due to a vulnerability and replaced it with Microsoft libraries. |
| 07/22/2022 | 1.0.0 (Deprecated) | Initial version. |

#### Code sample

Code samples utilizing the client library NuGet package are available to download from the [Service Provider website](https://courtnotifysp.lacourt.org/).

##### Set Account Information

To call Web API, all you need to do is declare the client id, secret, and subscription key.

Call the‘ BaseAPI.**Initialize()**’ method from where your application launches. Then, the client library takes care of authentication for you. This is only necessary one time in your application life cycle. TimeOffSet can now be specified for date and time.

BaseAPI.Initialize(

           clientID: "<<Copy and paste your client id here>>",

           secret: "<<Copy and paste your secrete here>>",

           subscriptionKey: "<<Copy and paste your development subscription here>>",

           timeOffset: new TimeSpan(-7, 0, 0));

##### Send Text Message

Call ‘MessageAPI.**SendTextMessage()’** method. It returns an ID that can be used to check the status.

Error property contains detailed information about why it failed. See the [‘Error Codes’ section](#API_Errors) for detail.

#region SendTextMessage

           var textReceiptNumber = string.Empty;

           var textMessageRequest = new TextMessageRequest

           {

               ServiceId = ServiceId,

               MessageGroupId = MessageGroupId,

               MessageGroupTitle = MessageGroupTitle,

               MobileNumbers = new List<string> { RecipientMobileNumber },

               TextMessage = TextMessage,

               CallbackURL = callbackURL,

               RequestResponseInText = new RequestResponseInText

               {

                   RequireResponse = true,

                   ResponseValidUntil = DateTime.UtcNow.AddDays(3),

                   RespondOptionList = new List<TextRespondOption> {

                       new TextRespondOption {RespondText = "Yes", RespondValue = "Y"},

                       new TextRespondOption {RespondText = "No", RespondValue = "N"}

                   }

               }

           };

           APIResult<List<TextMessageResult>, APIError> apiResultText  = MessageAPI.SendTextMessage(textMessageRequest);

           if (apiResultText.Successful)

           {

               //Save ReceiptNumber to check its status later.

               textReceiptNumber = apiResultText.Value.FirstOrDefault().ReceiptNumber;

           }

           else //Faile to send

           {

               if (apiResultText.Error.Code == ((int)APIErrorCodeEnum.InvalidPhoneNumber).ToString())

               {

                   //It failed to send the text message because it's invalid phone number.

                   //See Error.Description and Error.Message for more information.

                   string errorCodeDescription = apiResultText.Error.Description;

                   string errorDetail = apiResultText.Error.Message;

               }

           }

           #endregion

##### Send Email Message

Call ‘MessageAPI.**SendEmailMessage()’** method. It returns an ID that can be used to check the status.

#region SendEmailMessage

         var emailReceiptNumber = string.Empty;

         var pdfFileName = "LASCSiteLogo1.pdf";

         var pdfFileBytes = File.ReadAllBytes(pdfFileName);

         var pdfBase64String = pdfFileBytes.ToBase64StringFromBytes();

         var emailMessageRequest = new EmailMessageRequest

         {

             ToEmails = new List<string> { RecipientEmail },

             EmailSubject = EmailSubject,

             EmailBody = EmailBody,

             ServiceId = ServiceId,

             MessageGroupId = MessageGroupId,

             MessageGroupTitle = MessageGroupTitle,

             EmailAttachments = new List<EmailMessageAttachment>

             {

                 new EmailMessageAttachment

                 {

                     FullName = pdfFileName,

                     Data = pdfBase64String,

                     MetaData = new EmailMessageAttachmentMetaData

                     {

                         ContentEncoding = Encoding.ASCII.WebName,

                         ContentType = MediaTypeNames.Application.Pdf

                     }

                 }

             }

         };

         APIResult<List<EmailMessageResult>, APIError> apiResultEmail

             = MessageAPI.SendEmailMessage(emailMessageRequest);

         if (apiResultEmail.Successful)

         {

             //Save ReceiptNumber to check the status later by calling GetMessageQueue.

             emailReceiptNumber = apiResultEmail.Value.FirstOrDefault().ReceiptNumber;

         }

         else //Faile to send

         {

             if (apiResultEmail.Error.Code == ((int)APIErrorCodeEnum.OptOut).ToString()) {

                 //It failed to send email message because Email Address Opt Out.

                 //See Error.Description and Error.Message for more information.

             }

         }

         #endregion

##### Check Message Status

Call ‘MessageAPI.**GetMessageQueue()**’ method. It returns the MessageQueueInfo object which contains all information of the message along with message status (StatusEnum property).

#region GetMessageQueue

           APIResult<MessageQueueResult, APIError > apiResultMessageQueue

               = MessageAPI.GetMessageQueue(textReceiptNumber);

           if (apiResultMessageQueue.Successful)

           {

               //Message Queue information was retrieved successfully.

               MessageQueueResult messageQueue = apiResultMessageQueue.Value;

               //Can check its status

               MessageStatusCd status = messageQueue.StatusEnum;

           }

           else // Failed to retrieve

           {

               if (apiResultMessageQueue.Error.Code == ((int)APIErrorCodeEnum.RelatedRecordIsNotFound).ToString())

               {

                   //Failed to retrieve the information.

                   //See Error.Description and Error.Message for more information.

               }

           }

           #endregion

##### Retrieve Message Log

Call ‘MessageAPI.**SearchMessageQueue()**’ method to search messages. It returns multiple message records matching the specified criteria.

#region SearchMessageQueue

           var messageQueueRequest = new MessageQueueRequest

           {

               FromDate = DateTime.Now.Date.AddDays(-1),

               ToDate = DateTime.Now

           };

           APIResult<List<MessageQueueResult>, APIError> apiResultMessageQueueLog

               = MessageAPI.SearchMessageQueue(messageQueueRequest);

           if (apiResultMessageQueueLog.Successful)

           {

               //Message Queue informations was retrieved successfully.

               foreach(MessageQueueResult m in apiResultMessageQueueLog.Value)

               {

                   //read the information.

                   MessageStatusCd status = m.StatusEnum;

               }

           }

           else

           {

               //It failed to send text message.

               if (apiResultMessageQueueLog.Error.Code == ((int)APIErrorCodeEnum.RelatedRecordIsNotFound).ToString())

               {

                   //Failed to retrieve the information.

                   //See Error.Description and Error.Message for more information.

               }

           }

           #endregion

##### Others

There are other methods available to check opt-in/out information. Please see the APICallExample.cs file for all examples in the library.

### Call Limit Policy

Depending on the Product ‘Development’ or ‘Production’ you are subscribed to, there is a maximum number of calls you can make to the API to send messages. This helps API users use the right API operation and prevents accidental API misuse.

For example, to send many messages at once by a nightly batch job must you use Bulk API ‘[SendTextMessageInBulk](#SendTextMessageInBulk)’ or ‘[SendEmailMessageInBulk](#SendEmailMessageInBulk)’, which is counted as only one call although you include 100K messages, instead of calling ‘[SendTextMessage](#SendTextMessage)’ or ‘[SendEmailMessage](#SendEmailMessage)’ for an individual message which costs big in network communication and increases failure rate.

If you need to use a higher volume because Bulk doesn’t work for you, then you may request a higher than normal production policy. Contact [System Admin](#Support_Contact) if you would like to use ‘HigherVolumeProduction’.

#### Development

Calls to API: 30 calls / 60 seconds

The lower call limit in the ‘Development’ subscription prevents accidental API misuse during development time.

#### Production

Calls to API: 300 calls / 60 seconds

#### HigherVolumeProduction

Calls to API: 3000 calls / 60 seconds

### Error Handling

API returns an error with detailed information. Here is the information on how to analyze an error. For more details on API Errors, go to [API Errors](#API_Errors).

#### Error Object

APIResult’s Error property contains:

* Code property: It matches the ‘Code’ value in the code list.
* Description property: It matches the ‘Description’ value in the code list.
* Message property: This contains further detailed information for the specific API call incident. This provides information for a developer to fix a bug expediently.

##### Code Sample with Client Library

When you get a returned object from an API call, you need to check the error with the ‘Error’ property like below.

if (apiResultText.Error.Code == ((int)APIErrorCodes.InvalidPhoneNumber).ToString())

{

    //It failed to send the text message because it's an invalid phone number.

    //See Error.Description and Error.Message for more information.

    string errorCodeDescription = apiResultText.Error.Description;

    string errorDetail = apiResultText.Error.Message;

}

### Warning Messages

Warnings messages are to notify Service Providers of a minor issue with their request JSON, but the message will still be sent through.

Example: If a Service Provider provides a duplicated email in their ‘ToEmail’ request, we will respond with this warning message: “Duplicated ‘ToEmails’: [mlee@lacourt.org](mailto:mlee@lacourt.org)

|  |
| --- |
| **Warning Messages** |
| Duplicated ‘ToEmails’ |
| ’FriendlyFromName is discarded because ‘FromEmail’ was not specified. |
| ’ReplyToName’ is discarded because ‘ReplyToEmail’ was not specified. |

### Specifying TimeOffset

Service Providers can specify the time offset using the Client Library to any time zone they would like to use. If not specified, then it will return the date and time in PDT.

Text

Description automatically generated

## Value Tables

List of values and their meaning specified in API.

### Message Type

Type of messages that the system supports.

|  |  |
| --- | --- |
| **Code** | **Name** |
| 1 | TextMessage |
| 2 | Email |

### MessageQueue Status

The status of a queued message.

|  |  |  |
| --- | --- | --- |
| **Code** | **Name** | **Definition** |
| 0 | PrePending | New MessageQueue record that has been created with only minimal validation. |
| 1 | Pending | Default status before calling notification service. |
| 2 | Sending | Before notification services are called, this is the initial status. |
| 3 | Sent | For text messages and emails, this status identifies that the message has been sent to the Notification Service. |
| 4 | Temporary Failed | This status is only temporary and will not be the final status for messages. |
| 5 | InProgress | This status shows that the MessageQueue record is being processed by our WebJob. |
| 7 | Delivered | Final status for text & email messages if they have been successfully delivered to their recipient. |
| 8 | Discarded | This message is not valid to be sent to the notification services. |
| 9 | Failed | The message has failed to send and an Error Message will display the reasons why. |

### Phone Invalid Reason

The list of reasons why a phone number was considered invalid.

|  |  |  |
| --- | --- | --- |
| **Code** | **Name** | **Description** |
| 1 | Twilio\_21612 | The phone number provided is not reachable via text message. |
| 2 | Twilio\_30004 | The phone number you are trying to reach has been blocked from receiving messages. |
| 3 | Twilio\_21401 | The phone number you specified was not a valid text message-enabled phone number. |
| 4 | Twilio\_21407 | This phone number does not support text messaging. |
| 5 | Twilio\_21614 | This phone number may be a landline number or formatted incorrectly. |
| 6 | Twilio\_30006 | Carrier failed to deliver a message to the landline or unreachable carrier. |
| 7 | Twilio\_63033 | The recipient has been blocked from receiving messages. |
| 8 | Twilio\_30003 | The phone number you are trying to reach is switched off or otherwise unavailable. |
| 9 | Twilio\_30005 | The phone number you are trying to reach may no longer exist. |
| 10 | Twilio\_30008 | The carrier failed to deliver to this phone number. |
| 11 | Twilio\_21211 | The phone number provided has an invalid format. |

### Email Invalid Reason

This is the list of why some email addresses are considered invalid. You may see other error codes for emails, but they are only temporary and will be retried again. If the error code has a permanent failure, the email is considered invalid and will not be retried.

If you see an error code for “250”, that email for Mailgun has been successfully delivered.

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Name** | **Description** | **Is the status a Permanent failure?** |
| 1 | Mailgun\_552 | The recipient's mailbox has exceeded its limits. | No |
| 2 | Mailgun\_550 | The recipient's mailbox does not exist. | Yes |
| 3 | Mailgun\_554 | The recipient’s email box is no longer in use. | Yes |
| 4 | Mailgun\_605 | The recipient’s mail server has rejected the email message. | Yes |
| 5 | Mailgun\_607 | The recipient has marked your message as spam. | No |
| 6 | Mailgun\_612 | The email provider was unable to connect to the mail server to send out the email. | No |
| 7 | Mailgun\_498 | The recipient's mailbox does not exist. | Yes |
| 8 | Mailgun\_511 | The recipient's mailbox does not exist. | Yes |
| 9 | Mailgun\_450 | The recipient's mailbox does not exist. | Yes |
| 10 | Mailgun\_499 | The recipient's mailbox does not exist. | Yes |

### API Errors

These are the list of errors returned when you call our API. Some API Errors are caused by specific MessageQueue Errors. For more details see [MessageQueue Errors](#MessageQueueErrors) below.

|  |  |
| --- | --- |
| **Code** | **Description** |
| 100 | Required value is missing. |
| 101 | Related record is not found. |
| 103 | Mismatch record |
| 107 | Duplicated Entry |
| 108 | Child Constraint |
| 110 | Opt-Out |
| 111 | Removed Service or Message Group |
| 113 | Invalid Parameter value |
| 114 | Expired |
| 115 | Deactivated Service or Message Group |
| 116 | Message Size Limit Exceeded |
| 117 | Blocked |
| 120 | Update Failed |
| 200 | Failed to send message |
| 500 | Error |
| 301 | Email File Layout Error |
| 302 | Text Message File Layout Error |

### MessageQueue Errors

List of error codes that describe what each error message is.

|  |  |
| --- | --- |
| **ErrorCode** | **Description** |
| 100 | Invalid Phone Number |
| 101 | Deactivated Service or Message Group |
| 102 | Related record is not found. |
| 103 | Invalid Client Id |
| 104 | Invalid Service |
| 105 | Duplicate Entry |
| 106 | Invalid Name |
| 107 | Invalid Email Address |
| 108 | Limit Exceeded |
| 109 | Invalid Domain Name |
| 110 | Twilio Status Update Failed |
| 111 | Email Error |
| 112 | Expired |
| 113 | Invalid Message Group |
| 114 | Mismatch record |
| 115 | Invalid Message Id |
| 116 | Invalid CallBackURL |
| 117 | Invalid Entry |
| 118 | Email Message Status Update Failed |
| 119 | Removed Service or Message Group |
| 120 | Failed to Send Message |
| 200 | Customer Opt-Out |
| 201 | Blocked |
| 300 | Required parameter is missing |
| 500 | Error |

### HTTP Requests Status Codes

Commonly used Status Codes.

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Description** | **Explanation** |
| 200 | OK | The request has succeeded. |
| 429 | Too Many Requests | Too many calls to the API have been made within the allowed time. |
| 400 | Bad Request | Cannot process the request due to an error on the user side e.g. invalid syntax |

Reference: <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

## Data Type Common Rule

### Date Time

* DateTime is returned in UTC, but when initializing the Client Library, we can provide the offset (Timespan) to convert the time into any timezone.
  + See the example in [How to specify a time offset in Client Library](#TimeOffSet)

### Phone Number

* The phone number is saved and returned with a numeric value only.

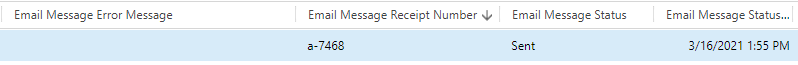
### Email Attachments

Service Providers can send emails with attachments using [SendEmailMessage](#SendEmailMessage) and [SendEmailMessageInBulk.](#SendEmailMessageInBulk)

## CallBack URL

CallBackURL is used to update the status of the message that was sent to the recipients.

Example URL: <https://crmportalqa.lacourt.org/Appointment/MessageStatus?receiptNumber=a-7468>&status=sent



1. From the Example URL that was provided, we are appending the following fields:
   1. [receiptNumber](#_RETURNS)
      1. This is a unique identifier of the message.
      2. Type: String
   2. Status
      1. Value: Pending, Sending, Sent, Delivered, Discarded, and Failed
      2. Type: String
   3. errorMessage
      1. Value: Any type of error message that can occur during the operation.
      2. Type: String
   4. respondValue
      1. Value: If Service Provider is using RequestResponse, then it will return the message that the recipient replied back with.
      2. Type: String
2. Examples:
   1. <https://crmportalqa.lacourt.org/Appointment/MessageStatus?reminderId=0a37d3f7-9986-eb11-8137-005056a30fb9&receiptNumber=a-7480&status=delivered&err>orMessage
   2. <https://crmportalqa.lacourt.org/Appointment/MessageStatus?reminderId=0a37d3f7-9986-eb11-8137-005056a30fb9&receiptNumber=a-7480&status=failed&errormessage=Removed> Service or Message Group
3. This is a practical example of how Court Notify uses callback URL with their reminder notifications. The receipt number is saved with the status of that email message and the date and time it was updated.

## Using Website instead of API calls

‘[Service Provider Website](https://courtnotifysp.lacourt.org/)’ is available for non-developer end-users to view logs and send messages without programming API calls.

The same ‘[Service Provider Website](https://courtnotifysp.lacourt.org/)’ is also used by a developer to manage API accounts and services. The features provided on the website might be limited and don’t support all API features.

### Service Provider

This is the home page for Service Providers when they are logged in.

#### How to download guides?

1. Go to the Help Guides header.
2. Click “Download Court Notify..” help guides.

Graphical user interface, text, application, email

Description automatically generated

#### How to download .NET Client libraries?

1. Go to the .NET Client Library header.
2. Click “Download CourtNotify API Client Library Using .NET..”

Graphical user interface, text, application

Description automatically generated

#### How to contact System Admin?

1. Scroll to the bottom of the page where the header says System Admin Contacts.

### Services

SP users can easily add services to use. Services are individual applications that belong to your organization.

Example: Los Angeles Superior Court has an Appointment Service, Traffic Payment Plan Service, etc.

Changes made on the Service page may take up to 15 minutes due to caching.

#### How to add a Service?

1. Go to the Services tab.
2. Click on the “**+ Add**” button on the right.

Graphical user interface, text, application, email

Description automatically generated

1. Fill out the fields with information on your service.

Graphical user interface, text, application, email

Description automatically generated

* 1. Title: This is the name of the Service that you are providing to recipients.

Example: “Appointment”, “Traffic Payment Plan”, “Jury Notification Service”

* 1. Description: Short description of what your service offers.

Example: “Traffic Payment Plan is a service that sends enrollment confirmations to recipients and payment reminders.”

* 1. Friendly From Email Name: If you want to use your own “From Email Friendly Name”, then you can specify the name here that recipients will see instead of the email address. Example: “Appointment” (Optional)
  2. From Email: This is the email address that you want your recipients to see. Example: [OCCAppointment@occ.org](mailto:OCCAppointment@occ.org) (Optional)
  3. Friendly Reply-to Email Name: If you want a different email for recipients to reply to, then add a “Friendly Reply-to Email Name”. Example: NO-REPLYOCC
  4. Reply-To Email: This is the email address that recipients will reply to if specified. Example: “NO-ReplyOCC@occ.org”
  5. Use Message Group: If checked, your Service must use [Message Groups](#_Message_Group). After this is saved, it cannot be changed.

This allows recipients to Opt-Out of the specific Message Group instead of the entire Service as a whole.

Example: This will prevent recipients from Opt-Out of the entire Service as a whole, they can only Opt-Out of that certain Message Group like for different Jury Cases. If the Service Provider wants to create a generic Notification group, they can create a new message group called “Traffic Payment Plan-General”.

1. Click on the **Save** button for your changes to create a new service.

#### How to delete a Service?

This is used to remove a Service that you might have accidentally created or a Service that is not in use.

1. On the Services page, you will see on the right of the table a “Delete” link button.
2. This button will only appear if the Service did not send any messages out yet.
   1. If the Service has messages that were sent out already, you can only “Disable” the service.

Graphical user interface

Description automatically generated with medium confidence

1. Click on the “Delete” link button to permanently delete the service.

Graphical user interface

Description automatically generated

#### How to edit a Service?

If you need to make changes to a Service, you can edit the features here.

1. On the Services page, on the right side of the table click on the “Edit” link button.

Graphical user interface

Description automatically generated with medium confidence

1. Click on the textbox that you want to edit.

Graphical user interface, text, application, email

Description automatically generated

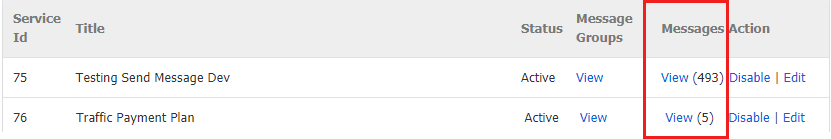
|  |  |
| --- | --- |
| **Field Name** | **Description** |
| Id | The registered service under which this message is sent. Service is the default group of messages, called a subscription that the recipient can Opt-in or out. |
| From Email with Friendly Name will be | This is the friendly From Email Name + From Email address that will be displayed to the recipient. |
| Reply-to Email with Friendly Name will be | This is the friendly Reply-to Email Name + Reply-to Email that will be displayed to the recipient. |
| Use Message Group | Yes: This service uses Message Groups.  No: This service does not use Message Groups. |
| Status | Active: The service is being used and can send messages.  Inactive: The service is not being used and cannot send messages. |
| Created On | The date and time the service was created. |
| Updated On | The date and time the service was updated. |

\*\* This list excludes the properties mentioned in “[How to add a Service?](#_How_to_add)”.

1. “Save” your changes.

#### How to view Messages related to the Service?

1. Look for the column “Messages” on the right of the table.
2. Click the link button “View” to see messages related to the Service on the [“Message Log” page](#_Message_Log_1).



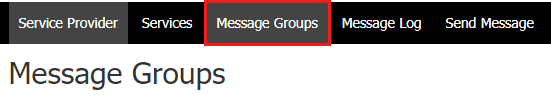
### Message Groups

SP users can view [message groups](#_Message_Group) that are being in use from Services.

Example: You might have a service called Traffic Payment Plan with different traffic tickets as message groups such as “Traffic Ticket #T34F34, Traffic Ticket #T3434D”.

#### How to view Message Groups?

1. Go to the Message Groups tab.

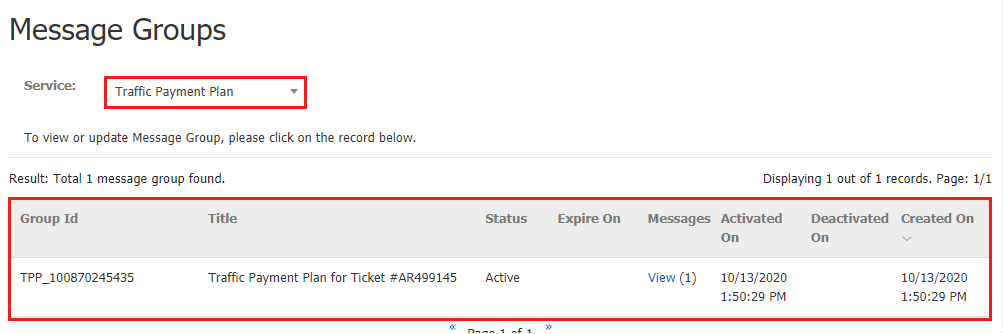


1. Click on the dropdown box with the placeholder “Select a Service”.

Graphical user interface, application

Description automatically generated

1. Choose which Service you want to view your Message Group(s) and the result(s) will be displayed below.



#### How to see more details for a Message Group?

1. Select a Service from the dropdown menu.
2. On the results displayed, click anywhere on the record that you want to see more details.

Graphical user interface, text, application

Description automatically generated

1. More details will be shown in the Message Group Detail popup.

#### How to download Message Groups?

1. Filter your items by Service and the date range.
2. Click on “Export to CSV” to retrieve your exported file.

Graphical user interface, text, application

Description automatically generated

#### How to edit the Expiration time for a Message Group?

1. When the Message Group Detail popup is opened, click on the text boxes where ‘Expire On:’ is located to change the date and time you want the Message Group to expire. Graphical user interface, application

   Description automatically generated

#### How long can I view my Message Groups for?

1. View the Message Groups page and look for the red text on top.
2. Then check the records displayed and look for the ‘Expire On’ date. This is how long the Message Groups will be available.

Graphical user interface, application

Description automatically generated

### ­­Message Log

View messages that your services have sent out or replies from your recipients. Service Providers can use the [“Filter” button](#_How_to_‘Filter’) to see all of the messages from certain criteria or use the [“Search” button](#_How_to_‘Search’) by a specific keyword to find the certain message(s).

#### How to ‘Filter’ messages?

This is the default display when Service Provider opens up the page with the Date Now to 1 month ago. Service Providers can see their messages from certain criteria.

1. On the top panel, select the information you want to ‘Filter’.

Graphical user interface, application

Description automatically generated

1. Service: These are the Services that you have created. Examples: “Appointment”, “Event”, “Traffic Payment Plan” etc.
2. Message Type: Messages can be either “Email” or “TextMessage”.
3. From Date and To Date: Set the range you want to view messages.
4. Status: Default is All
   1. Status explanations can be found on [MessageQueue Status](#_MessageQueue_Status).
5. Message Group: Enter ‘messageGroupId’ or ‘messageGroupTitle’ to search for certain message groups.

e.g.)

1. Results appear below:

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Table Header** | **Description** |
| Receipt Number | Unique Id for messages. |
| Created On | The date and time the message was created. |
| Service | The name of the Service. |
| Message Type | The type of message whether it would be a TextMessage or Email. See [the list of message types](#_Message_Type_1). |
| Status | See the [list of status](#_MessageQueue_Status). |
| Status Updated On | The date and time the Status was updated on. |
| Message Group | Recipient-friendly text for MessageGroupId. |
| Recipient | Can be either a mobile number or an email address. |
| Message | Text Message or Email Body. |
| Bulk Receipt | Yes: This message was sent as part of a Bulk Message. No: This message was not sent as part of a Bulk Message. |
| Respond? | Yes: The recipient is required to respond to the message. No: No response is required from the Recipient. |

1. When you click on the record, it will pop up the Message Detail log.
2. Message Detail pops up more detailed information listed below.

A screenshot of a computer

Description automatically generated

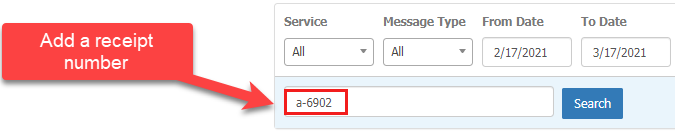
|  |  |
| --- | --- |
| **Field Name** | **Description** |
| Send Message Try Count | The number of attempts this message was retried on our batch job. |
| Requested to Hide Unsubscribe | Yes: The unsubscribe link is not available for this email message. No: The unsubscribe link is available for this email message. |
| Segment Count | One segment is 160 characters or less. One Text Message can be one or more segments. |
| Status Detail by Text Message Service | The status of the Text Message with the most common statuses are undelivered, delivered, and queued.  See a [common list of statuses](#_MessageQueue_Status). |
| Status Detail by Text Message Service Updated On | The date that the error message and detail were updated. |
| Error Code by Text Message Service | The error code is associated with the Status Detail by Text Message Service.  For more information, please visit the website [Twilio Common Error Codes](https://support.twilio.com/hc/en-us/articles/223181868-Troubleshooting-Undelivered-Twilio-SMS-Messages). |
| Reply-To Email | The email address that you specified for your recipient to reply to. |
| Status Detail by Email Service | The status of the Email Messages provided from MailGun.  See the [MailGun statuses](https://documentation.mailgun.com/en/latest/user_manual.html#events). |
| Status Detail by Email Service Updated On | The date that the error message and detail were updated. |
| Error Code by Email Service | The error code is associated with the Status Detail by Email Message Service.  More details will be added soon. |
| Message Group Id | Id to sub-group messages under the service. (i.e., Case Number) |
| Message Group Expire On | The date and time the Message Group will be expired and once expired it will change the status to inactive. |
| Message Group Status | Indicate if this Message Group is active or not.  False: MessageGroup is not active. True: MessageGroup is active. |
| Message Group Deactivated On | The date and time the Message Group was deactivated. |
| Message Group Activated On | The date and time the Message Group was activated. |
| CallBackURL | The URL that was passed from the Service Provider’s system will be updated with the Message-Id, status, and error message. |
| CallBack Try count | The number of times this URL was called to be updated. |
| Calback Last Called On | The date and time the CallBackURL was called. |
| Respond Text | The user-friendly text that a recipient replied with. |
| Response by Recipient | The values that are stored in DB when a recipient selects this option. |
| Bulk Message Queued On | The date and time, the Bulk Message was queued to our batch job to send out. |
| Message Count by the Bulk Message | The total amount of messages related to the Bulk Message group that was sent out. |
| Cost | Cost of this message. |
| Error Code | Unique id of error that this message has not been sent.    See [the list of message queue errors](#_MessageQueue_Error). |
| Error Message | A generic error message explaining what the Error Code means. |

\*\* This list excludes the properties mentioned on number 2 where the Table Header is explained.

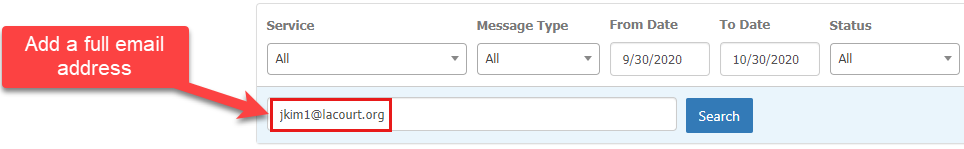
#### How to ‘Search’ for a message?

Service Providers can search for a specific keyword to find a record.

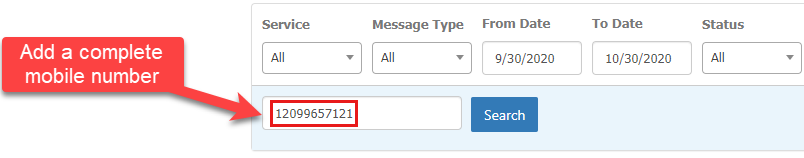
1. Search for a ‘Receipt Number, ‘recipient’s Email Address’, ‘recipient’s Mobile Number’ or ‘Bulk Receipt’ on the text box.
   1. Examples of ‘Receipt Number’, ‘Recipient email’, or ‘Recipient mobile number’ can be found in “[SendTextMessage Returns](#_RETURNS)” or “[SendEmailMessage Returns](#_RETURNS_1)”
   2. Example of what a ‘Bulk Receipt’ can be found in ‘[SendTextMessageInBulk Returns](#_RETURNS_3)’ and ‘[SendEmailMessageInBulk Returns’](#_RETURNS_4).
   3. How to search for a ‘Receipt Number’?



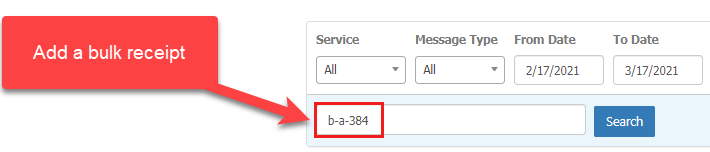
* 1. How to search for a ‘recipient’s Email Address’?



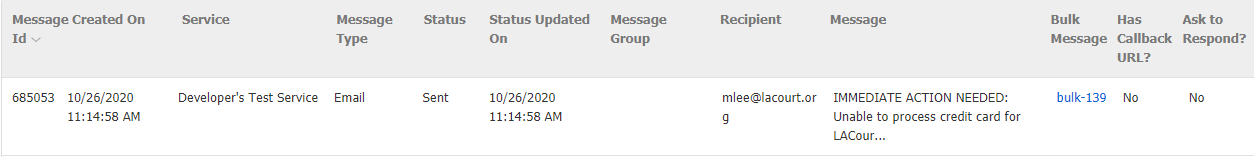
* 1. How to search for a ‘recipient’s Mobile Number’?



* 1. How to search for a ‘Bulk Receipt’?



1. Click on the “**Search**” button and the result will appear below:



#### How to download Message Logs?

1. On the Message Log page, click on “Export to CSV” to retrieve your exported file.

Graphical user interface, text, application

Description automatically generated

#### How long can I view my Message Logs?

1. View the Message Log page and look for the red text on top.
2. The red text will let you know how long you can view your Message logs.

Graphical user interface, application

Description automatically generated

### Directory

The directory lists the information of an email address or phone number in which a message has already been sent to.

#### View Directory

1. Click the ‘Directory’ label in the navigation bar.

Graphical user interface, website

Description automatically generated

2. Choose either ‘Email Address’ or ‘Mobile Phone’.

Graphical user interface, text, application

Description automatically generated

1. Enter the ‘Email Address’ or ‘Mobile Phone’ that you wish to search for.

Graphical user interface, text, application, email

Description automatically generated

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| Email/Phone ID | Unique email or phone ID for messages |
| Email Address/Mobile Number | Email address or phone number of the recipient |
| Invalid | Valid state of the email/phone (True/False) |
| Verified | Email/phone verified by user (True/False) |
| Verified On | Date email/phone verified by user |
| Invalid Checked By | The MessageQueue record associated with the status update (Sync-Status) |
| Invalid Checked On | Date marked invalid |
| Invalid Reason | Reason marked invalid |
| Invalid Changed By App User | The name of admin that changed to valid |
| Is Opt-In | Status of the user opt-in to messages |
| Opt-in On | Date of the user opt-in to messages |
| Opt-in Type | Type of messages, email or text |
| Opt-out On | Date opt-out of messages |
| Opt-out Type | Type of messages opt-out, email or text |

#### Change to Valid

If an email address or phone number is marked as invalid and you wish to change it to valid, click the blue check mark labeled ‘Change to Valid’ in the ‘Invalid’ field. This will allow Court Notify to retry when you send a message.

Graphical user interface, text, application, email

Description automatically generated

### Send Message

Service Providers (s) can easily send text messages or emails in bulk.

#### How to Send Bulk Online?

1. Navigate to the **‘Service Provider’** tab from the home page.

Graphical user interface, website

Description automatically generated

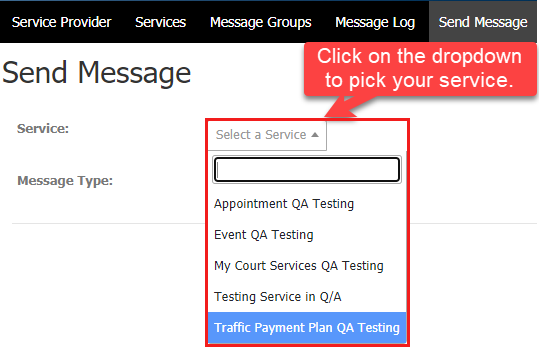
1. Click on the **‘Send Message’** tab.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. **Select** the **‘Service’** you want to use to send messages from.

e.g.) If you are sending emails from your Traffic Payment Plan service, use “Traffic Payment Plan”.



1. Select whether you want to send text message(s) or email(s).

Graphical user interface, application

Description automatically generated

* 1. For Email messages
     1. Download the ‘Email File Template.csv’ file.

Graphical user interface, text, application

Description automatically generated

* + 1. The optional fields in the .CSV are required headers even though the values under them can be empty.
    2. Fill out the corresponding columns.
       1. Email Recipients: Type in an email address in this format [email@lacourt.org](mailto:email@lacourt.org) in column A
       2. For multiple email addresses, you can type in column A using comma “,” or semicolon “;” mlee@lacourt.org;DVoong@lacourt.org
       3. Email Subject: The subject of your email

e.g.) Traffic Payment Plan Reminder

* + - 1. Email Body: This is your message to the email recipients. You can also use raw HTML to format your message. Websites like <https://html-online.com/editor/> can help format your HTML code properly.
      2. Message Group Id (Optional): If your [service has ‘UseMessageGroup’](#_Services_1) checked, then you must create a Message Group Id. Id to sub-group messages under the service. A case number is a common example of this.

e.g.) TPP\_100870245435

* + - 1. Message Group Title (Optional): If your [service has ‘UseMessageGroup’](#_Services_1) checked, then you must create a Message Group Title. Recipient-friendly text to see if they want to read what the message is about.  
         e.g.) Traffic Payment Plan for Ticket #AR499145
      2. Reply-to Email (Optional): If you require a Reply-to email for recipients to reply to, you can specify a Friendly Name with the Reply-to email here.  
         e.g.) Traffic Payment Plan Replies <TPPReplies@occourt.org>  
         e.g.) TPPReplies@occourt.org
      3. From Email (Optional): You can use your own FromEmail like “[occnotify@occourt.org](mailto:occnotify@occourt.org)” by registering their domain with Court Notify.   
         Please see the “[Why do I have to register our domain](#Why_do_I_have_to_register_our_domain)” section.  
         e.g.) NO-Reply <NO-ReplyOCC@occourt.org>  
         e.g.) occnotify@occourt.org

**Example of CSV File**

Graphical user interface, text, application, Word

Description automatically generated

**Example of CSV File with multiple recipients**

Text

Description automatically generated

* 1. For Text Messages
     1. Download the ‘Text Message File Template.csv’ file.

Graphical user interface, text, application

Description automatically generated

* + 1. Fill out the corresponding columns.
       1. Mobile Number Recipients: Type in mobile numbers using this format ‘12099657121’.
          1. For multiple mobile numbers, you can type in column A using comma “,” or semicolon “;” 12099657121;12344567891
       2. Message: This is your message to the Mobile Number recipients.  
          e.g.) You have an upcoming traffic ticket payment due on 12/25/20. Please submit payment to lacourt.org.
       3. Message Group Id (Optional): If your [service has ‘UseMessageGroup’](#_Services_1) checked, then you must create a Message Group Id. Id to sub-group messages under the service. A case number is a common example of this.

e.g.) TPP\_100870245435

* + - 1. Message Group Title (Optional): If your [service has ‘UseMessageGroup’](#_Services_1) checked, then you must create a Message Group Title. Recipient friendly text to see if they want to read what the message is about.

e.g.) Traffic Payment Plan for Ticket #AR499145

**Example of CSV File**

Graphical user interface, text, application, chat or text message

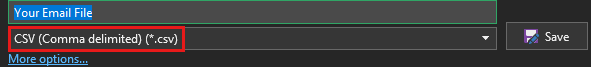
Description automatically generated

**Example of CSV File with multiple recipients**

Graphical user interface, text, application

Description automatically generated

1. **Save** your file as CSV.



1. Go back to the Send Message page and Click on the button “**Choose File**” to upload your CSV.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. **Review and Confirm** if everything is correct.
   1. Email

Graphical user interface, text, application

Description automatically generated

* 1. Text Message

Graphical user interface, text, application, email

Description automatically generated

1. After confirmation of your CSV file, click on the ‘**Send Text Message’ or ‘Send Email’** button.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

1. A confirmation message will display stating:

A picture containing text

Description automatically generated

1. The emails or text messages will be queued to our batch job which will be processed after 5 minutes.

Pleased by advised, if there are a lot of emails to be sent, delivery time may take longer than expected.

### Testing

### Browser Support

* We recommend using Google Chrome for the most supported UI.
* Internet Explorer I.E. is not supported on our Court Notify Website.

# Release Note

This section describes the history of the production release including the changes, bug fix detail, etc.

10/04/2022: Deploy fix to Client Library NuGet Package for .NET Framework update to version 2.1.

09/20/2022: Adding new V2 for ‘SendTextMessageInBulk’ and ‘SendEmailMessageInBulk’.

09/12/2022: Deploy fix to ‘SendTextMessageInBulk’ operation using the Client Library Standard version.

08/12/2022: Adding new NuGet package for .NET Framework 1.2.0

08/11/2022: Adding new NuGet package for .NET Core.

08/05/2022: Adding a fix to ServiceMonitor on production.

08/04/2022: Added NuGet package with new dependency (HttpClient) and removed RestSharp.

08/01/2022: Add updated Client Libraries to Service Provider website.

07/22/2022: Update Client Library with specifying version 1 using Nuget.

07/20/2022: Create version 1 on APIM Message API with all the existing collections and operations.

07/12/2022: Improve Web API by making the action methods ASYNC and fixing display issue on the Message Group page.

06/24/2022: Deploy fix to RequestResponse on handling multiple replies and update Azure certificates.

06/22/2022: Deploy fix to Sync Email Status regarding invalid emails and remove performance logging.

06/01/2022: Adding new table BulkMessageQueueItem and archiving and purging.

05/26/2022: Refreshed MessageGroup table due to caching when value changes are changed. Added Powerpoint to the Service Provider website for Service Providers to view.

05/23/2022: Enhanced logging on Production. Fixed CreatePhoneLookup issue that is throwing object reference not set.

05/19/2022: Add additional logging.

05/13/2022: Add additional logging.

05/12/2022: Add additional logging.

05/11/2022: Added additional logging for ProcessLogging.

05/09/2022: Added additional logging for ProcessLogging.

05/06/2022: Added missing 500 errorcode to MessageQueueAPIErrorCode table.

05/05/2022: Added additional logging for ProcessLogging, moved MessageQueueErrorCodes and APIError codes to the Database, and adding additional code to ping WebJobs instead of using logic app.

05/04/2022: Fixed issue where a recipient is replying with list commands and not receiving the correct reply back from CourtNotify.

05/02/2022: Adding logging for each step on bulk sending of emails and text messages.

04/25/2022: Added a new logic app to sync the ‘TextConversationLog’ Status and StatusUpdatedOn. Update MessageGroups with ExpireOn that are NULL to have an expire on date to be purged.

04/20/2022: Upgrading Twilio Node.js V12 to be V14.

04/15/2022: Fixed bug related to Stored Procedure for updating TextConversationLog and fixed SyncStatus issue on updating invalid email addresses with the correct Errorcode.

04/13/2022: Fixed bug for object not found on TextConversationlog. Fixed issue with handling commands that weren’t working on Production.

04/06/2022: Improved all untouched API operations and Webjobs.

03/28/2022: Adding Indexing and truncating on Production Database.

03/23/2022: Changed Purging logic to 90 days to enable it to start running.

03/22/2022: Add logging and purging enhancements & add the hotfixes which include fix to Custom SP, Public website, and Send Message operations.

03/09/2022: Add new field ‘CompletedOn’ to ‘BulkMessageQueue’ table.

03/07/2022: Added more logging on Custom SP website, fix incorrect ToDate displaying, fix exporting times, increased the size on unsubscribe page to be bigger, fixed SentOn not being updated if it failed, added ‘ErrorCode’ to callbackurl and made an overlay of the custom sp when loading pages.

03/04/2022: Adding a logic app to alert System Admin if Distributors go down and fix issue with SyncStatusFromTwilio regarding MessageQueue without a MessageId. Fixed issue on CreateVerificationCode with AuthForType. Deploy fix for UnknownValueException thrown with Mailgun.

03/03/2022: Adding a fix to BulkMessageQueue records inserting multiple MessageQueue records for textmessages.

03/02/2022: Added major changes to bulk request handling with new webjobs to distribute and send out messages.

02/14/2022: Added code to remove “on behalf of” on Mailgun.

02/07/2022: Add logging for Stored Procedure to see why a nonbulk message is picked up by a webjob

02/02/2022: Added code to SendTextMessage, SendEmailMessage, and stored procedures to make status as “sending” earlier.

02/01/2022: Modified GetPhoneOptInOut and GetEmailOptInOut to retrieve records that aren’t expired.

01/26/2022: Updated Email deliverability by adding HTML tags in html body, text version of a html email, and list-unsubscribe header.

01/21/2022: Removed logic to mark records as discarded in Sync Email/Text Message.

12/29/2021: Added logging to identify why GetLongURL timeout error is occurring.

12/28/2021: Modified stored procedure to handle orphaned records within 5 minutes and not 5 seconds.

12/20/2021: Create a generic admin account for Prod on Court Notify Custom SP.

12/17/2021: Stopped using Exchange Web Service.

12/13/2021: Add generic message for unhandled errors on Custom SP website, Allow SPs to download information from the Custom SP website, fix error on SyncEmailMessageStatus, convert logic apps to webjob to purge records that have passed the retention date, and add a new database to store archived data.

12/07/2021: Update emailmessage table fields ReplyToEmailUsed and FromEmailUsed to be nvarchar(400), Remove ‘Reply-To Email’ textbox from Custom SP, and add missing fields to .CSV to support Reply-To email and From Email.

12/02/2021: Add logic to Handle duplicated subscriptionNames, support Reply-To Email in Service and Organization, add updates to Custom SP Website, Change SP Site Homepage Title, Add new column ReplyToEmailUsed and Rename EmailSender**.**

11/10/2021: Added a new table called ‘Organization’ to reference with ‘ClientApp’ table. SP can now specify their ‘FromEmail’ once they register with Los Angeles Superior court. Added logic to separate permanent failures for emails with returned errors and added logic to Sync all domains when updating ‘MessageQueue’ statuses.

11/01/2021: Adding new logic to update SyncStatus for email to update the statuses correctly.

10/28/2021: Turned off Warmup process for dedicated IP on Mailgun.

10/26/2021: !Bug - Prod: Incorrect MQ Status When Mailgun Responds with Blocked Error

10/25/2021: Fixed bug of Mailgun returning “invalid from email” by using RestSharp when calling Mailgun to send the email.

10/18/2021: Add enhancement for RequestQueue Enqueue to trigger all Dequeue WebJobs and improved BulkMessageQueueHandler to split BulkMessageQueue record to MessageQueue records to maximize parallel process.

09/23/2021: Fix bug where ‘MessageQueue’ record is retried after being left as “Sending” and added logging for ‘BulkMessageQueueHandler’ to create duplicate webjobs.

09/22/2021: Fixed Twilio issue of not setting the status to failed, fixed ErrorMessageData on BulkMessageQueue to store proper error message, Added BulkMessageQueueHandler WebJob and MessageQueueHandler WebJob to replace existing logic apps.

08/31/2021: Fixed transient failure on ShortURL.

08/30/2021: Added fix again for DequeueBulkMessageQueue.

08/27/2021: Fixing error from DequeueBulkMessageQueue.

08/24/2021: Added logic to handle failed api requests with enqueueing and dequeueing, when messages are requested from Service Provider, the date and time will now be added, and fixed a bug from the DequeueBulkMessageQueue operation.

08/18/2021: Added .NET Standard library support.

07/22/2021: Fixed bugs on updating SyncEmailMessageStatus to update with [temporary failures](#_MessageQueue_Status), added a retention in days for Email and Text Messages, and fixed the error on Custom SP | Send Message page.

07/13/2021: Add new APIs to verify Email and Mobile Number, Create a batch job to purge old file attachments in DB, and a batch job to clean up old expired verification codes.

06/24/2021: Removed extra fields from ‘EmailMessage’ table.

06/02/2021: When a bulk request comes in, it is handled right away with our new logic. Added BulkMessageQueueHandlerInstance table.

05/18/2021: Added a generic error message if an email with an invalid domain is passed through and other system related email errors.

05/11/2021: Added logic for batch job to retrieve all BulkMessageQueue records instead of just one. Event and Appointment will be using our BulkMessaging services. FromEmail has been removed from our API. Fixed Mailgun issue of error 421. SP can now use the [GetBulkMessageQueue](#_GetEmailOptInOutInfo) to retrieve information regarding the BulkMessage messages. Attachments being sent out on Client library now have a compression feature enabled as true.

05/05/2021: Fixed CheckMailGunStatus batch job calling SyncEmailMessage, Added RegEx to accept more fields for Message Group ID in Custom SP, and fixed Custom SP Message Log column Respond? To show if recipient responded or not.

04/29/2021: Removed ‘FromEmail’ from API, Added attachment size with Zip, Implemented EWS with OAuth, Added a new method called GetBulkMessageQueue, SP can now view BulkMessageQueueId on Custom SP after sending a Message.

04/21/2021: Fixed Message Log displaying issue and Services to show UseMessageGroup = True.

04/17/2021: Introduce ReceiptNumber, Add Reply-To, Refactor API, Normalize Email and TextMessage body, Support Email Attachments, Fix Email Service Account field, Add datetime offset and fix of sendtextmessage throwing incorrect APIError 500.

03/30/2021: Fixed bug for Mailgun that returns status as pending instead of failed.

03/29/2021: Disabled Mailgun and Twilio callbackURL to update right away.

03/25/2021: Deployed fix for ‘BulkMessageQueue’ to remove QueuedMessageData after processing and use a stored procedured for BulkMessageQueue in Entity Framework.

03/09/2021: Added a fix to the client helper library to return datetime as UTC.

02/08/2021: Fixed query for ‘SyncEmailStatus’ on selecting Mailgun statuses for “sent”.

02/01/2021: Added Nlogging configurable switch on SysSetting and Appointment to use Mailgun.

1/26/2021: Fixed the response time issue on production that was holding up operations by removing the NLOG switch.

01/21/2021: Added new batchjob to update the status from MailGun if it’s not updated and call webhook of SP to update status for email, Add delay for updating status on failed updates, and fixed NULL values on ‘Email’ and ‘Phone’ table.

01/11/2021: Added new field on Send Message only for ‘Admins’.

01/05/2021: Added new setting to delay update from Twilio.

01/04/2021: Fixing of BulkMessageQueue not populating on the DB. Deploying of Webhooks callbackstatus for MailGun and update to Custom SP Message Detail popup page.

12/24/2020: Added new fix to a bug of MessageQueue table column status not being updated.

12/23/2020: Added new email service provider to send out emails and updated Custom SP with numbering formats.

12/23/2020: Refactored Email and Text Messages and updated our batch job of handling bulk messages.

12/14/2020: Added fix for when a recipient tries to reply back to Court Notify System and there is no response back.

12/11/2020: Added changes to Custom SP website, handling log exception for verificationCode, improved batch job, improved SendTextMessage and SendEmailMessage, Updated Expire On to be 60 days, and added new message for inactive message groups when a customer tries to unsubscribe.

12/3/2020: Creating [‘Message Groups’ page](#_Message_Groups) and ‘Send Text Bulk’, Add error page for 404, 500, Fix grammar mistake, Remove item on Configuration on Azure, and Improve Court Notify Header Page

11/17/2020: Improving Error Log for RetrysendMessage and SendTextMessage, Addiing retry logic for RetrySendMessage, & Refactoring MessageId for Text Message.

11/05/2020: Hide reports tab, hide ‘Delete’ link button under Services, fix Service Page on ‘Edit’ to not show query in URL, remove ‘Abbreviation’ from Services, and improve Message Log to support Bulk Message Queue.

11/04/2020: Updated [APIError messages](#_API_Error_1) with Error ‘113’ and added validation for ‘MessageGroupId’.

11/02/2020: ‘[UseMessageGroup’](#_How_to_add) has been implemented and fix of ‘GetMessageQueue’ on API Calls.

10/29/2020: Make Message Log Site ASYNC loading, Drop down is dynamic on Message Log, Fixed minor validation on uploading CSV files, and added more description of Document and Client Library.

10/28/2020: Fixed the “Loading” issue when user times out.

10/27/2020: Records show everything up to To Date on Message Log, fixed total records displayed on the grid in Message Log and added new page information to display how many records are displayed per page

10/26/2020: Added stored procedure for Message Log page and new changes to Send Bulk Message.

10/23/2020: Added new enhancement to Message Log.

10/20/2020: Added fixes to Send Message and Message Log and changes to ‘Unsubscribe’ from the public website.

10/19/2020: Enhanced Message Log with new bug fixes.

10/16/2020: Message Log and Message Detail has been added. Also, fixed bug where if SP didn’t finish signing up on Profile Page and clicks on the Service Provider Custom website, an error message will occur.

10/15/2020: Added enhanced logic for Text Conversation. Refer to Text Message Replies.

10/13/2020: Added new feature to allow Service Providers to upload a CSV file and send it as an email. Refer to the ‘[Send Bulk Online](#_How_to_Send)’ section.

09/23/2020: Added exception message for 'RetrySendMessage' and Fixed null value error for ‘SyncStatus’ messages.

09/22/2020: Fixed SyncStatus to update ‘Sending’ Messages.

09/17/2020: Fixed SyncStatus to update ‘Sent’ Messages, Adding ExpireOn logic for MessageGroup, and BulkMessageQueue Table to keep track of bulk messaging. Added [ExtendMessageGroupExpiredOn](#_ExtendMessageGroupExpiredOn) to extend your ExpireOn DateTime if needed and [UpdateMessageGroupTitle](#_UpdateMessageGroupTitle) to change your title for message groups.

09/08/2020: All MessageGroup or Services will now have a bracket [ ] in the email or text message, Batch Job has been updated to send multiple emails from our new service accounts, Twilio lookup has been implemented to identify more details on why numbers have failed.

08/21/2020: Fixed Logging Error and Updating Error from Log Exception.

08/20/2020: Fixed MessageReplyHandler issue on retrieving incorrect subscription message, fixed CheckTwilioStatus Batchjob, and SQL Server error.

08/14/2020: Fixed error on CheckDisposed function and displaying incorrect title for MessageGroup and Service.

08/11/2020: Updated logic to ‘SyncStatus’ and fixed a batch job error.

08/10/2020: Fixed bug with ‘SyncStatus’.

08/06/2020:

* Added [Limit Calls](#_Limit_Calls) and [HTTP Requests Status Codes](#_HTTP_Requests_Status), updated SendTextMessage, and SendEmailMessage with new information regarding Limit calls.
* Updated Acknowledgement Response to not change the Messagequeue Table Status.
* Added [SendTextMessageInBulk](#_SendTextMessageInBulk) and [SendEmailMessageInBulk](#_SendEmailMessageInBulk).
* Added Final Delivery Status for Service Providers to know that if a message is undelivered or delivered.
* Added logic to handle invalid messages properly.

07/31/2020: Fixed GetMessageQueue bug to be able to search for all statuses regardless of them being “Failed” or “Sent”.

07/28/2020: CourtNotifySp now has an official domain.

07/06/2020: Added new logic for GetNewCode on reusing verification code.

05/27/2020: Added acknowledgment response for SendTextMessage and SendEmailMessage.

05/11/2020: Log Exception framework to log errors. Added new error codes.

04/23/2020: Added column “TryCount” in MessageQueue. Added logic to handle pending messages.

04/22/2020: Added ActivateMessageGroup API, CheckEmailStatus API, CheckMobileNumber API.

04/20/2020: Added Twilio logic to retry messages.

04/17/2020: Improved Batchjob for alerts, Fixed punctuation on APIError messages, and Fixed APIError return from GetMessageQueueLog

04/10/2020: WebAPI config on App Service is configurable for all Environments.

04/09/2020: Implemented feature to log when a customer signs on the SP website.

04/08/2020: Fixed batch job SyncStatus to update statuses with failed correctly.

04/07/2020: Added logic to update Message Queue status with Twilio statuses not yet added and added HideUnsubscribe as a return value for GetMessageQueue and GetMessageQueueLog.

04/06/2020: Added AuditLog feature to keep track of changes made on Services and Client Applications, fixed bug when services are disabled, operations should not work, and reorganized APIs for Client App.

04/01/2020: Added HideUnsubscribe field to SendEmailMessage

03/28/2020: Fixed the bug to update the correct field for DateTime when updating Services.

03/27/2020: Fixed Invalid mobile numbers on SendTextMessage. Added new operation GetNewCodeWithExpireOn.

03/26/2020: Fixed API Error Code for GetMessageQueueLog and DeactivateMessageGroup.

03/25/2020: Updated the subscription logic and changed verbiage for Text Messages.

03/24/2020: Admin page update.

03/23/2020: Added logic to check all status in the batch job. Added logic to make MessageGroupTitle a required field now when sending messages with MessageGroupId.

03/16/2020: Added browser support.

3/12/2020: Service Provider Website – Updated API Client Library for .NET Core 2.0 and 3.0. Added new features when a Service Provider registers for a client application. Secrets have been changed. An access token has been fixed.

3/4/2020: Service Provider Website – basic Service management features are added. The basic process is now fully available.

3/3/2020: API Client Library for .NET (Core and Framework) version is available.

2/4/2020: API – the initial version of API (version 1) is launched. API is ready to send text and email messages.