DirectTEXT Integration & API Guide

Version 2.12

Table of Contents

For more information, contact your Account Director or Technical Program Manager.

https://www.kaleyra.com

Table 1. Document Version History

Version	Updated	Changes
2.1	6/30/16	1. Changed standard SMS file structure for FTP drop-off in "Drop- Off Delivery" section 2. Introduced MMS functionality and file structure for FTP file drop-off in "Drop-Off Delivery" section
2.2	12/28/16	Introduced section for multi-byte encoding for sending/receiving emoji in SMS body text for MT and MO
2.3	06/25/18	Design update
2.4	07/27/18	Refreshed version of the original API guide that includes detailed information on using emojis in messages as well as the Carrier Lookup API

Version	Updated	Changes
2.5	05/02/19	1. DirectTEXT API URL changed to: https://directtext.mgage.com. This is reflected in all the examples. 2. Corrected the list of parameters included in SMS and MMS MOs. 3. Added new status/error codes specific to non-US and non-Canadian destinations. 4. Minor cleanup.
2.6	05/17/19	Updated the *MMS MOs: Message Notification Contents* section.
2.7	05/06/20	Document references US and UK endpoints available to customers
2.8	08/27/20	1. Added new Message Status API. 2. Updated MO response examples for SMS and MMS 3. Added DR example 4. Updated Status values table
2.9	04/06/21	Added new API to send multiple messages in a single request .
2.10	06/15/21	Details on new smpp_errcd field returned in Delivery Receipts .
2.11	12/10/21	1. Kaleyra re-branding. Updated images to reflect Kaleyra branding and removed references to mGage throughout. API endpoints have not been updated at this time. 2. Text for status codes has been updated
2.12	02/06/22	1. Added ability to use indexes on additional parameters in API for multiple message flexibility in a single request: send_date, local, reply_to, reporting_key1, reporting_key2, client_recip_id

PREFACE

Overview

This guide details how Kaleyra's DirectTEXT API can be used to add SMS or MMS messaging capabilities to existing systems and applications. Both the DirectTEXT API and Drop-Off Delivery messaging methods are discussed. Basic familiarity with Internet/web concepts and at least one programming language (for the DirectTEXT API) is assumed.

DirectTEXT

DirectTEXT provides a straightforward method to add SMS or MMS message capabilities to your existing applications and systems. Based on the Kaleyra Messaging Platform, DirectTEXT allows you to send or receive messages to/from nearly any mobile device in the world, regardless of the country or carrier network. Depending on your needs, either the Messaging Application Program Interface (API) or Drop-Off Delivery methods can be used.

Messaging API

The DirectTEXT Messaging API allows developers to easily integrate SMS or MMS messaging into their applications. Using the API, it is possible to send SMS/MMS messages, receive SMS/MMS messages, and obtain status information for previously-sent SMS messages. Integration with the API can be done with any Internet-capable programming language, and several examples of using the API using different languages are provided in: APPENDIX A: DirectTEXT API CODE EXAMPLES

Carrier Lookup API

Using the DirectTEXT API, customers can lookup and retrieve carrier information associated with their subscribers' mobile numbers.

Drop-Off Delivery

Using Drop-Off Delivery, a text file containing messages to deliver (as well as configuration information) is delivered to Kaleyra via secure FTP. Drop-Off Delivery is useful for bulk sending of a large number of messages, and the file format used is easily manipulated with shell scripts and scripting languages like Perl.

SENDING MOBILE-TERMINATED MESSAGES (MTs)

Delivery Process

The Messaging API allows you to send mobile-terminated (MT) messages to your users' mobile phones. Messages are delivered via authenticated HTTP requests to Kaleyra servers, with the message details specified in either GET or POST parameters. Kaleyra then processes the message and forwards it to the appropriate carrier for delivery to the recipient's mobile phone.

All API examples described in this document reference the https://directtext.mgage.com endpoint, however, please use the US or UK API endpoint listed below that most closely corresponds with your connectivity requirements or depending on the region where your short codes, long codes or sender IDs are being provisioned. Please consult with your Technical Program Manager to confirm the correct API endpoint to use and to obtain your API credentials (username and password).

US - Atlanta GW

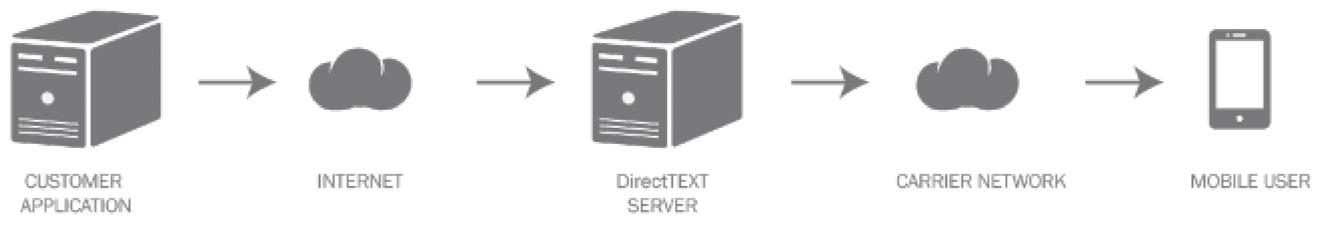
```
https://directtext.mgage.com
```

US - Minneapolis GW

```
https://directtext-msp.mgage.com
```

UK GW

```
https://directtext-uk.mgage.com
```



Call Format

To send a message via the Messaging API, simply make a HTTP request in this format:

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=<shortcode>&recipient=<
mobilenumbers>&body=<messagetext>&reporting_key1=<key1>&reporting_key2=<key2>&client_reci
p_id=<ClientRecipID>&send_date<date>&carrier=<carrier name>&local=<true or false>
```

MT Parameters

An explanation of the parameters appears in the table below:

Parameter	Description
body	The text of the message to be sent. NOTE: This message must be URL encoded.
carrier	If the carrier is already known for a mobile number, this parameter may be used to include the name of the carrier - for e.g. 'verizon', 'att', 'sprint', 'tmobile' etc.
channel	Use the values 'sms' for SMS message, or 'multimedia' for MMS messages. If using 'multimedia', be sure to include the 'content_url' parameter in the request. If no channel type is specified, the system will default to an SMS request.
client_recip_id	Parameter that allows for tracking of an HTTP API request and can be used for reporting purposes. Applies to all recipients/mobile numbers that are part of the request.
content_url	The value of this field should be the URL that points to the contents of the MMS message (such as an image or a video or audio clip).
local	Enables Time Zone delivery for the mobile numbers specified in the request. This parameter should be used in conjunction with the 'send_date' parameter, and only works for North America phone numbers. It allows for the scheduled delivery of the message to take place at the specified time in each Time Zone based on the mobile number's Area Code. For example, if the message has a send time of 11:00 am, mobile users on the Eastern, Central, Mountain and Pacific time zones would each receive the message at 11:00 am their time. To implement this feature, set the local value to 'true'.
mo_reference_id	The mo_reference_id should be included when sending an MT in response to an MO (e.g. when the end-user texts a keyword to the short code and is expecting a response back instantly). This will increase the priority of the message so as to provide a more seamless, 2-way conversational experience. Requests containing a mo_reference_id will have a higher priority compared to requests that don't.
recipient	The mobile phone number of the message recipient. Multiple recipients may be specified (up to 1000 recipients may be included). NOTE: Must meet E.164 formatting guidelines. For US domestic traffic, the full 11-digit number (including the 1 for country code) must be specified.
reply_to	The short code, long code or sender ID over which the message should be sent. It must be registered/provisioned in Kaleyra's platform and associated with your account before you can send/receive messages on it. Reach out to your Technical Program Manager for any questions related to the setup of your account.

Parameter	Description
reporting_key1 and reporting_key2	Optional parameters that can be used to associate user-defined data with the message for reporting purposes.
send_date	This parameter allows the messages to be scheduled for a specific delivery date and time. The date format should be “2018-01-01 16:00:00”, or the ISO-8601 format.
ttl	TTL stands for Time-To-Live. The time is relative to the time that the request is received. The value of this field should be the number of minutes that the request should be kept “alive” in Kaleyra. By the time the specified number of minutes have elapsed, if the messages in the request have not been successfully handed off to the carrier, Kaleyra’s gateway will automatically expire all messages that were part of the request.

Response Format

Responses to Messaging API calls are returned in XML using the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<httpApiResponse>
    <code>code</code>
    <description>description</description>
    <recipients>
        <recipient>
            <mobileNumber>mobile number</mobileNumber>
            <messageId>message id</messageId>
        </recipient>
    </recipients>
</httpApiResponse>
```

A response code of 100 indicates that the message was successfully accepted for processing. A list of the mobile numbers and their corresponding message IDs are also shown for logging purposes. A response code other than 100 indicates an error condition, with details specified in the ‘description’ element of the XML. Please refer to the table below for a list of possible error codes.

Code	Status	Definition
901	Profane content	
992	Invalid Body	“Missing or empty parameter. Body is required.” The ‘body’ field of the request is either empty or invalid.
994	Invalid reply_to	“Missing or empty parameter. reply_to is required.” The reply_to field of the request is either empty or invalid.

Code	Status	Definition
998	Invalid request	A component of the request is invalid. Details will appear in the description element of the response returned by the API call.
999	Internal error	An internal error occurred within Kaleyra's DirectTEXT server.

Example: Sending a plain-text SMS to a Single Recipient

The following API request will send the message "Hello from DirectTEXT" to the mobile phone number 14045552900 over the short code 12345.

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&body=Hello%20From
%20DirectTEXT&recipient=14045552900
```

Example: Sending a plain-text SMS to Multiple Recipients

The following URL will send the message "Hello from DirectTEXT" over short code 12345 to three different mobile phone numbers: 14045552900, 14045553900, and 14045554900.

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&body=Hello%20From
%20DirectTEXT&recipient=14045552900&recipient=14045553900&recipient=14045554900
```

Example: Sending a SMS containing Multi-Byte Emojis

The following URL will send the message "Thanks for your business. 👍 😃" over short code 12345 to three different mobile phone numbers: 14045552900, 14045553900, and 14045554900.

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&body=Thanks%20for
%20your%20business.%20%F0%9F%91%8D%20%F0%9F%98%83&recipient=14045552900&recipient=1404555
3900&recipient=14045554900
```

For more information on using Emojis in your SMS, refer to: APPENDIX B: USING EMOJIS IN MESSAGING

Example: Sending a Scheduled SMS with Time Zone delivery enabled and with Reporting parameters

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter/?reply_to=12345&recipient=140455
54900,13125555900,19705554900&body=This%20is%20a%20test%20message%20from%20mGage&local=tr
ue&send_date=2018-07-24%2016:00:00&reporting_key1=mGageTest-
20170331&client_recip_ID=mGage
```

This request will result in each mobile number receiving the message at 4pm in their local time zone on 07/24/2018. Time Zone is determined based on the mobile number's Area Code. Hence, the 404 number (a Georgia area code) will receive the message at 4pm ET, the 312 number (an Illinois Area Code) will receive the message at 4pm CT, and the 970 number (a Colorado area code) at 4pm MT.

Example: Sending a SMS with a higher priority and with a specific TTL

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter/?reply_to=12345&recipient=140455
54900&body=This%20is%20a%20test%20message%20from%20mGage&ttl=10&mo_reference_id=MO-TEST
```

This request has a TTL (Time-To-Live) of 10 minutes, which means it will stay alive in Kaleyra's platform for 10 minutes. If the message has not been handed off to the carrier within 10 minutes, the message will automatically expire. Due to the presence of an **mo_reference_id** in the request, the message will be prioritized and treated as an MT response to a MO (i.e. 2-way SMS).

Example: Sending a MMS to Multiple Recipients

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter/?reply_to=12345&recipient=140455
54900,13125555900,19705555900&body=This%20is%20a%20MMS%20Test&channel=multimedia&content_
url=https://1i95qs29u5hu1feo5cwhejq8-wpengine.netdna-ssl.com/wp-
content/uploads/2017/06/logo@2x.png
```

Sending Multiple SMS Message Content in a Single Request

In addition to sending a single piece of message content to multiple MSISDNs, the Direct Text API provides clients the flexibility to customize and send multiple pieces of message content to up to 1000 MSISDN's in a single request. This is accomplished by using the same content parameters, but including a number (an index) at the end of the parameter. One or all of them can be used in the message, but it is recommended that a default value (with no index) be included in the call, particularly for reply_to so that calls will have a default to use and will not fail.

The endpoint to access this API varies from the one referenced in SENDING MOBILE-TERMINATED MESSAGES (MTs).

Call Format

To send a message, make a HTTP request in the format below. Further examples are provided below. The concept is that if an index is included on a parameter, then it will ….

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?reply_to=<shortcode>&recipie
nt1=<mobilenumbers>&body1=<messagetext1>&recipient2=<mobilenumbers>&body2=<messagetext2>&
reporting_key1=<key1>&reporting_key2=<key2>&client_recip_id=<ClientRecipID>&send_date<dat
e>&carrier=<carriername>&local=<true or false>
```

MT Parameters

An explanation of the parameters are described in the table below:

Parameter	Description
body1	message body content for 1st message group to send. Required to provide a recipient list defined in corresponding parameter recipient1. NOTE: This message must be URL encoded.
body2	message body content for 2nd message group to send.
bodyN	message body content for Nth message group to send. Upto 1000 messages can be included
recipient1	List of mobile phone numbers for 1st message body. NOTE: Must meet E.164 formatting guidelines. For US domestic traffic, the full 11-digit number (including the 1 for country code) must be specified.
recipient2	List of mobile phone numbers for the 2nd message body.

Parameter	Description
recipientN	List of mobile phone numbers for the Nth message body. Up to 1000 recipients may be included in the request.
carrier	If the carrier is already known for a mobile number, this parameter may be used to include the name of the carrier - for e.g. 'verizon', 'att', 'sprint', 'tmobile' etc.
channel	Use the value 'sms' only
client_recip_id	Parameter that allows for tracking of an HTTP API request and can be used for reporting purposes. Applies to all recipients/mobile numbers that are part of the request. This will also be the default client_recip_id used in requests that do not contain a specific client_recip_id indexed param.
client_recip_id_1 or client_recip_id1	Parameter that allows for tracking of an HTTP API request and can be used for reporting purposes. Applies to all recipients/mobile numbers that are part of the request that has an index of 1.
client_recip_id_2 or client_recip_id2	Parameter that allows for tracking of an HTTP API request and can be used for reporting purposes. Applies to all recipients/mobile numbers that are part of the request that has an index of 2.
client_recip_id_N or client_recip_idN	Parameter that allows for tracking of an HTTP API request and can be used for reporting purposes. Applies to all recipients/mobile numbers that are part of the particular indexed (Nth) request.
local	Enables Time Zone delivery for the mobile numbers specified in the request. This parameter should be used in conjunction with the 'send_date' parameter, and only works for North America phone numbers. It allows for the scheduled delivery of the message to take place at the specified time in each Time Zone based on the mobile number's Area Code. For example, if the message has a send time of 11:00 am, mobile users on the Eastern, Central, Mountain and Pacific time zones would each receive the message at 11:00 am their time. To implement this feature, set the local value to 'true'.
local_1 or local1	Ensures only US numbers will be scheduled for delivery of the message to take place at the specified time in each Time Zone based on the mobile number's Area Code for those with the index of 1.
local_2 or local2	Ensures only US numbers will be scheduled for delivery of the message to take place at the specified time in each Time Zone based on the mobile number's Area Code for those with the index of 2.
local_N or localN	Ensures only US numbers will be scheduled for delivery of the message to take place at the specified time in each Time Zone based on the mobile number's Area Code for those with the index of N.
mo_reference_id	The mo_reference_id should be included when sending an MT in response to an MO (e.g. when the end-user texts a keyword to the short code and is expecting a response back instantly). This will increase the priority of the message so as to provide a more seamless, 2-way conversational experience. Requests containing a mo_reference_id will have a higher priority compared to requests that don't.

Parameter	Description
reply_to	The short code, long code or sender ID over which the message should be sent. It must be registered/provisioned in Kaleyra's platform and associated with your account before you can send/receive messages on it. This will be considered the DEFAULT reply_to for any indexed groups that do not have a specific indexed reply_to. Reach out to your Technical Program Manager for any questions related to the setup of your account.
reply_to_1 or reply_to1	The short code, long code or sender ID over which the message should be sent for the message group with index of 1. It must be registered/provisioned in Kaleyra's platform and associated with your account before you can send/receive messages on it. Reach out to your Technical Program Manager for any questions related to the setup of your account.
reply_to_2 or reply_to2	The short code, long code or sender ID over which the message should be sent for the message group with index of 2. It must be registered/provisioned in Kaleyra's platform and associated with your account before you can send/receive messages on it. Reach out to your Technical Program Manager for any questions related to the setup of your account.
reply_to_N or reply_toN	The short code, long code or sender ID over which the message should be sent for the particular indexed message group. It must be registered/provisioned in Kaleyra's platform and associated with your account before you can send/receive messages on it. Reach out to your Technical Program Manager for any questions related to the setup of your account.
reporting_key1 and reporting_key2	Optional parameters that can be used to associate user-defined data with the message for reporting purposes. These are not indexed.
reporting_key1_1 (or reporting_key11) and reporting_key2_1 (or reporting_key21)	Optional parameters that can be used to associate user-defined data with the message for reporting purposes for the group with index of 1.
reporting_key1_2 (or reporting_key12) and reporting_key2_2 (or reporting_key22)	Optional parameters that can be used to associate user-defined data with the message for reporting purposes for group with index of 2.
reporting_key1_N (or reporting_key1N) and reporting_key2_2 (or reporting_key2N)	Optional parameters that can be used to associate user-defined data with the message for reporting purposes for group.

Parameter	Description
send_date	This parameter allows the messages to be scheduled for a specific delivery date and time. The date format should be “2018-01-01 16:00:00”, or the ISO-8601 format.
ttl	TTL stands for Time-To-Live. The time is relative to the time that the request is received. The value of this field should be the number of minutes that the request should be kept “alive” in Kaleyra. By the time the specified number of minutes have elapsed, if the messages in the request have not been successfully handed off to the carrier, Kaleyra’s gateway will automatically expire all messages that were part of the request.

Response Format

Response to the API calls are returned in XML using the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <httpApiResponse id="body1">
        <code>code</code>
        <description>Description</description>
        <recipients>
            <recipient>
                <mobileNumber>mobile number</mobileNumber>
                <messageId>message id</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
    <httpApiResponse id="body2">
        <code>code</code>
        <description>Description</description>
        <recipients>
            <recipient>
                <mobileNumber>mobile number</mobileNumber>
                <messageId>message id</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
</bulkHttpApiResponse>
```

A response code of 100 indicates that the message was successfully accepted for processing. A list of the mobile numbers and their corresponding message IDs are also shown for logging purposes. A response code other than 100 indicates an error condition, with details specified in the ‘description’ element of the XML. Please refer to the table below for a list of possible error codes.

Code	Status	Definition
901	Profane content	
992	Invalid Body	“Missing or empty parameter. Body is required.” The ‘body’ field of the request is either empty or invalid.
994	Invalid reply_to	“Missing or empty parameter. reply_to is required.” The reply_to field of the request is either empty or invalid.
998	Invalid request	A component of the request is invalid. Details will appear in the description element of the response returned by the API call.
999	Internal error	An internal error occurred within Kaleyra’s DirectTEXT server.

Example: Sending multiple SMS message content to multiple recipients

Request

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?client=2notify&channel=sms&c
ustomer_id=1&reporting_key1=key1&reporting_key2=key2&reply_to=26161&body1=message1&recipi
ent1=MDN1,MDN2&body2=message&recipient2=MDN3
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <httpApiResponse id="body1">
        <code>100</code>
        <description>Success. Message accepted for delivery to the following 
recipients.</description>
        <recipients>
            <recipient>
                <mobileNumber>MDN1</mobileNumber>
                <messageId>262463325360</messageId>
            </recipient>
            <recipient>
                <mobileNumber>MDN2</mobileNumber>
                <messageId>262463325361</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
    <httpApiResponse id="body2">
        <code>100</code>
        <description>Success. Message accepted for delivery to the following 
recipients.</description>
        <recipients>
            <recipient>
                <mobileNumber>MDN3</mobileNumber>
                <messageId>262463325360</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
</bulkHttpApiResponse>
```

Example: Sending particular reporting_key1 and reporting_key2 parameters to ensure proper reporting is easily done by using reporting_key values.

Request

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?reply_to=[shortcode]&recipie
nt1=MDN1,MDN2&body1=message_body1&body2=message_body2&recipient2=MDN3,MDN4&reporting_key1
=rpk1&reporting_key2=rpk2&customer_id=43&client_recip_id1=client_receipt1&client_recip_id
2=client_receipt2&send_date=YYYY-MM-DD HH24:MI:SS&local=false
```

Example: Customers oriented to US only can choose to send with local=true to ensure numbers receive the messages for their specific time zone.

Request

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?reply_to=[shortcode]&recipie
nt1=MDN1,MDN2&body1=message_body1&body2=message_body2&recipient2=MDN3,MDN4&reporting_key1
=rpk1&reporting_key2=rpk2&customer_id=43&client_recip_id=client_recip_id1&send_date=YYYY-
MM-DD HH24:MI:SS&local=true
```

Example: Specific message body depending on the recipient list can be achieved with body_x or bodyx (where x is an index number matching a recipient’s index number).

Request

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?reply_to=[shortcode]&recipie
nt1=MDN1,MDN2&body1=message_body_for_shop_1&body2=message_body_for_shop_2&recipient2=MDN3
,MDN4&reporting_key1=rpk1&reporting_key2=rpk2&customer_id=43&client_recip_id=client_recip
_id1&send_date=YYYY-MM-DD HH24:MI:SS&local=true
```

Example: Multiple send dates can be managed on a single call as soon as the proper send_date_x or send_datex index and recipient_listx indexes are used.

Request

```
https://directtext.mgage.com/a2w_preRouter/bulkHttpApiRouter?reply_to=[shortcode]&recipie
nt1=MDN1,MDN2&body1=message_body_for_shop_1&body2=message_body_for_shop_2&recipient2=MDN3
,MDN4&reporting_key1=rpk1&reporting_key2=rpk2&customer_id=43&client_recip_id=client_recip
_id1&send_date1=YYYY-MM-DD HH24:MI:SS&send_date2=YYYY-MM-DD HH24:MI:SS&local=true
```

Example: Response when a message body in the request has profanity

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <httpApiResponse id="body1">
        <code>901</code>
        <description>Profanity filter is enabled for this user and the provided content
was rejected. If you believe this message does not contain profanity and was rejected in
error, please speak with your Air2web Project Manager.</description>
        <recipients>
            <recipient>
                <mobileNumber>1424202883</mobileNumber>
                <messageId>184195302538</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
    <httpApiResponse id="body2">
        <code>100</code>
        <description>Success. Message accepted for delivery to the following
recipients.</description>
        <recipients>
            <recipient>
                <mobileNumber>14165532636</mobileNumber>
                <messageId>184195302539</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
</bulkHttpApiResponse>
```

Example: Response when request is missing a message body

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <httpApiResponse id="body1">
        <code>992</code>
        <description>Missing or empty parameter. body is required</description>
    </httpApiResponse>
    <httpApiResponse id="body2">
        <code>992</code>
        <description>Missing or empty parameter. body is required</description>
    </httpApiResponse>
</bulkHttpApiResponse>
```

Example: Response when request is missing a field such as reply_to

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <code>994</code>
    <description>Missing or empty parameter. reply_to is required</description>
</bulkHttpApiResponse>
```

Example: Response when request includes an invalid number

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <httpApiResponse id="body1">
        <code>998</code>
        <description>Invalid request. Please specify a recipient parameter with a validly
formatted E164 phone number.</description>
    </httpApiResponse>
    <httpApiResponse id="body2">
        <code>100</code>
        <description>Success. Message accepted for delivery to the following
recipients.</description>
        <recipients>
            <recipient>
                <mobileNumber>14165532636</mobileNumber>
                <messageId>184195302567</messageId>
            </recipient>
            <recipient>
                <mobileNumber>1424202885</mobileNumber>
                <messageId>184195302568</messageId>
            </recipient>
        </recipients>
    </httpApiResponse>
</bulkHttpApiResponse>
```

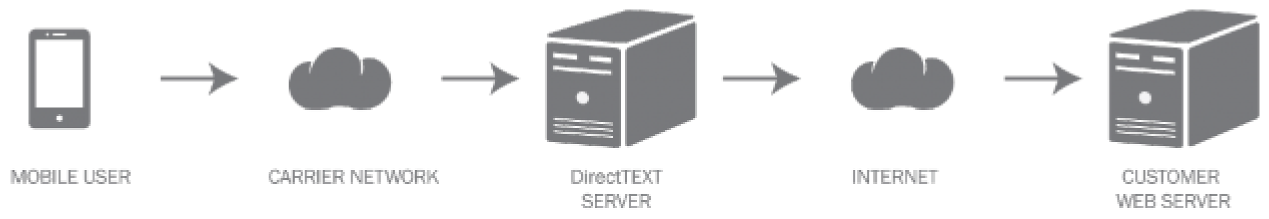
Example: Response when request exceeds more than 1000 recipients

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkHttpApiResponse>
    <code>998</code>
    <description>998.Too many recipients, found 1015 recipients, maximum limit is 1000
</description>
</bulkHttpApiResponse>
```

RECEIVING MOBILE-ORIGINATED (MO) MESSAGES

The DirectTEXT Messaging API allows you to receive mobile-originated (MO) messages from users' mobile phones. Before you can receive messages, you must work with your Technical Program Manager to complete the provisioning process and provide an endpoint URL/webhook that you wish use to receive the notifications on.

When a user sends an SMS message from his mobile phone, it is received by the carrier network and routed to an Kaleyra messaging server. The Kaleyra DirectTEXT server converts this message into a form-encoded or URL-encoded HTTP POST and forwards it to an endpoint URL on a web server (i.e. a webhook URL) that you have implemented and hosted (similar to how an HTML form is submitted).



Generally, the URL you provide is for a web-based script or program that will process the received message and send a reply using the message sending capabilities described in the previous section. Examples of receiving messages in various programming languages are included in Appendix A of this guide.

SMS MOs: Message Notification Contents

Request Example

```
POST <CLIENT_URL_PATH> HTTP
Host: <CLIENT_URL_HOST:PORT>
Content-Type: application/x-www-form-urlencoded
inbound_address=23157
&router=outspoken
&carrier=verizon
&message_orig=Yes
&message=Yes
&message_id=1000000
&message_subid=0
&status=Processing
&status_code=10
&a2w_mo_ref_id=1000000
&country_code=1
&device_address=11234567890
```

The following POST parameters are included in SMS MOs:

Key	Description
message_id	Kaleyra-assigned unique message ID that is given to a message when the MO is received from the carrier.
message_subid	For multi-part MOs (i.e. MOs containing long message text), this parameter corresponds to the part number of the MO message. By default single-part MOs will have a message_subid of '0'. Multi-part MOs will have subid = '1', '2', etc. depending on how many parts the MO has been split into. Each split part will still have the same message_id.
a2w_mo_ref_id	Currently this parameter will have the same value as message_id
device_address	Mobile number of the wireless device that the message originated from.
country_code	The country code associated with the mobile number. E.g. '1' for US mobile numbers, '44' for UK, etc.
inbound_address	Short code or sender ID that the MO message was sent to.
message_orig	Message body or content of the text as originally sent by the subscriber from their mobile device.
message	Filtered version of the original message body or content of the text. The system filters out certain matching regular expressions from MOs, such as: "Re:", "In Reply To.", "-----Original Message-----", etc. as these prefixes may be automatically inserted by the end-user's device to MOs.
carrier	The carrier associated with the mobile number.
router	The Kaleyra server/message router that processed the MO.
channel	Always set to default value of "sms"
status	Corresponds to the status description of the MO as it is processed by the Kaleyra servers.
status_code	The status code/value of the MO as it is processed by the Kaleyra servers.

MMS MOs: Message Notification Contents

MMS MOs are returned in a URL-encoded format The following POST parameters are included.

Key	Description
fetchAttachment Url	Applicable only for MMS. This is an Kaleyra-generated URL and resolves to the location from where the MO multimedia content can be fetched. The content is stored within a single attachment and should be fetched by executing an HTTP GET request to the received **fetchAttachementUrl**, which is in the format: https://mms.mgage.com/vmmg/fetchAttachment/<message_id>; where **message_id** contains a numeric Kaleyra-generated identifier that is assigned to uniquely identify the MO message that is received and processed from the carrier. The **fetchAttachmentUrl** will return a HTTP response with all pieces of the MMS message, text and/or media as **Content-Type: multipart/mixed** with **Content-Transfer-Encoding: base64**. MIME multipart standard formatting is used. Note that **fetchAttachementUrl** will be empty for MMS MOs containing a **subject** only. **Supported Media Types (base64 encoded)**: *Text/Plain*, *Image (GIF/JPEG/PNG)*, *Video (3GPP/MP4)*, *Audio (AMR/MP3)*.
device_address	10-digit US mobile number that the message originated from.
inbound_address	Short code or sender ID that the MO message was sent to.
message_id	Kaleyra-assigned unique message ID that is given to a message when the MO is received from the carrier. The same ID is include at the end of **fetchAttachmentUrl** above.
subject	Includes the message subject as originally sent by the subscriber from their mobile device (if any); UTF-8, URL-encoded. MMS MOs are not required to have a message subject, so this parameter will be empty if the mobile user has not sent the message subject from their device.
message_orig	Message body or text (UTF-8, URL encoded) as originally sent by the subscriber from their mobile device (if any). MMS MOs are not required to have a body text, so this parameter will be empty if the user has not sent in any text.

Below is a sample request that shows an HTTP POST of the URL-encoded MMS MO to a customer's endpoint. The MO contains the subject, the message text and a JPEG image that can be fetched from the **fetchAttachmentUrl**.

```
POST <CLIENT_URL_PATH> HTTP
Host: <CLIENT_URL_HOST:PORT>
Content-Type: application/x-www-form-urlencoded
fetchAttachmentUrl=https%3A%2F%2Fmms.mgage.com%2Fvmmg%2FfetchAttachment%2F196129834607
&device_address=11234567890
&inbound_address=45678
&message_id=196129834607
&subject=MMS%20MO%2 0Test
&message_orig=MMS%20MO%20with%20a%20JPEG%20image
```

In the above example, the customer’s web service would fetch the content from the URL: https://mms.mgage.com/vmmg/fetchAttachment/196129834607. The resulting HTTP Response file would include both the Text and Image media within a single attachment.

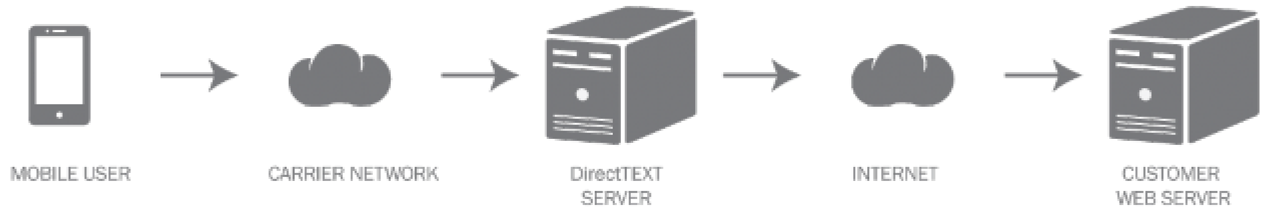
In general, the HTTP Response files can include different types of media depending on the type of MO that the mobile user has sent, such as:

- Single Text media MMS MO
- Single Image media MMS MO
- Single Video media MMS MO
- Single Audio media MMS MO
- A combination of Text, Image, Video, Audio media MMS MO

STATUS NOTIFICATIONS / DELIVERY RECEIPTS

Status Notifications or Delivery Receipts (DRs) are status updates on the progress or state of a message within the carrier's network. This is an optional configuration depending on whether you wish to receive these notifications. To receive them, contact your Technical Program Manager and request to register a DR route for your short code, long code or sender ID. You will need to provide an endpoint or webhook URL on which you want to receive the DRs (similar to the MO endpoint URLs). Using HTTP form-encoded POSTs, DirectTEXT relays the status and delivery receipt data points through HTTP embedded name-value pairs to your webhook URL that you have implemented and hosted within your system.

Note: DRs are currently supported for SMS messages only.



DRs can be configured for a short code and carrier combination and you may choose one of the following configuration options:

- **Final Delivery Receipts ONLY**: Kaleyra will configure routes in such a way that you will only receive the final delivery receipt for your MT messages, as long as the underlying carrier has sent the delivery receipt to Kaleyra. The final delivery receipt or status of a message may denote either success or failure.
- **Intermediate Status Receipts + Final Delivery Receipts**: This option will allow you to receive ALL status updates - intermediate as well as final - as the message transitions or progresses from one state to the next on it's delivery path.

Not all carriers provide status notifications. If Kaleyra receives a final status from the carrier we provide it to you. "Carrier Success" and a status code of 200, which is the most common status and status code, indicates that the carrier has confirmed successful delivery of message to the subscriber's device. For information on other status codes, refer to the Status Values table. For carriers that don't provide DR information, the final status is typically "Carrier Received", with a status code of 40. This denotes that Kaleyra delivered the message to the carrier and the carrier acknowledged the receipt of the message.

Status Notification/ Delivery Receipt Contents

The following POST parameters are sent in the status/delivery receipt:

Code	Status
carrier	Carrier from which the subscriber MO originated.
channel	Channel used for message delivery: e.g. 'sms'.

Code	Status
client_message_id	Not used with DirectTEXT Messaging API
date_format	Indicates the format of the timestamp of the update_date parameter. For example, a date_format of MM/dd/yyyy HH:mm:ss,SSS would indicate that the timestamp is in the format 01/22/2010 14:54:18,541.
device_address	Mobile number of the wireless device receiving the SMS message.
inbound_address	A source address (short code/long code/sender ID) provisioned in the Kaleyra system as well as registered/provisioned on a carrier's network to which a subscriber sends a MO.
message_id	Kaleyra-assigned message ID that is given to a message when you send an MT using the Messaging API. It maps the message status to the original request when the MT was sent.
message_subid	If the message is split because it's longer than 160 characters, it will result in more than one status notification. The status notification for the first part of the message (or if the message is 160 characters or less, it will be for the entire message) will have a message_subid of 0. The second part of the message will have a subid of 1, the third of 2, etc.
reporting_key1 and reporting_key2	Populated if these values were included when sending the message using the Messaging API.
router	The name of the Kaleyra server that processed the message.
status	The current status of the message as it relates to the carrier network. See "Codes and Definitions" below for status codes.
status_code	Numeric value that denotes success, failure or intermediate status of the message.
status_info	Represents detailed information corresponding to the numberic status code. *Note*: This information can change, and we advise you not to parse it.
smscid	Carrier assigned message ID that is given to a message when you send an MT using the Messaging API.
update_date	A date/time stamp reflecting the system time on the Kaleyra hosted router when the carrier status/delivery receipt was received.
uaprof	User agent profile, contains information for content formatting purposes. This will be included in the DR. Default value is null.

Code	Status
smpp_errcd	Includes the SMPP error code that is returned by Kaleyra and carrier systems. Provides better insights into the failure of a message. The SMPP error code detail, on top of what the existing “status_code” value provides, can help better debug customer issues. SMPP Error code value is only provided for relevent status codes. For other statuses the field will be empty or not be passed. Refer to table below for more details

Status Values

The status parameters in delivery and status receipts are based on a carrier’s system internal values that are subject to change. The following table lists possible status values and their definitions based on current available data.

Code	Status	Definition	SMPP Error Code
5	Initial Processing	The message was received by the prerouter.	
6	Initial Processing	The message was received by MT Pipeline.	
7	Router Communication Error	An internal Kaleyra error. Upon receiving this error, the content provider should wait 5 minutes for a subsequent status notification. If no status notification is received, the content provider should start the retry algorithm.	
9	Successfully Sent to Router	The prerouter successfully sent the message to the router.	
10	Processing	The router successfully received the message from the prerouter.	
16	No Route found	Unsupported carrier	
20	Message sent to carrier	Carrier is assessing the MT request.	000
30	Retry	Message queued for retry (only applicable if status receipts are received also).	
40	Carrier Received	Carrier has accepted the message for delivery. In some cases, this is the last available status from a carrier.	000

Code	Status	Definition	SMPP Error Code
100	MO Success	The MO was successfully posted to the webhook/endpoint URL configured for the short code/sender ID.	000
105	MT Short code is blocked	The carrier is blocking traffic on the specified short code.	045
110	Fail	Indicates that Carrier returned a failure to deliver message. In some cases, indicates Internal Kaleyra errors such as route not allowed, message expired etc.	Hexadecimal value from Carrier is returned. Refer to SMPP Error Codes for most common error codes. In some cases if failure occurs with Kaleyra 045 is returned.
120	MO Route Failure	No route found for shortcode, customer, carrier or error sending MO	null
121	Enqueue Fail	Internal error – JMS enqueing error.	null
124	Retry Router Request Fail	The message expired before communication with the router was restored.	
129	Carrier could not be resolved	Kaleyra was unable to resolve the MDN to a valid wireless carrier. The number is either a landline or invalid or incorrectly formatted.	
130	Split	The message was split.	
131	Profanity Blocked	CleanAIR – Inappropriate word(s) detected and blocked	
132	Msg Paused for URL Approval	CleanAIR – URL detected and blocked	
133	Paused: Profanity and URL found	CleanAIR – Inappropriate word(s) and URL detected and blocked	
200	Carrier Success	Carrier indicates successful delivery of message to the subscriber's device.	000
210	Carrier Manual Ack	Response from carrier: manual ack.	000
250	Carrier Expired	The default period for the SMSC (typically 48-72 hours) has passed without successful message delivery. This typically happens if the subscriber remains out of coverage or keeps their phone turned off for an extended period of time.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes

Code	Status	Definition	SMPP Error Code
260	Carrier Deleted	A manual status indicating that the message has been deleted by the carrier's SMSC itself, without delivery.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
270	Carrier Undeliverable	The message couldn't be delivered. This happens because sometimes carriers' SMSCs will accept messages for recipients they don't own due to data roaming agreements, but they can't deliver the message because of any variety of reasons (one would be the subscriber's prepaid credits have run out or the subscriber is not valid). This typically happens for SMSCs outside of the United States where data roaming over SMSCs is typical.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
275	Carrier Undeliv Permanent	The message couldn't be delivered. This error is specific to non-US destinations, where undeliverable status was received from the carrier or delivery partner. The nature of the error state is a permanent failure, which means this mobile number is not a valid delivery destination and no further delivery attempts should be made.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
277	Carrier Undeliv Temporary	The message couldn't be delivered. This error is specific to non-US destinations, where undeliverable status was received from the carrier or delivery partner. The nature of the error state is a temporary failure, which means future delivery attempts can be made.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
280	Carrier Unknown	The stopgap error code generated by the SMSC when no other possible condition exists.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
290	Carrier Rejected	This typically happens for carriers outside of the US when a data roaming carrier rejects the particular message.	Error Code from carrier is passed on. Refer to SMPP Error Codes for list of error codes
99999	Unknown	Kaleyra has received a delivery receipt/status notification outside the carrier-defined or Kaleyra-known status receipt codes.	

Delivery Receipt Example

```
POST <CLIENT_URL_PATH> HTTP
Host: <CLIENT_URL_HOST:PORT>
Content-Type: application/x-www-form-urlencoded
inbound_address=23157
&router=verizon
&carrier=verizon
&message_id=1000000
&message_subid=1
&status=Carrier Success
&status_code=200
&device_address=11234567890
&reporting_key1=test_key1
&status_info=
&update_date=12/26/2019 17:10:00,000
&smscid=2r431oi3774guc2uh13uk5sl7sr4
&uaprof=
&client_message_id=
&date_format=MM/dd/yyyy HH:mm:ss,SSS
&channel=sms
&reporting_key2=uefg-h364-dr5h-euruyt
&smpp_errcd=2649
```

SMPP Error Codes

The following table lists the standard SMS SMPP submission and delivery error codes returned by Kaleyra and carrier systems in the US. These errors are returned in the delivery receipt to clients to better debug issues.

While the table below covers most common SMPP error codes, there are certain situtations where clients may receive additional error codes. In such cases please reach out to the Kaleyra Technical Support Team for more details.

10 DLC Specific Error Codes

Code	Description	Carrier	Source
2649	Quota Exceeded Error	T Mobile	Carrier
068	Spam Message Detected/Rejected	ATT	Carrier
069	Sending Limit Reached	ATT	Carrier
06A	Message has invalid tagging data	ATT	Carrier

SMS Submission and Delivery Errors

Code	Description	Explanation	Source	Action
00a	SMPP Invalid Address	The SMPP address (phone number) is malformed or invalid OR the shortcode (originator address) is not valid. If you are sure both the originator and destination address are valid, please check with Kaleyra support for resolution.	Carrier	Do Not Retry, Check
00b	Invalid address	The carrier system rejected the submission because the destination address is not a subscriber of the carrier. If you are sure the destination address belongs to the carrier, please contact Kaleyra support.	Carrier	Do Not Retry, Remove
0ff	Temporary SMSC Error	Carrier SMSC has operational issues and dropped the message. Please try again.	Carrier	Retriable, Short Term
40f	Delivery failed due to insufficent funds	This is an SMS delivery error that is sent to a prepaid device with insufficient funds to receive the standard rate SMS.	Carrier	Do Not Retry
41a	Invalid address	The phone number does not exist in the carrier system. If you are sure the destination address belongs to the specific carrier (reflected in carrier lookup service), please contact Kaleyra support.	Carrier	Do Not Retry, Remove

Code	Description	Explanation	Source	Action
41b	Blocked MDN for the shortcode	This errorcode happens when a consumer is sent an MT from a shortcode that is blocked per consumer's profile settings.	Carrier	Do Not Retry
4a4	Invalid VzW program_id	The product code (IPC) that is used for the MT message contains an invalid program_id for VzW. Please contact Kaleyra support.	Carrier	Retriable, ONLY after fixing the problem
001	Message too long	SMS Message is longer than what carrier can support.	Kaleyra	Do Not Retry
004	Message has been deleted	The message has been deleted in the carrier SMSC. This is occurring because the serving switch is returning cause code SMS termination denied. This indicates that the HLR is set to block all SMS termination. When the SMSC gets this cause code back the carrier deletes the message because it will never get delivered no matter how many delivery attempts are made.	Carrier	Do Not Retry, Remove
005	Delivery Failure	The message failed to reach the subscriber's handset.	Carrier	Do Not Retry
007	Unknown message state.	The message was invalid. This occurs when you attempt to send content to a handset which does not support that format. Or Invalid Registered Delivery Flag.	Kaleyra	Do Not Retry, Check
008	Message in Rejected State	Temporary SMSC error.	Kaleyra	Retriable, Short Term
014	Message Queue Full	Message queue is full. This is a passed-through carrier error.	Carrier	Retriable, Short Term
015	Insufficient Prepaid Balance	Verizon SurePay prepaid validation error from subscriber having insufficient prepaid balance to receive standard rate SMS message.	Carrier	Retriable, Long Term
023	Invalid Originator Address	The message was not delivered because the short code was not correctly provisioned on the carrier side.	Carrier	Do Not Retry, Check
024	Invalid Destination Address	The message was not delivered because the phone number does not appear in the carrier's subscriber database. If you are sure the destination address belongs to the carrier, please contact Kaleyra support.	Kaleyra	Do Not Retry, Check
035	Prepaid Balance Exceeded	A Verizon prepaid subscriber has an insufficient balance for receiving a standard rate SMS message.	Carrier	Retriable, Long Term

Code	Description	Explanation	Source	Action
045	Insufficient Prepaid Balance	See error 15.	Verizon Wireless	Retriable, Long Term
045	Submit failed	All addresses are rejected by the Subscriber Policy Rules. The following rules are applied: MVNO policy, Subscriber Type policy, Subscriber Account Status policy, Subscriber Blacklist policy.	T-Mobile	Do Not Retry
045	Submit failed	The shortcode is not active.	Other Carriers	Do Not Retry, Check
064	ESME Receiver Reject Message Error Code	Temporary SMSC error.	Carrier	Retriable, Short Term
065	Undeliverable	This is a passed-through carrier error.	Carrier	Do Not Retry
066	Receiver reject	This is a passed-through carrier error.	Carrier	Do Not Retry
068	Message Queue Full	See error 63490.	Carrier	Retriable, Short Term
074	Submit Failure	The message failed to reach the subscriber's handset.	Carrier	Do Not Retry
099	General System Error	Verizon protocol or system availability error.	Carrier	Retriable, Short Term
403	Insufficient balance	This is a passed-through carrier error. The prepaid subscriber has a zero balance to receive standard rate SMS message.	Carrier	Retriable, Long Term
404	Invalid device address or callback number	The number is not a 10-Digit number or not a TELUS MDN.	Carrier	Do Not Retry, Remove
412	Shortcode inactive	The shortcode is not active on the carrier system.	Carrier	Do Not Retry, Check
413	Short Code Expired	The short code expired. No activity is allowed on this short code.	Carrier	Do Not Retry, Check
414	Short Code blocked	The shortcode is in blocked status on the carrier side.	Carrier	Do Not Retry, Check
415	Undeliverable	The program start date in the carrier tool is not reached yet.	Carrier	Retriable, ONLY after fixing the problem

Code	Description	Explanation	Source	Action
417	Blocked MDN	Do not retry. MDN blocked on PMG. The MDN is no longer active and should be removed from any subscriptions.	Carrier	Do Not Retry, Remove
418	Invalid Reseller	Do not retry. Not a valid Sprint MVNO or reseller for PMG. The MDN belongs to an unsupported reseller on Sprint.	Carrier	Do Not Retry, Remove
419	MDN Hotlined by Carrier	Do not retry. Carrier hotlined the MDN. Sprint has temporarily suspended the subscriber's services due to non-payment.	Carrier	Do Not Retry, Remove
421	Undeliverable	Unsupported mobile device. This is a Sprint Aircard or similar device that cannot accept SMS messages.	Carrier	Do Not Retry, Remove
423	Rejected	Indicates that for the given shortcode new subscribers will not be allowed to receive or send messages.	Carrier	Do Not Retry
425	Rejected	Indicates that for the given shortcode new subscribers will not be allowed to receive or send messages for premium programs. If this error happens for standard rate programs, please contact Kaleyra support team.	Carrier	Do Not Retry, Check
440	Deactivated Subscriber	Inactive phone number. This error also occurs if the subscriber has SMS blocking turned on. In both cases, these subscribers should be removed immediately.	Carrier	Do Not Retry, Remove
448	Undeliverable	The message was blocked by the Verizon content filtering mechanism because the subscriber's rating is less than the content rating of the PSMS message being sent. When it receives this error code from Verizon, Kaleyra will initiate an automatic free MT message as follows: "Free Message. Your request has been blocked. Only the account holder may change your access privileges." In response to this error, Kaleyra clients must remove the MDN from the affected subscription product.	Carrier	Do Not Retry
0C4	Invalid Optional Parameter Value	Invalid TLV Value. The data content of a TLV is invalid and cannot be decoded. Note: This is typically seen in cases where Program Id or campaign Id TLV is invalid in the case of T-Mobile.	Carrier	Retriable, ONLY after fixing the problem
024	Message delivery failed at External Entity	This is a generic system error retruned by the carrier Note : This error is seen for Sprint and ATT Only	Carrier	Do Not Retry, Check

Code	Description	Explanation	Source	Action
090	Invalid or Incorrect destination address	The carrier system rejected the submission because the destination address is not a subscriber of the carrier. If you are sure the destination address belongs to the carrier, please contact Kaleyra support. Note : This error is seen for Sprint and ATT Only	Carrier	Do Not Retry
091	Invalid or Incorrect Routing	The Message was blocked by the carrier due to incorrect routing configuration either on Kaleyra side or carrier side. Please contact Kaleyra support Note : This error is seen for Sprint and ATT Only	Carrier	Do Not Retry

Message Status API

When sending MT messages using the DirectTEXT API, Kaleyra's Messaging Gateway automatically assigns a Message ID for every message sent. Customers can choose to lookup the status of a particular message using the Message Status API. Please work with your Account Director or Technical Program Manager for more information.

This API currently has restricted usage limitations in place. API should not be used to lookup statuses in bulk. It is purely meant for troubleshooting individual messages and investigation.

Call Format

To check the status for a given message ID, simply make a HTTP request in the format below. Basic authentication will be required for all lookups.

Request Type: GET or POST

```
https://directtext.mgage.com/a2w_preRouter/checkMessageStatus?messageId={{messageId}}
```

Parameter	Description
messageId	Required field that represents the message ID of a message that was sent

Message Status Contents

The following POST parameters are sent in the response:

Code	Status
id	Kaleyra-assigned message ID for the MT sent
subId	If the message is split because it's longer than 160 characters, it will result in more than one status notification. The status notification for the first part of the message (or if the message is 160 characters or less, it will be for the entire message) will have a message_subid of 0. The second part of the message will have a subid of 1, the third of 2, etc.
status	The current status of the message as it relates to the carrier network. See "Codes and Definitions" below for status codes.
statusCode	Numeric value that denotes success, failure or intermediate status of the message.
update_date	A date/time stamp reflecting the system time on the Kaleyra hosted router when the carrier status/delivery receipt was received.

Example:

```
https://directtext.mgage.com/a2w_preRouter/checkMessageStatus?messageId=235524456530
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<statusCheckResponseClient>
    <statuses>
        <routerLogStatusClient>
            <id>237554406587</id>
            <subId>0</subId>
            <status>Carrier Success</status>
            <statusCode>200</statusCode>
            <updateDate>2020-08-27T19:48:54Z</updateDate>
        </routerLogStatusClient>
    </statuses>
</statusCheckResponseClient>
```

CARRIER LOOKUP API

When sending MT messages using the DirectTEXT API, Kaleyra's Messaging Gateway automatically performs a carrier lookup to determine the carrier associated with a mobile number, and thus routes the message to the appropriate carrier for delivery to the mobile device. In addition, customers can choose to perform their own carrier lookups before sending a message. This is a value-added service provided by Kaleyra. Carrier lookups are beneficial for cleaning up your subscriber lists, or to check the validity of phone numbers that your subscribers have submitted via a web form or a point-of-sale application. Please work with your Account Director or Technical Program Manager for more information on this service and to obtain your API credentials.

Call Format

To perform carrier lookups using the API, simply make a HTTP request in the format below. We recommended not exceeding 100 mobile numbers within a single API request. Basic authentication will be required for all lookups.

Request Type: GET or POST

```
https://directtext.mgage.com/a2w_preRouter/carrierLookup?mn={{mobilenumber}}
```

Parameter	Description
mn	Represents the phone number that you wish to lookup the carrier information for

The API Response is in a JSON format with the following parameters. The API will return a *401 Unauthorized* HTTP error if the user provides invalid credentials, or provides no credentials at all. In the case of an error, the **success** element will be "false". An additional code and message element will be present on the response, and results will be absent. Refer to the table below for global and individual status/error codes. For all global errors, omit the entire **results** element from the response. All success parameters should be set to "true" if and only if its respective *statusCode* is 0. All other numbers for status codes should set **success** to "false".

Parameter	Description
mobileNumber	Represents the phone number that you included in the lookup request
carrierName	Represents the wireless carrier network associated with the phone number

Global Status Code Reference

Status Code	Message	Reason
0	Success	Successful request
100	No valid phone numbers	User submitted a request with 0 valid phone numbers. If at least one number is valid, this error does not apply.
101	System currently unavailable. Please retry later.	Carrier lookup is unavailable; carrier lookup can't connect to external providers.

Individual Status Code Reference

Status Code	Message	Reason
0	Success	Successful request
200	Invalid phone number	User submitted a phone number that does not meet E.164 guidelines.
201	Unknown carrier	Carrier lookup was performed on the phone number, but it returned null or an otherwise unknown carrier (phone number may be a land line).

Example: Carrier Lookup Request for Valid and Invalid Phone Numbers

```
https://directtext.mgage.com/a2w_preRouter/carrierLookup?mn=14045551214&mn=13109577318
```

Response

```
{"success":true,"statusCode":0,"results":[{"success":true,"statusCode":0,"mobileNumber":"
13109577318","carrierName":"sprint"},{"success":false,"statusCode":200,"mobileNumber":"14
04555121","carrierName":"Invalid phone number"}]}
```

Example: Carrier Lookup Request for a Landline Number

```
https://directtext.mgage.com/a2w_preRouter/carrierLookup?mn=14049009306
```

Response

```
{"success":true,"statusCode":0,"results":[{"success":true,"statusCode":0,"mobileNumber":"
14049009306","carrierName":"null"}]}
```

DROP-OFF DELIVERY

Drop-Off Delivery provides customers with the ability to send SMS or MMS messages based on the content of a text file that is delivered to Kaleyra via FTP. To use Drop-Off Delivery, an Kaleyra customer transmits a text file to an Kaleyra FTP server. This text file contains the SMS or MMS messages to be sent, the message recipients, and other information such as the delivery schedule. Kaleyra supports Secure FTP (SFTP), which encrypts both commands and data, preventing passwords and sensitive information from being transmitted in the clear over a network. Your Kaleyra Technical Program Manager will supply the FTP server and login credentials. Kaleyra customers have the option to further secure a file with PGP encryption. Public keys can be provided to customers that wish to send encrypted files via FTP to Kaleyra. This enables added security to files as only Kaleyra can decrypt the files with the private key.

Note: Supported file extensions include .txt and .csv. Files may also be zipped with a .zip extension. If an alternative file extension is required, please contact your Technical Program Manager.

SMS File Format

Drop-Off Delivery text files must be in the following format for SMS.

```
email: <recipient>
shortcode: <short code>
start_date: <start date>
end_date: <end date>
msg_subject: <subject>
msg_body: <default message body>
phone:
<message entries>
```

Example: SMS File

Below is an example of a Drop-Off Delivery text file that would be used to send payment reminder messages via SMS. In this example, 5 messages are sent to 5 recipients via the short code 12345. The messages will be sent on January 20, 2018 at 10:00 AM EST.

```
email: smserrors@acme.com
shortcode: 12345
start_date: 01/20/2018 10:00 AM
end_date: 01/21/2018 10:00 AM
msg_subject: Balance reminder
phone:
+14045550001;;Reminder from Acme: balance of $23.93 is due 1/29.
+14045550020;;Reminder from Acme: balance of $49.17 is due 1/29.
+14045550300;;Reminder from Acme: balance of $36.23 is due 1/29.
+14045554000;;Reminder from Acme: balance of $109.61 is due 1/29.
+14045555000;;Reminder from Acme: balance of $58.20 is due 1/29.
```

An explanation of each field in the file appears in the table below.

Field	Description	Required
email	The email address that should receive any error messages during the transmission.	Yes
msg_body	The field is optional. msg_body is the default value that is set for the message text sent to each MSISDN in the recipient line. If no message text is specified in the individual recipient line, this default value is used for all recipients.	No
msg_subject	The subject for the SMS message	No
mt_handler	Only required when sending MMS. Value should always be MU.	No
shortcode	The short code over which the messages should be sent. This must be a valid short code assigned to your organization within Kaleyra's systems.	Yes
start_date	The date and time at which the messages transmission should start. If the specified time has already passed when the file is delivered via FTP, the messages will be sent as soon as possible. That timing is affected by other scheduled messages. Files should be received at least 24 hours prior to the scheduled start date. The date and time is based on the Eastern time zone, and should be specified in mm/dd/yyyy hh:mm AM/PM format (for example, 12/11/2009 11:00 AM).	No

Field	Description	Required
end_date	The date and time at which the message transmission should end. If the end date is reached before all messages have been sent, the remaining messages will be dropped. The date and time is based on the Eastern time zone, and should be specified in mm/dd/yyyy hh:mm AM/PM format (for example, 12/11/2009 11:00 AM)	No
phone	Following the phone heading include a separate line for each recipient and message you wish to send. The format of the message lines is: `<mobilenum>;<carrier>;<message>;<userdata1>;<userdata2>;<client_recip_id>`	Yes

Field	Description
mobilenum	Required field, the mobile number of the message recipient. Format must follow the E.164 specification.
carrier	Optional field that lists the carrier the mobile number belongs to, normally left blank.
message	The SMS content to be delivered to the recipient. Note: The content cannot contain the ; character.
userdata1	Optional customer-defined strings that can be used for reporting purposes.
userdata2	Optional customer-defined strings that can be used for reporting purposes.
client_recip_id	Optional customer-defined strings that can be used for reporting purposes.

MMS File Format

Drop-Off Delivery text files must be in the following format for MMS.

```
email: <recipient>
shortcode: <short code>
start_date: <start date>
end_date: <end date>
msg_subject: <subject>
mms_url: <url to be sent to all recipients>
phone:
<message entries>
```

Example: MMS File

Below is an example of a Drop-Off Delivery file that would be used to send a marketing message via MMS. In this example, the same multi-media content is sent to 5 recipients via the short code 12345. The messages will be sent on January 20, 2019 at 10:00 AM EST.

```
email: smserrors@acme.com
shortcode: 12345
start_date: 01/20/2019 10:00 AM
end_date: 01/21/2019 10:00 AM
mms_url: http://www.fakeurl.biz
phone:
+14045550001
+14045550020
+14045550300
+14045554000
+14045555000
```

Note: Supported file extensions include .txt and .csv. Files may also be zipped with a .zip extension. If an alternative file extension is required, please contact your Technical Program Manager.

An explanation of each field in the file appears in the table below. Unless explicitly specified, all fields must be present in the file, even if they are blank.

Field	Description	Required
email	The email address that should receive any error messages during the transmission.	
mms_url	The url of the multi-media content that will be delivered to all recipients in the file.	
msg_subject	The subject for the MMS message	
mt_handler	Value should always be MU.	

Field	Description	Required
shortcode	The short code over which the messages should be sent. This must be a valid short code assigned to your organization within Kaleyra's systems.	
start_date	The date and time at which the messages transmission should start. If the specified time has already passed when the file is delivered via FTP, the messages will be sent as soon as possible. That timing is affected by other scheduled messages. Files should be received at least 24 hours prior to the scheduled start date. The date and time is based on the Eastern time zone, and should be specified in mm/dd/yyyy hh:mm AM/PM format (for example, 12/11/2009 11:00 AM).	
end_date	The date and time at which the message transmission should end. If the end date is reached before all messages have been sent, the remaining messages will be dropped. The date and time is based on the Eastern time zone, and should be specified in mm/dd/yyyy hh:mm AM/PM format (for example, 12/11/2009 11:00 AM)	
phone	Following the phone heading include a separate line for each recipient and message you wish to send. The format of the message lines is: `<mobilenum>;<carrier>;<message>;<userdata1>;<userdata2>;<client_recip_id>`	

Field	Description
mobilenum	Required field, the mobile number of the message recipient. Format must follow the E.164 specification.
carrier	Optional field that lists the carrier the mobile number belongs to, normally left blank.
message	The SMS content to be delivered to the recipient. Note: The content cannot contain the ; character.
userdata1	Optional customer-defined strings that can be used for reporting purposes.
userdata2	Optional customer-defined strings that can be used for reporting purposes.
client_recip_id	Optional customer-defined strings that can be used for reporting purposes.

APPENDIX A: DirectTEXT API CODE EXAMPLES

The section contains several examples of using the DirectTEXT Message Delivery API with various utilities and programming languages. In all of these examples, you will need to replace your-username, your-password, 12345, and 14045552900 with your actual username, password, short code, and recipient mobile phone number, respectively.

wget Command Line Example

The command line below sends a message using the wget utility and prints the result to standard output (stdout). Please note that the command has been formatted on multiple lines for clarity.

```
wget -O- \
    --user your-username \
    --password your-password \

"https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&recipient=140455
52900&body=Test%20message%20sent%20via%20Air2Web%20DirectTEXT%20from%20wget"
```

cURL Command Line Example

The command line below sends a message using the cURL utility and prints the result to standard output (stdout). Please note that the command has been formatted on multiple lines for clarity.

```
curl -u your-username:your-password \

"https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&recipient=140455
52900&body=Test%20message%20sent%20via%20Air2Web%20DirectTEXT%20from%20curl"
```

Perl

Sending a Message

This example Perl script uses the LWP and LWP::UserAgent packages to make an authenticated HTTP request to the DirectTEXT server.

```
#!/usr/bin/perl
#
# Example of sending messages through the Kaleyra DirectTEXT
# message delivery API with Perl.
```

```
#
use strict;
use LWP;
use LWP::UserAgent;

my $user = 'your-username'; my $pass = 'your-password';
my $baseURL = "https://directtext.mgage.com/a2w_preRouter/httpApiRouter?";
my $replyTo = "12345";
my $recipient = "14045552900";
my $messageBody = "Test message sent via Kaleyra DirectTEXT from Perl";

# URL encode message body
$messageBody =~ s/([^A-Za-z0-9])/sprintf("%%%02X", ord($1))/seg;

# Construct URL
my $URI = $baseURL;
$URI .= "reply_to=" . $replyTo;
$URI .= "&recipient=" . $recipient;
$URI .= "&body=" . $messageBody;

# Define user agent
my $ua = LWP::UserAgent->new();

# Create authenticated request
my $request = HTTP::Request->new(GET => $URI);
$request->authorization_basic($user, $pass);

# Send request
my $response = $ua->request($request);

# Get response
my $content = $response->content();

# Print result
my $responseCode; my $responseStatus;

if ($content =~ /<code>(\d+)<\/code><description>([\s\S]+)<\/description>/i) {
    $responseCode = $1;
    $responseStatus = $2;
}

if ($responseCode == "100") {
    print("Message sent successfully!\n$content\n");
} else {
    print("Error occurred: $responseCode, $responseStatus\n$content\n");
}
```

Receiving a Message or Status Notification

The example below uses the CGI module to extract components of a MO or status notification and write the values to a log file called “DirectTEXTReceived.txt”.

```
#
# Example of processing Kaleyra DirectTEXT MO/status notifications with Perl.
#
use strict;
use CGI;

# Load POST data into a new CGI object
my $query = new CGI;

# Extract values and create timestamped output line my $output;
$output = scalar(localtime(time)) . " -- ";
$output .= "Carrier: " . $query->param('carrier') . "; ";
$output .= "Channel: " . $query->param('channel') . "; ";
$output .= "Device address: " . $query->param('device_address') . "\n";

# Write output to file
open(FILEH, ">>DirectTextReceived.txt") || die("Cannot open log file: $!\n");
print(FILEH $output);
close(FILEH);
```

Java Example

Sending a Message

This example Java console application uses the built-in URL and Authenticator classes to make an authenticated HTTP request to the DirectTEXT server.

```
//
// Example of sending messages through the Kaleyra DirectTEXT
// message delivery API with Java
//
```

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Authenticator;
import java.net.PasswordAuthentication;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class DirectTEXTSend {
    public static void main(String[] args) {
        try {
            final String user = "your-username";
            final String pass = "your-password";
            String baseURL = "https://directtext.mgage.com/a2w_preRouter/httpApiRouter?";
            String replyTo = "12345";
            String recipient = "14045552900";
            String messageBody = "Test message sent via Kaleyra DirectTEXT using Java";

            // URL encode message body
            messageBody = URLEncoder.encode(messageBody, "UTF-8");

            // Construct URL
            StringBuffer URI = new StringBuffer();
            URI.append(baseURL);
            URI.append("reply_to=").append(replyTo);
            URI.append("&recipient=").append(recipient);
            URI.append("&body=").append(messageBody);

            // Create authenticator for http request
            String result = "";
            Authenticator.setDefault(new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(user, new String(pass).toCharArray
());
                }
            });

            // Open connection and send request
            URL url = new URL(URI.toString());
            URLConnection conn = url.openConnection();
            // Get the response
            BufferedReader rd = new BufferedReader(new InputStreamReader(conn
.getInputStream()));
            StringBuffer sb = new StringBuffer();
            String line;
            while ((line = rd.readLine()) != null) {
                sb.append(line);
            }
            rd.close();

            // Print results
            result = sb.toString();
            String responseCode = "";
            String responseMessage = "";
            Matcher matcher = Pattern.compile("<code>(\\d+)<\\/code><description>([\\s
\\S]+)<\\/description> ").matcher(result);
            if (matcher.find()) {
                responseCode = matcher.group(1);
                responseMessage = matcher.group(2);
            }
            if (responseCode.equals("100")) {
                System.out.println("Message sent successfully!\n" + result + "\n");
            } else {
                System.out.println("Error occurred: " + responseCode + ", " +
responseMessage + "\n" + result + "\n");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Receiving a Message or Status Notification

The example Java sevlet below extracts components of a MO or status notification and write the values to a log file called “DirectTEXTReceived.txt”. The parameters for the MO or status notification are contained in the request object and can be accessed using the getParameter() method.

```
//
// Example of processing Kaleyra DirectTEXT MO/status notifications with Java.
//

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

public class DirectTEXTRecv extends HttpServlet {
    public void doPost(
            HttpServletRequest request,
            HttpServletResponse response
    ) throws ServletException, IOException {

        response.setContentType("text/html");
        String output;
        output = new Date().toString() + " -- ";
        output += "Carrier: " + request.getParameter("carrier") + "; ";
        output += "Channel: " + request.getParameter("channel") + "; ";
        output += "Device address: " + request.getParameter("device_address") + "; ";

        FileWriter outFile = new FileWriter("DirectTextReceived.txt", true);
        PrintWriter out = new PrintWriter(outFile);
        out.println(output);
        out.close();
    }
}
```

PHP Example

Sending a Message

This example PHP script uses the cURL PHP module to make an authenticated HTTP request to the DirectTEXT server.

```
<?php
```

```php
//
// Example of sending messages through the Kaleyra DirectTEXT
// message delivery API with PHP.
//

if (function_exists("curl_init"))
{
    // initialize a new curl resource
    $ch = curl_init();
    $user = 'your-username';
    $pass = 'your-password';
    $baseURL = "https://directtext.mgage.com/a2w_preRouter/httpApiRouter?";
    $replyTo = "12345";
    $recipient = "14045552900";
    $messageBody = "Test message sent via Kaleyra DirectTEXT from PHP";

    // URL encode message body
    $messageBody = urlencode($messageBody);

    $URI = $baseURL;
    $URI .= "reply_to=" . $replyTo;
    $URI .= "&recipient=" . $recipient;
    $URI .= "&body=" . $messageBody;

    // Set URL to connect to
    curl_setopt($ch, CURLOPT_URL, $URI);

    // Set authentication options
    curl_setopt($ch, CURLOPT_USERPWD, $user . ":" . $pass);

    // Set header supression
    curl_setopt($ch, CURLOPT_HEADER, 0);

    // Disable SSL peer verification curl_setopt($ch,
    CURLOPT_SSL_VERIFYPEER, false);

    // Indicate that the message should be returned to a variable
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    // Make request
    $content = curl_exec($ch);
    $responseCode = "";
    $responseStatus = "";

    if (preg_match('/<code>(\d+)<\/code><description>([\s\S]+)<\/description>/',
$content, $matches))
    {
        $responseCode = $matches[1];
        $responseStatus = $matches[2];
    }

    if ($responseCode == "100")
    {
        print("Message sent successfully!\n$content\n");
    }
    else {
        print("Error occurred: $responseCode, $responseStatus\n$content\n");
    }

    // Clean up
    curl_close($ch);
}
else
{
    print("ERROR: curl library is not installed");
}
?>
```

Receiving a Message or Status Notification

The example below extracts components of a MO or status notification and write the values to a log file called “DirectTEXTReceived.txt”. The parameters for the MO or status notification are contained in the $_POST global variable.

```php
<?php
//
// Example of processing Kaleyra DirectTEXT MO/status notifications with PHP.
//

# Extract values and create timestamped output line
$output = date("r", time()) . " -- ";
$output .= "Carrier: " . $_POST['carrier'] . "; ";
$output .= "Channel: " . $_POST['channel'] . "; ";
$output .= "Device address: " . $_POST['device_address'] . "\n";

# Write output to file
$fh = fopen("DirectTEXTReceived.txt", 'a') or die("can't open file");
fwrite($fh, $output);
fclose($fh);
?>
```

C# Example

Sending a Message

This example C# console application uses the WebRequest and NetworkCredential classes to make an authenticated HTTP request to the DirectTEXT server.

Note: You must manually add a reference to the System.Web class in Visual Studio for this code to compile. In the Solution Explorer, right click **References and select Add Reference. On the .NET tab, select System.Web and click the OK** button.

```csharp
//
// Example of sending messages through the Kaleyra DirectTEXT
// message delivery API with C#.
//

using System;
using System.Text.RegularExpressions;
using System.Text;
using System.Net;
using System.IO;
using System.Web;

namespace DirectTEXTSend
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
        String user = "your-username";
        String pass = "your-password";
        String baseURL = "https://directtext.mgage.com/a2w_preRouter/httpApiRouter?";
        String replyTo = "12345";
        String recipient = "14045552900";
        String messageBody = "Test message sent via Kaleyra DirectTEXT via C#";

        // URL encode message body
        messageBody = System.Web.HttpUtility.UrlEncode(messageBody);

        // Construct URL
        StringBuilder URI = new StringBuilder();
        URI.Append(baseURL);
        URI.Append("reply_to=" + replyTo);
        URI.Append("&recipient=" + recipient);
        URI.Append("&body=" + messageBody);

        // Create a request for the URL and set credentials
        WebRequest request = WebRequest.Create(URI.ToString());
        request.Proxy = null;
        request.Credentials = new NetworkCredential(user, pass);

        // Get the response.
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();
        Stream dataStream = response.GetResponseStream();
        StreamReader reader = new StreamReader(dataStream);

        string responseFromServer = reader.ReadToEnd();

        // Print response
        string responseCode = "";
        string responseMessage = "";

        Match match = Regex.Match(responseFromServer,
            @"<code>(\d+)<\/code><description>([\s\S]+)<\/description>");
        if (match.Success)
        {
            responseCode = match.Groups[1].Value;
```

```
                responseMessage = match.Groups[2].Value;
            }

            if (responseCode == "100")
            {
                Console.WriteLine("Message sent successfully!\n" + responseFromServer +
"\n");
            }
            else
            {
                Console.WriteLine("Error occurred: " + responseCode + ", " + responseMessage
+ "\n" + responseFromServer + "\n");
            }

            // Cleanup up
            reader.Close();
            dataStream.Close();
            response.Close();
            }
        }
}
```

Receiving a Message or Status Notification

The example below uses the codebehind file of an ASP.NET page to extract components of a MO or status notification and write the values to a log file called “DirectTEXTReceived.txt”. The parameters for the MO or status notification are contained in the Request.Form object.

```
//
// Example of processing Kaleyra DirectTEXT MO/status notifications with ASP.NET and C#.
//
using System;
using System.Web;
using System.IO;

namespace DirectTEXTRecv
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            String output;
            output = DateTime.Now.ToString("");
            output += "Carrier: " + Request.Form["carrier"] + ";";
            output += "Channel: " + Request.Form["channel"] + ";";
            output += "Device address: " + Request.Form["device_address"] + "; ";

            TextWriter tw = new StreamWriter("DirectTEXTReceived.txt ");
            tw.WriteLine(output);
            tw.Close();
        }
    }
}
```

ASP Example (VBScript)

Sending a Message

This example VBScript-based ASP page uses the XmlHttp ActiveX object to make an authenticated HTTP request to the DirectTEXT server.

```
<%@ Language=VBScript %>
<%
'
' Example of using the Kaleyra DirectTEXT message delivery API from ASP using VBScript.
'
user = "your-username"
pass = "your-password"

baseURL = "https://directtext.mgage.com/a2w_preRouter/httpApiRouter?"
replyTo = "12345"
recipient = "14045552900"
messageBody = "Test message sent via Kaleyra DirectTEXT from VBScript"

' URL encode message body
messageBody = Server.URLEncode(messageBody)

' Construct URL
URI = URI & baseURL
URI = URI & "reply_to=" & replyTo
URI = URI & "&recipient=" & recipient URI = URI & "&body=" & messageBody

Set http = Server.CreateObject("Microsoft.XmlHttp")
http.open "GET", URI, False, strUserID, strPassword
http.send ""

Response.write(http.responseText)
Set http = Nothing
%>
```

Receiving MO Messages and Status Notifications

The example below extracts components of a MO or status notification and write the values to a log file called “DirectTEXTReceived.txt”. The parameters for the MO or status notification are accessed using Request.Form.

```
<%@ Language=VBScript %>
<%
'
' Example of processing Kaleyra DirectTEXT MO/status notifications with ASP and VBScript
'

output = Date & " " & Time & " -- ";
output = output & "Carrier: " & Request.Form("carrier") & "; "
output = output & "Channel: " & Request.Form("channel") & "; "
output = output & "Device address: " & Request.Form("carrier")

Set objFSO = CreateObject("Scripting.FileSystemObject")

Set objTextFile = objFSO.OpenTextFile("DirectTEXTReceived.txt", 8, True)
objTextFile.WriteLine(output)
objTextFile.Close
%>
```

APPENDIX B: USING EMOJIS IN MESSAGING

DirectTEXT API enables users to send emojis in the body text of a message. This guide details the list of multi-byte values DirectTEXT users can use in their API calls. SMS text messages contain a maximum of 1120 bits. Body text is limited to either 160 or 70 characters (depending on the language used). The API requires the values for emoji be UTF-8 URL Encoded. Use of any other value will cause the message to fail.

GSM Encoding

The character set in Latin-based languages (English, Spanish, etc) contain 7 bits per character. Therefore, 1 SMS in English could have a max of 160 characters.

UTF Encoding

Unicode Transformation Format (UTF) encoding (Chinese, emoji, Arabic, Japanese, etc) use 16 bits per character. Therefore, 1 SMS using emoji or any other Unicode encoding could have a max of 70 characters.

Message Splitting

Messages containing more than 160 or 70 characters are automatically split. GSM encoded messages (Latin-based languages) are split into 153 character chunks. The remaining 7 characters are used for segmentation info and to concatenate the individual messages back together on the handset. UTF encoded messages (emoji, Arabic, etc) are split into 67 character chunks. The remaining 3 characters are used for segmentation info and to concatenate the individual messages back together on the handset.

Example: Sending Multiple 4-Byte Emojis to Recipients

The following URL will send the message "Thanks for your business. 👍 😃" over short code 12345 to three different mobile phone numbers: 14045552900, 14045553900, and 14045554900.

```
https://directtext.mgage.com/a2w_preRouter/httpApiRouter?reply_to=12345&body=Thanks%20for
%20your%20business.%20%F0%9F%91%8D%20%F0%9F%98%83&recipient=14045552900&recipient=1404555
3900&recipient=14045554900
```

List of Emoji Values

A comprehensive list of Emojis along with Unicode Tables can be found here.