version 4.0

Mogreet Messaging System (MoMS) API

Anthony Rossano, Blake Dy, Paul Forsyth, Daniel Yanisse

Rev. 09-11-2012

contact: support@mogreet.com

555 Rose Avenue Studio G Venice, CA 90291

Table of Contents

Overview

The Mogreet Messaging System (abbreviated **MoMS)** is Mogreet, Inc.'s core content management, user and list management, transcoding, and message delivery platform, serving millions of SMS (text messages), MMS (multimedia messages) and mobile video views to mobile networks in the U.S. and internationally each month.

This document describes the programmatic APIs available to integrate messaging, transcoding, content delivery and other functionality into your applications and services.

Mogreet offers a full-service enterprise platform license, and a simpler pay-as-you-go developer API service. The web GUI for API developers is called the Developer Dashboard, and is available here:

http://developer.mogreet.com/dashboard

The web interface to the Enterprise Platform, including analytics, reporting, content management, targeting and more is called the Campaign Manager (CM), and is located at:

http://connect.mogreet.com

Many administrative functions, including campaign setup, video uploading and message testing can be performed through the CM. Please ask your account manager for a login and access.

The Mogreet APIs are simple, RESTful HTTP calls, using either GET or POST, and can be made from any modern network connected application framework, web framework, social network app, and in fact from any device or system connected to the global internet.

The APIs are divided into these main categories:

moms : The moms APIs initiate sends, blasts and other messaging.

cm: The CM APIs enable programmatic campaign management

Security Requirements

All requests to the MoMS APIs require a client account in MoMS system, and a client id and a token. If you are a developer you can sign up online at http://developer.mogreet.com for a free trial (no credit card or other billing info required). If you are an enterprise client and need credentials, please contact support@mogreet.com , your client services representative, or call account services at 310.566.0713.

Client ID & Token

Developer clients will receive the Client ID and Token during signup, and can retrieve these at any time in the developer dashboard.

Enterprise clients receive client id and token from the account manager during on-boarding, and may recover client id and/or token in your Campaign Manager web interface at:

http://connect.mogreet.com

Under 'Client Profile' in the advanced tab, or by emailing your account representative.

HTTPS

All of the calls made to the MoMS and CM APIs require the use of HTTP over SSL. This prevents a man in the middle from eavesdropping on your connections. Only the meta_store API allows regular HTTP calls, because the meta_store does not store or report sensitive information or allow sending of messages.

Server Side vs. Client Side

The MoMs and CM API calls should only be made from a server-side portion of your application. The MoMS and CM APIs should not be made on the web client side, i.e. through JavaScript or another browser language. Making Client-side API calls will expose your client_id and credentials to the public, will prevent IP white listing, and would expose your organization to malicious use of your services.

Only the stats API should be used client-side, because the stats API is designed to be called from JavaScript or another client-side language, and does not require the use of a client_id or token.

White Listing Source IPs

MOMS API platform can optionally check the source IP address of each request to verify that the originating server has permission to request the service. If you know your originating IP addresses, and they will remain constant, you may provide these IP addresses to limit access on your behalf to those IP blocks. Enterprise customers can contact your account representative, developer customers can provide IPs during signup or via the developer dashboard.

Response and Error Codes

The MoMS response to the API call has two parts: the HTTP response code, and specific error codes contained in the body of the HTTP response to provide additional information about the API request itself.

The API error code tells you more about your request to the MoMS or CM API service, and what specific aspect of your call might have caused the failure in the event that the call did not succeed. The API response code, status and message are delivered in XML in the body of the HTTP 200 response.

HTTP Response Codes

Response Code	Description
200	The request was processed successfully. Check the response code and message of the response XML for additional information regarding the request. See the API Error Codes below for additional information on the response codes.
4xx	The request was rejected by the API service. In most cases, you'll receive this message if the source IP of the request is not white-listed or your client_id and token don't match.
5xx	An error occurred with our servers. Please retry your request. If the issue persists, contact support@mogreet.com

API Response Codes

All 200 HTTP responses contain an API response code, status and message.

Here is a sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<response status="error" code="445">
  <message>
    <![CDATA[The request was rejected. The entered "name" parameter is invalid.]]>
  </message>
</response>
```

Here is an exhaustive list of all the current codes.

Code	Description
1	Successful request.
400	The request was rejected for unknown reason.
403	Access Forbidden: bad client id or auth token, or ips not whitelisted
404	Unknown request. Verify object.method is correct.
430	Missing “campaign_id” parameter.
431	Invalid “campaign_id” parameter.
432	The campaign does not exist. It has been deleted or the campaign_id is incorrect.
433	Missing “from” parameter.
434	Invalid “from” parameter.
435	Missing “to” parameter.
436	Invalid “to” parameter.
437	Missing "from_name" parameter.
438	Missing "to_name" parameter.
439	Missing the "message" parameter.
440	The entered promo code does not match any approved promotions.
443	request is missing a required parameter
444	invalid list_id parameter
445	The request was rejected. The entered "name" parameter is invalid.
460	Bad Media API request
461	Invalid Media Size (over 150 MB on upload)
462	Invalid Media Type (unsupported file)
463	Invalid Media Name
500	Internal fatal error uncaught.
600	Transaction terminated for unknown reasons.

601	Transaction terminated for unknown reasons.
602	User type unknown. Returns which user and user input.
603	User carrier unknown. Returns unsupported user phone.
604	User blacklisted by carrier. Returns user phone and carrier.
605	User carrier unsupported. Returns user phone and carrier.
606	User opted out. Returns user phone.
607	Content is not valid.

Response Format

The API responses are available in two flavors: XML and JSON.

XML

The default format for API responses is as XML in the body of the HTML response. The 'Content-Type' header will be set to 'text/xml'. Wherever there is message text in an XML element, it will be encapsulated in a CDATA block. Here is an example response from a system.ping command:

```
<?xml version="1.0"?>
<response code="1" status="success">
<message><![CDATA[pong]]></message>
```

JSON

You may also request the response to be returned in JSON format, by appending the 'format=json' parameter to the API call you are making. The JSON text will be returned in the HTML response body, with a ContentType of 'application/json'.

For instance, this call:

https://api.mogreet.com/moms/system.ping?client_id=133&token=ffaef12174023a9ba6fd880270ef5552&format=json

Has this response:

```
{
	"response": {
		"code": "1",
		"status": "success",
		"message": "pong"
	}
}
```

Test Connection

A quick and easy way to test successful integration over HTTPS is through the *system.ping* method. A successful test will return a HTTP 200 XML response with the text 'pong' in the

element response/message. Here's a sample call to get started (fill in with your credentials once your IPs have been white-listed):

https://api.mogreet.com/moms/system.ping?client_id=&token=

Calling the MoMS API

After completing the required setup process and optionally providing IP Addresses to white list, you are ready to test your connection to our API servers. Requests should be sent as an HTTPS GET or POST request to:

```
https://api.mogreet.com/moms/object.method
```

With the required parameters for each *object.method* listed below.

All requests require SSL/TLS protocol to provide encryption and secure identification of the server. In addition, your client id and token are required parameters for every request. The following example shows the common structure of all HTTPS GET requests understood by the MoMS and CM API servers:

```
https://api.mogreet.com/moms/object.method?client_id=123456&token=abcdef&parameter=value…
```

If your client_id and token do not match, or if some other request parameter does not belong to the given client, your request will fail with a '403 Forbidden' response. Another common cause of '403 Forbidden' is issuing the API call from an IP Address that is not whitelisted. Not all accounts require whitelisting. Ask developer support (support@mogreet.com) if you are unsure whether your account requires access from white listed IPs.

Request	Response

example of mismatching id and token, ips not whitelisted, asked to access a resource not belonging to the authenticated client, etc	`<?xml version="1.0" encoding="UTF-8"?>`
	`<response status="error" code="0">`
	`<message>`
	`<![CDATA[Mogreet: 403 Forbidden]]>`
	`</message>`
	`</response>`

Overview of MoMS API Objects and Methods

The following is a simple table of the available SMS/MMS Messaging API calls. A description of each one follows, with explanations of all parameters, responses and errors.

Object	Method	Request URL:
system	ping	https://api.mogreet.com/moms/system.ping
transaction	send	https://api.mogreet.com/moms/transaction.send
transaction	lookup	https://api.mogreet.com/moms/transaction.lookup
transaction	mo	https://api.mogreet.com/moms/transaction.mo
user	getopt	https://api.mogreet.com/moms/user.getopt

user	setopt	https://api.mogreet.com/moms/user.setopt
user	uncache	https://api.mogreet.com/moms/user.uncache
user	info	https://api.mogreet.com/moms/user.info

MoMS Methods Full Description

System Object

The system object tests connectivity to MOMS API servers.

system.ping

Use the ping method to test connectivity and monitor the status of the MoMS API servers. Ping is useful to test your credentials and white-listed IPs.

Request:

https://api.mogreet.com/moms/system.ping

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/system.ping?client_id=12345&token=abcde	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[pong]]></message> </response>

Same, with the format=json flag:
 	
https://api.mogreet.com/moms/system.ping?client_id=12345&token=abcde&format=json
	{
	"response": {
	"code": "1",
	"status": "success",
	"message": "pong"
	}
	}

Transaction Object

Use these methods to send SMS and MMS messages and check the status of the send.

transaction.send

Use the send method to initiate an SMS or MMS transaction. The delivery of the transaction depends on your campaign setup (called the campaign flow). Clients can customize their campaign flow through the Campaign Manager or with the help of their account manager. A successful request returns a message id and hash associated with the transaction created from the request. It's important to record and track the response message id and hash if you want to check the status and history of the transaction using the *transaction.lookup* method later on.

Request:

https://api.mogreet.com/moms/transaction.send

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
campaign_id	An ID connected to a specific campaign setup in the Campaign Manager or provided by your account representative.
to	The mobile number (MSISDN) of the handset you would like to send to.
from	The mobile number (MSISDN) of the handset you would like to send from. Or the shortcode associated with the campaign. (Optional – if not included, this parameter will default to the shortcode associated to the campaign).
message	Depending on your campaign set up, the message presented to the "to" user.
content_id	An integer value associated to a piece of content ingested through the Campaign Manager. You'll find all your content ids under the media section. (Optional, used for SMS and MMS delivering audio, image or video)
content_url	A publicly accessible URL of an image, audio or video. MOMS will automagically ingest the content and deliver it as specified by the campaign flow. (Optional, used for SMS and MMS delivering audio, image or video)
to_name	DEPRECATED see udp_* parameter below - Depending on your campaign setup, you may provide the receiver's name along with the message.
from_name	DEPRECATED see udp_* parameter below - Depending on your campaign setup, you may provide the sender's name along with the message.
callback	If provided with a valid URL, any errors with the transaction will be sent to this

	URL via XML over HTTP. See description below.
udp_*	UDP stands for user defined parameter. You may pass in any number of udp_* parameters as you like. UDPs are customized to your campaign flow. Your account representative will work with you to get customize UDPs to fit your needs. See description below.

callbacks

If your request includes the callback argument , the callback URL will be sent an XML HTTP POST request with some information about the result of the send. Using this callback you can also receive error reports and take action. Here are examples of a successful and unsuccessful send:

Request	Response
a successful transaction.send	<?xml version="1.0"?> <response code="1" status="success"> <message><![CDATA[Content MMS Sent]]></message> </response>
a failed transaction.send	<?xml version="1.0"?> <response code="603" status="error"> <message><![CDATA[We could not determine the mobile phone carrier you are attempting to send to. Please make sure you entered the receiver's phone number correctly and try again. Thank you.]]></message> </response>

Note: you can also set up a callback URL to catch MO messages sent in by consumers to your campaign. See the section 'Processing MO messages'.

user defined parameters

Clients may also add user defined parameters (udp) to further customize their campaigns. Since campaigns differ from client to client and even from one client's campaign to another, the API allows you to customize the experience by submitting your own user defined parameters (udp). The parameters enable you to customize your message flow with our product team. The udp data is most commonly used to insert some custom text into the messaging dynamically: i.e. – product description or some other dynamic data.

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/transaction.send?client_id=12345&token=abcde&campaign_id=123&to=5551236789&f	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[Mogreet successfully

rom=5551236543&message=hello%20world&content_id=321	sent!]]></message> <message_id>123456789</message_id> <hash>a12b3c4d</hash> </response>
a bad MSISDN (mobile number) https://api.mogreet.com/moms/transaction.send?client_id=12345&token=abcde&campaign_id=123&to=1236789&from=5551236543&message=hello%20world&content_id=456	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="603"> <message><![CDATA[Invalid to destination]]></message> </response>

with response set to json:	{
	"response": {
	"code": "1",
	"status": "success",
	"message": "Your message is on its way.",
	"message_id": "118277728",
	"hash": "9g5efkxo"
	}
	}

Note: The message_id and hash elements in the return response are required in the *transaction.lookup* method (also see *user.transactions* to obtain a user's transaction message_id and hash).

transaction.lookup

The lookup method returns info, status, and history of the requested transaction. Here is a list of elements returned:

Name	Description
campaign_id	The campaign id associated to this transaction.
to	The mobile number (MSISDN) of the receiving handset.
from	The mobile number (MSISDN) of the sender or the shortcode associated to the campaign.
to_name	The name of the receiver.
from_name	The name of the sender.
content_id	The content id of the content sent to the receiver.
status	A text description of the transaction status.
history	A list of the events (text descriptions) of the transaction.

Request:

https://api.mogreet.com/moms/transaction.lookup

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
message_id	A unique ID returned from a successful *transaction.send* request or from *user.transactions* method.
hash	A hash returned from a successful *transaction.send* request or from a *user.transactions* method.

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/transaction.lookup?client_id=12345&token=abcde&message_id=123456789&hash=1a2b3c4d	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[transaction found]]></message> <campaign_id>123</campaign_id> … <status>complete</status> <history> <event>foo</event> <event>bar</event> <event>baz</event> </history> </response>
https://api.mogreet.com/moms/transaction.lookup?client_id=12345&token=abcde&message_id=123456789&hash=1a2b3c4e	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="0"> <message><![CDATA[transaction not found]]></message> </response>

transaction.mo

The mo method simulates exactly what happens when a user sends in a 'mobile originated' message to a given shortcode. In other words, if a user texts 'hello' to a shortcode and receives a content item back on the handset, the same result can be had by making a

transaction.mo call. The transaction.mo call 'triggers' the behavior of the campaign linked to the keyword in the transaction.mo call, just as a user MO does.

Request:

https://api.mogreet.com/moms/transaction.mo

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
shortcode	The shortcode which will process the transaction.mo. This shortcode should have a campaign running which matches a keyword sent in the 'message' of the MO.
from	The phone number to identify as the sender of the MO.
message	The message body of the MO. This should begin with a keyword which matches a campaign running on the shortcode. If no matching keyword is found, an error message will be returned via text to the mo sender.

A successful call will return an HTTP 200 with XML response.

The phone number specified will receive the same experience as if they had texted in to that shortcode. The MoMs will process the text of the MO, look for a keyword, find a match with active campaigns running on that shortcode, and then execute the campaign logic and return a predefined result to the phone number specified. If the message of the MO does not match any known keyword, a generic "Sorry, the command or keyword you texted in was unrecognized..." message will be returned via SMS to the phone number specified.

See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/transaction.mo?client_id=7&token=a487b7fba7180573cccc8958e45d4a59&from=3108889999&message=mogreet&shortcode=94444	<response code="1" status="success"> <message> <![CDATA[Text to Shortcode Processed]]> </message> <message_id>90220442</message_id> <hash>guf2s1vr</hash> </response>

mo callback

When a consumer sends a text (either SMS or MMS) to your shortcode or keyword, the platform will look up the campaign which is assigned to handle that incoming traffic, and will process it automatically, returning a message, or content, or performing some other message flow. This includes automatic Stop and Help behavior, and will opt people in and out of the campaign without intervention from the developer. However, there are many cases where the developer does want to receive notification of the text coming in, in addition to processing the MO (Mobile Originated message) with what ever message flow is attached to that campaign.

In this case, the developer can set up a global callback to receive notifications from the system when users text in. Those notifications can also handle passing image content in from a user. For instance, if the developer wanted to run an 'ugly dog' campaign, he could promote a call to action requesting users to take a picture of their pooch and send it in to a shortcode and keyword via MMS. The MoMS platform will receive the message, including any data, extract the picture, host it temporarily for your access, and pass an XML callback to the URL specified containing the user phone number, what message text was sent in and a URL to access the image data.

The mo callback URL is set up in the Campaign Manager, or during the developer on-boarding process. Contact your account manager for help implementing an MO callback. Below are examples of the XML returned when the MoMS receives an MO.

Request	Response
user texts in an image. multiple images are supported in a single MO!	<?xml version="1.0"?> <mogreet> <campaign_id>12345</campaign_id> <msisdn>1234567890</msisdn> <carrier><![CDATA[CarrierName]]></carrier> <message><![CDATA[Hello, World]]> </message> <subject><![CDATA[]]>My Subject</subject> <images> <image> <![CDATA[http://d2c.bandcon.mogreet.com/mo-mms/images/123_4567.jpeg]]> </image> </images> </mogreet>

User Calls

The user object exposes users' basic information, transactions, and campaign status.

user.getopt

The getopt method returns the opt in status of any mobile number. The following table gives a description of each status code:

Status	Status Code	Description
OPTEDIN	1	User is opted into the campaign.
OPTEDOUT	-2	User is opted out of the campaign.
GREY	0	User has been asked to join the campaign but has not yet confirmed joining.
UNSEEN	-1	User is unknown to the campaign.

Request:

https://api.mogreet.com/moms/user.getopt

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
number	A mobile number (MSISDN).
campaign_id	A campaign id to search on. (Optional - if excluded, returns all opt in statuses for the client's campaigns)

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/user.getopt?client_id=12345&token=abcde&number=5553217809&campaign_id=1	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[getopt request]]></message> <number><![CDATA[5553217809]]></number> <campaign id="1"> <status code="1">OPTEDIN</status> </campaign> </response>
https://api.mogreet.com/moms/user.getopt?client_id=12345&token=abcde&number=5553217809&campaign_id=2	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[getopt request]]></message> <number><![CDATA[5553217809]]></number> <campaign id="2"> <status code="-2">OPTEDOUT</status>

	</campaign> </response>
https://api.mogreet.com/moms/user.getopt?client_id=12345&token=abcde&number=5553217809&campaign_id=3	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[getopt request]]></message> <number><![CDATA[5553217809]]></number> <campaign id="3"> <status code="-1">UNSEEN</status> </campaign> </response>
https://api.mogreet.com/moms/user.getopt?client_id=12345&token=abcde&number=5553217809	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[getopt request]]></message> <number><![CDATA[5553217809]]></number> <campaign id="1"> <status code="1">OPTEDIN</status> </campaign> <campaign id="2"> <status code="-2">OPTEDOUT</status> </campaign> <campaign id="4"> <status code="0">GREY</status> </campaign> </response>

user.setopt

The setopt method sets the opt in status of any mobile number. The following table gives a description of each status code available to set:

Status	Status Code	Description
OPTEDIN	1	User is opted into the campaign.
OPTEDOUT	-2	User is opted out of the campaign.

Request:

https://api.mogreet.com/moms/user.setopt

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.

number	A mobile number (MSISDN) to opt in or out of a campaign.
campaign_id	A campaign id to change the opt in status for.
status_code	See the above table for available codes to use here.

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/user.setopt?client_id=12345&token=abcde&number=5553217809&campaign_id=1&status_code=-2	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[opt status updated]]></message> </response>

user.uncache

The uncache method clears the user carrier and handset info from the Mogreet cache. This method should be used in cases where a user recently switched carriers/networks or handsets.

Request:

https://api.mogreet.com/moms/user.uncache

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
number	A mobile number (MSISDN).

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/user.uncache?client_id=12345&token=abcde&number=5553217809	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[user uncached]]></message> </response>

user.info

The info method returns the user carrier and handset info if available.

Request:

https://api.mogreet.com/moms/user.info

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
number	A mobile number (MSISDN).

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/user.info?client_id=12345&token=abcde&number=5553217809	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[user info]]></message> <carrier id="3">AT&T</carrier> <handset id="2773">Apple iPhone</handset> </response>

user.transactions

The transactions method returns the user's transactions (open and closed). This method should be used for debugging purposes and customer service assistance in conjunction with *transaction.lookup*.

Request:

https://api.mogreet.com/moms/user.transactions

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
number	A mobile number (MSISDN).
campaign_id	A campaign id to search on. (Optional - if excluded, returns all opt in statuses for

	the client's campaigns)
start_date	(Optional) Narrow search by adding a date to start searching on [YYYY-MM-DD]
end_date	(Optional) Narrow search by adding a date to stop searching on [YYYY-MM-DD]

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response
https://api.mogreet.com/moms/user.transactions?client_id=12345&token=abcde&number=5553217809&campaign_id=1	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message><![CDATA[user transactions]]></message> <campaign id="1" name="campaign1"> <transaction message_id="123456" hash="a1b2c3e4" datestamp="2009-11-26 15:48:08"> <to number="5553217809">Joe</to> <from number="5559874564">Jane</from> </transaction> </campaign> </response>

MSISDN (user and handset) Lookup

user.lookup

The lookup method is unique in that it can return information about any Mobile user with service on the supported mobile carriers, whether or not that user is currently opted into a campaign. This information is useful for targeting content to users on a specific carrier or mobile device.

When you make the user.lookup API call, the MoMS will first determine the mobile carrier of the target user. This carrier information will be available in all cases except where the target user MSISDN is not on a supported mobile network, is a land line, or a Google voice number. The MoMS will next attempt to determine the specific handset device type. The device type can be determined in just over 50% of all cases, and this percentage rises to nearly 100% if the user has received content from Mogreet in the past.

This API call has specific fees per use of the API, so be mindful of exceeding the free usage allowed during implementation and testing.

Request:

https://api.mogreet.com/moms/user.lookup

with the following parameters:

Name	Description
client_id	Your client id. Log onto the Campaign Manager to access your client id.
token	Your token. Log onto the Campaign Manager to access your token.
number	A mobile number (MSISDN).

A successful call will return an HTTP 200 with XML response. See the following request and response examples:

Request	Response

`https://api.mogreet.com/moms/user.lookup?client_id=12345&token=abcde&number=2063695205`	`<?xml version="1.0"?>`
	`<response code="1" status="success">`
	`<message><![CDATA[user info]]></message>`
	`<number><![CDATA[12063695205]]></number>`
	`<carrier id="3"><![CDATA[AT&T]]></carrier>`
	`<handset id="6372"><![CDATA[Apple iPhone 4]]></handset>`
	`</response>`

The carrier information returned will include the Mogreet carrier id (in the above example, 3) as well as the text name associated with that carrier. In many cases, a carrier may have more than one carrier id, reflecting the consolidation in the carrier markets.

Here is a short table of the most common carriers:

2	T-Mobile
3	AT&T
4	Verizon Wireless
5	Sprint
6	Nextel
7	Alltel
8	Bell Mobility
9	TracFone (AT&T)
24	Metro PCS
27	Telus
28	Western Wireless
29	Rogers
36	US Cellular
41	Cellular One Dobson
45	NTELOS

Campaign Manager (cm) Methods

In addition to the MoMS methods primarily used for single sends and other single transactions, there are API calls available to programmatically control the Campaign Manager interface. The calls are useful for programmatically managing campaigns, including the creation and manipulation of lists, of keywords, of content, and initiating bulk sends of messages to given lists.

These API calls have a destination URL that contains the identifier 'cm' in the path, like this:

https://api.mogreet.com/cm/list.create?client_id=…….

There are four 'objects' in the Campaign Manager API set: Message, Keyword, Blast, and List.

Message Object

The Message object has methods to programmatically get and set the message contents of a campaign. This is used primarily in In-App messaging: the app makes the message.get API call to retrieve the text copy of the campaign and any media associated with the campaign, then displays the text and media to the user. It is also possible to SET the contents of a campaign in this matter – for instance, to replace the text of a 'joke of the day' campaign programmatically from a list of jokes.

message.get

Use the message.get method to read the text and media contents of a campaign from any IP connected device, application or service. Using this method you can consume the same campaign content that is used in an SMS or MMS campaign, in a handset app, social media app, web site, etc. This is a simple way to 'Publish' content to apps all over the mobile internet.

Request:

https://api.mogreet.com/cm/message.get

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
campaign_id	The campaign to fetch the message and media content from.

A successful call will return an HTTP 200 with XML response. The returned information contains the text of the campaign message, and if the campaign includes media such as image, audio, or video, the response will include a 'oneClick' URL that will perform device

detection and display or play the appropriate version of the content for the capabilities, resolution, OS, etc of the device.

See the following request and response examples:

Request	Response
https://api.mogreet.com/cm/message.get?client_id=7&token=abcde&campaign_id=18363	<?xml version="1.0" encoding="UTF-8"?> <response code="1" status="success"> <message>message retrieved</message> <message_text>Hello World !</message_text> <content> http://m.mogreet.com/oc/isgb8ng8z </content> </response>

message.set

Use the message.set method to update the existing text message of a campaign. For instance, in a campaign that returns the leaderboard positions for the Formula 1 racing season, message.set could be used after each race to update the text of the campaign to reflect the current standings. In this version of the API only the text of the campaign can be updated. Future versions will also allow media to be replaced.

Request:

https://api.mogreet.com/cm/message.set

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
campaign_id	The campaign to update with new message copy.
message	The text content (copy) which will replace the existing message text returned by the campaign.
content_id	The content_id of a media item you have ingested. It will be returned with the message.

A successful call will return an HTTP 200 with XML response.

See the following request and response examples:

Request	Response
https://api.mogreet.com/cm/message.set?client_id=7&token=abcdef&campaign_id=18363&message=hello%20Anthony	<?xml version="1.0" encoding="UTF-8"?> <response code="1" status="success"> <message>message modified</message> </response>

Keyword Object

The keyword object has methods that facilitate the use of multiple keywords within a client account. Keywords are the text labels used to identify campaigns within shortcodes: keywords must be unique within shortcodes, but each shortcode can have as many keywords assigned as necessary. For this reason, the keyword object has methods to check the availability of a given keyword on either the default shortcode for the client or a specified shortcode, in the event that the client has more than one shortcode to choose from.

Once a keyword has been checked to assure availability, that keyword can be added to either the default campaign, or to a specified campaign in the case where the client has more than one campaign. If a keyword is assigned to a campaign, it can also be removed, which frees up the keyword for reuse by any client on that shortcode.

A primary feature and purpose of the keyword API object is to allow multiple keywords to be bound to the same campaign. For instance, if developer 'Bob' has a campaign 1234 which has the default keyword 'january' on shortcode 31313, then when a user texts in 'january' to the shortcode 31313, the campaign 1234 will react to the input, and return a callback to the URL provided by developer Bob. Bob could then use the keyword.check API call to make sure that the keyword 'february' is available on shortcode 31313, and then the keyword.add API call to bind 'february' to Bob's campaign 1234. In this way a campaign can have multiple keywords covering different use cases.

keyword.list

Use the list method to enumerate all the keywords which belong to a given client. This call is useful for checking to see if a given list exists already, or just to check out what lists you do have access to.

Request:

https://api.mogreet.com/cm/keyword.list

With the following parameters:

Name	Description
client_id	Your client id.

token	Your token.
campaign_id (optional)	If omitted, the default campaign for the client is used. If included, the keywords bound to that campaign are listed.

A successful call will return an HTTP 200 with XML response. Note that there is always one MASTER keyword per campaign. The default keyword is marked with the attribute 'master="true"'. Master keywords cannot be deleted. All other keywords are identified with the attribute 'master="false"'

Response examples:

Request	Response
https://api.mogreet.com/cm/keyword.list?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&campaign_id=10699	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>keywords retrieved</message> <keywords> <keyword master="true">THEPLAY</keyword> <keyword master="false">SH1ZZL3</keyword> </keywords> </response>

keyword.check

Use the keyword.check method to test whether a provided keyword is already in use on the given shortcode.

Request:

https://api.mogreet.com/cm/keyword.check

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
shortcode (optional)	The shortcode to check for the keyword. Useful for accounts with multiple shortcodes. If omitted, the keyword check will use the default shortcode for the given client.
keyword	The exact text of the shortcode to test for existence.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/keyword.check?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&keyword=theking # the keyword is available	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>Keyword available</message> <available>true</available> </response>
https://api.mogreet.com/cm/keyword.check?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&keyword=news&shortcode=21534 # the keyword is taken on the specific shortcode!	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>Keyword taken</message> <available>false</available> </response>

keyword.add

Use the keyword.add method to bind another keyword to an existing campaign. MO (mobile originated) SMS or MMS messages which begin with that keyword will be routed to the given campaign and handled there.

Request:

https://api.mogreet.com/cm/keyword.add

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
campaign_id	The campaign_id to bind the new keyword to. The message flow specified in the campaign will be triggered by an MO SMS or MMS to the shortcode used by the campaign. Depending on the campaign setup, an HTTP callback can be sent to a URL specified in the campaign setup. See your account manager for details.
keyword	The exact text of the keyword to test for existence.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/keyword.add?client_id=1234&token=a487b7fba7180573cccc	<?xml version="1.0" encoding="UTF-8"?>

8958e45d4a60&keyword=theking&campaign_id=10699 #success!	<response status="success" code="1"> <message>keyword added</message> </response>
https://api.mogreet.com/cm/keyword.add?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&keyword=theking&campaign_id=10699 #failure, the keyword is now taken	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="448"> <message> <![CDATA[Unable to add keyword. Check availability.]]> </message> </response>

keyword.remove

Use the keyword.remove method to remove a keyword from an existing campaign. The keyword will become available for other campaigns on the same shortcode.

Request:

https://api.mogreet.com/cm/keyword.remove

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
campaign_id	The campaign_id to remove the keyword from.
keyword	The exact text of the keyword to match and remove.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/keyword.remove?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&keyword=theking&campaign_id=10699	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>keyword removed </message> </response>

List Object

The list object manages the creation, editing, use and destruction of lists belonging to a given client. Lists are lists of cell numbers available for use in campaigns, however, lists belong to clients – not to campaigns.

list.list

Use the list method to enumerate all the lists which belong to a given client. This call is useful for checking to see if a given list exists already, or just to check out what lists you do have access to.

Request:

https://api.mogreet.com/cm/list.list

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.

A successful call will return an HTTP 200 with XML response. Note that there is always one DEFAULT list per campaign, corresponding to the default keyword for that campaign. The default list is marked with type 'automatic'. Default 'automatic' lists cannot be destroyed or manipulated. All other lists are identified with type 'uploaded'. Each list element in the XML returned has an id attribute, which shows the internal list id for that element. You may use the list.list API call to enumerate the available lists and compare human readable names to recover the list id for future use.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.list?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>lists retrieved</message> <lists> <list id="9"> <name>TESTING</name>

	<type>automatic</type> </list> <list id="15"> <name>Jordan's First Blast list</name> <type>uploaded</type> </list> <list id="30"> <name>TEST BLAST LIST</name> <type>uploaded</type> </list> </lists> </response>

list.info

Use the info API call to find out more about a given list, including how many numbers are in the list, when the list was created, and when it was most recently updated.

Request:

https://api.mogreet.com/cm/list.info

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to add the numbers to. List_ids are returned in the list.list API call, or you may have stored the list_id returned when the list was created with the list.create API call.

A successful call will return an HTTP 200 with XML response. The <last used> xml response element (node) is a date/time field that represents the most recent time the list was blasted to.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.info?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=9999	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <list id="1125"> <name> <![CDATA[atrlist1]]> </name> <type>uploaded</type> <total>6</total>

	<created>2011-12-12T21:10:53Z</created> <updated>2011-12-12T21:10:53Z</updated> <last_used></last_used> </list> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message> </response>

list.create

Use the create method to add a new list. The list is created empty. In addition to the standard client_id and token params, pass in a name for the new list. The name parameter is required later to destroy the list, to prevent accidental list destruction.

Request:

https://api.mogreet.com/cm/list.create

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
name	The user readable name given to the list. URL-encode any spaces or other special characters(or avoid them altogether).

A successful call will return an HTTP 200 with XML response. The response will include the id of the list, for use in subsequent calls. It is possible to create multiple lists with the same human readable name, however each will have a separate id number.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.create?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&name=girlfriends	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>list created</message> <list id="1124"/> </response>

list.append

Use the append API call to add new numbers to a list. The call requires a list_id, to which numbers will be appended, and a list of mobile numbers delimited by commas. The maximum quantity of mobile numbers which may be appended in a single API call is 1024.

Request:

https://api.mogreet.com/cm/list.append

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to add the numbers to. List_ids are returned in the list.list API call, or you may have stored the list_id returned when the list was created with the list.create API call.
numbers	A comma delimited list of mobile numbers. You may omit the leading 1 for mobile numbers in the US. All US numbers should include an area code. All international numbers should include country code, city code and destination mobile number as per the MSISDN number spec.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.append?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=1234&numbers=6264872442,3109998888	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>request processed</message> <statistics> <created>1</created> <duplicate>0</duplicate> <rejected>0</rejected> </statistics> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message> </response>

list.prune

Use the prune API call to remove one or more mobile numbers from a list. The call takes a comma separated list and removes those numbers it finds to match in the target list_id. The maximum quantity of mobile numbers which may be pruned in a single API call is 1024.

Request:

https://api.mogreet.com/cm/list.prune

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to add the numbers to. List_ids are returned in the list.list API call, or you may have stored the list_id returned when the list was created with the list.create API call.
numbers	A comma delimited list of mobile numbers to be removed from the target list. You may omit the leading 1 for mobile numbers in the US. All US numbers should include an area code. All international numbers should include country code, city code and destination mobile number as per the MSISDN number spec.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.prune?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=1234&numbers=6264872442,3109998888	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>request processed</message> <statistics> <deleted>1</deleted> <not_found>0</not_found> <rejected>0</rejected> </statistics> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message>

	</response>

list.send

The list.send API method performs a bulk send of a given message and content to all the MSISDNs in the list, obeying the opt-in status of each number on the given campaign and shortcode. Because lists belong to clients, rather than specific campaigns, the method requires a list_id and a campaign_id.

The campaign_id given determines the message flow and the default message text and content. single campaign_ids are available from your account manager, or programmatically from the Campaign object and the campaign.list method (coming soon).

The message body can be overridden for this blast by specifying the optional message parameter.

The default content item can be overridden either by specifying the content_id parameter (for content already ingested into the Campaign manager) or by specifying the content_url parameter, which must be a publicly accessible URL pointing to an appropriate media type: audio file, image file, or video file. Allowed media types and specs are detailed in Appendix A.

The optional callback parameter may be set to a valid email address. If the callback parameter is present, an email will be sent at the conclusion of the list.send, containing delivery statistics and other information to assist in debugging.

Any number of optional parameters may also be passed to the list.send method by adding parameters which start with udp_ which stands for 'user defined parameter'. These will be ignored by the standard campaign types, however, custom development can be done to handle the additional parameters. For instance, if a campaign required the ability to define an alternate message to be used in the blast in certain situations, that alternate message could be passed in to the list.send API call by adding a udp_alternate_message parameter.

Request:

https://api.mogreet.com/cm/list.send

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to initiate a delivery to.
campaign_id	The id of the campaign which determines the message flow, and default behavior

	for the blast.
subject	The subject line of the message, used in MMS.
message	The text of the message you wish to deliver. The text should be in ASCII text, avoid special characters. Remember to URL-encode the entire API call string.
content_id	The Mogreet content_id describing the item of content (audio, image, video, etc) you wish to be delivered to the target handset. In most cases this will be delivered as a single MMS. In cases where the carrier or handset will not support the content type, an SMS with a link to the content hosted via mobile web page may be substituted. Optional, in cases of text only MMS.
content_url	A URL describing the location of a content file, to be ingested, transcoded, and used in the list.send, and delivered to all the handsets. Delivered as MMS where possible, as SMS with a link to a mobile page where no MMS is available. Content items are cached for reuse, and deleted from the cache after a period of time. See Appendix A for details on allowed file types and specifications. (Optional)
to_name	A label to be added where possible in the message subject. Ex. to_name=Toastmasters would add the address 'Toastmasters' to the message. (Optional)
from_name	A label to identify the sender. The messages will show the shortcode (or long code) of the campaign as the canonical sender, this label is used only in the message body or subject line. (Optional)
udp_*	user defined parameters which may be added to pass in specific information for custom campaign logic.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response

`https://api.mogreet.com/cm/list.send?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=1126&campaign_id=604&&subject=hello&message=helloworld`	`<?xml version="1.0" encoding="UTF-8"?>`
	`<response status="success" code="1">`
	`<message>queued</message>`
	`</response>`

list.download

The download API call takes the usual token and client_id, plus a list_id, and after validation, returns the contents of the list back to the caller in the response body.

Request:

https://api.mogreet.com/cm/list.download

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to retrieve.
format (optional)	An optional parameter "format=csv". When set, the data is returned in csv format (with Content-Type: text/csv)

A successful call will return an HTTP 200 with XML response. The list element is named <recipients> and each MSISDN returned is in a <recipient> element. The <time_zone> xml response element shows the time zone of the recipient parsed from the MSISDN given.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.download?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=9999	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>data exported</message> <recipients> <recipient> <number>2061235555</number> <time_zone>pacific</time_zone> </recipient> <recipient> <number>9998881212</number> <time_zone></time_zone> </recipient> </recipients> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message> </response>

list.empty

The list.empty API call completely empties a list from the account of the calling client. Any numbers stored in the list are no longer accessible via that list_id, but the list_id itself still remains, and can be re-used by adding new numbers with list.append. Emptying a list does not indicate that the constituent mobile numbers have been 'opted out' – merely that the list no longer contains any numbers.

Request:

https://api.mogreet.com/cm/list.empty

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to destroy.
name	The human readable name of the list, required here as a fail safe to avoid accidental list emptying, or emptying a different list than was intended.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.empty?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&list_id=1234name=my_subscribers	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>list reset</message> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message> </response>
a request with appropriate list_id, but non-matching name.	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="445"> <message> <![CDATA[The request was rejected. The entered "name" parameter is invalid.]]> </message> </response>

list.destroy

The destroy API call completely removes a list from the account of the calling client. Any numbers stored in the list are no longer accessible via that list_id. If the numbers also exist in a different list which still exists, they remain accessible via that list_id. Destroying a list does not indicate that the constituent mobile numbers have been 'opted out' – merely that the list is no longer available for list manipulation.

Request:

https://api.mogreet.com/cm/list.destroy

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
list_id	The unique identification number for the list you wish to destroy.
name	The human readable name of the list, required here as a fail safe to avoid accidental list destruction, or the destruction of a different list than was intended.

A successful call will return an HTTP 200 with XML response.

Response examples:

Request	Response
https://api.mogreet.com/cm/list.destroy?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60&name=my_subscribers	<?xml version="1.0" encoding="UTF-8"?> <response status="success" code="1"> <message>list destroyed</message> </response>
a request with someone else's list or a non existent list	<response status="error" code="444"> <message> <![CDATA[The request was rejected. The entered "list_id" parameter is invalid.]]> </message> </response>
a request with appropriate list_id, but non-matching name.	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="445"> <message> <![CDATA[The request was rejected. The entered "name" parameter is invalid.]]> </message> </response>

Media Object

The Mogreet platform has fully featured content ingestion, transcoding and serving features to make using audio, images and video in your mobile apps, web and messaging simple and painless. You never need to know what content container, codec, image size or frame rate is appropriate for a given media device – the Mogreet platform does that for you. You simply ingest a piece of content via an API call, and get back a content id and a URL that resolves to the content. You can use that URL (called a 'smart URL') wherever you need to access the content in a mobile message, application, or on the mobile web. When a remote mobile device hits the URL, the Mogreet platform reads the device type and profile information, and delivers the optimal version of the content for that device using the most appropriate delivery mechanism.

There are API calls to list all the content you have ingested, to retrieve the Smart URL for a given content id, and to destroy a piece of content hosted in the system.

As usual the easiest way to test the API calls is with curl in a terminal shell. For example, in the command below, if you replace the client id with your own client id and the token with your own token and enter the command in a terminal or shell window:

```
curl "https://api.mogreet.com/cm/media.list?client_id=123&token=a5d4a59"
```

You will receive an immediate response back as XML.

You can easily test the media upload with curl as well, using the –F flag to send a file as multipart/form-data:

```
curl -F "file=@myvideo.mp4" \
"https://api.mogreet.com/cm/media.upload?client_id=849&token=e15bfeab1cb0&type=video&n
ame=VideoPostFromCurl"
```

Remember to replace the "file=" with the name and location of your own file, and to substitute your own client_id and token.

media.list

The media list API call takes your client_id and token, and returns an XML formatted response detailing all the items of content that you have ingested, the associated content ids, and the Smart URL used to access the content on a mobile device.

Request:

https://api.mogreet.com/cm/media.list

With the following parameters:

Name	Description

client_id	Your client id.
token	Your token.
format	Optional. When set to 'json' the response will be in JSON format rather than XML.

A successful call will return an HTTP 200 with XML response, having the following elements for each media item in the media list:

Element Name	Description
content_id	A unique integer describing that content master in the Mogreet system. Can be used in other media API calls to modify or delete the content item.
smart_url	The URL where this content can be accessed on all devices
name	The name of the content item given at the time the content was ingested
type	One of: Audio, Video, Image
filename	The filename of the file ingested

Response examples:

Request	Response
https://api.mogreet.com/cm/media.list?client_id=1234&token=a487b7fba7180573cccc8958e45d4a60	<?xml version="1.0" encoding="UTF-8"?> <response code="1" status="success"> <message>media list retrieved</message> <media_list> <media> <content_id>149286</content_id> <smart_url>http://m.mogreet.com/oc/1ombg6q</smart_url> <name>RetroDemoVideo</name> <type>video</type> <filename>5153-myvideo.mp4 </filename> </media> </media_list> </response>
a media.list call returning a JSON result	{ "response": { "code": "1", "status": "success",

	"message": "media list retrieved",
	"media_list":
	[
	{"name": "bzola",
	"type": "image",
	"content_id": "160667",
	"smart_url":
	"http://m.mogreet.com/oc/iunjiw6zs",
	"filename": "7605-bzola.jpg"
	},
	{
	"name": "moTest: audio",
	"type": "audio",
	"content_id": "159632",
	"smart_url":
	"http://m.mogreet.com/oc/iw8enwq60",
	"filename": "34-barsanti.mp3"
	},
	]
	}
	}

media.upload

The media upload API call ingests, transcodes, and hosts content for you. You can use the HTTP POST method using multipart/form data and include the content to be ingested as (explain) in a 'file' field. If you prefer, and the content file is available publicly via HTTP, you can specify a URL in the API call, and the Mogreet platform will retrieve the file from that location, transcode, store and host the data.

In either case, the API call returns to you a 'smart URL' which identifies the content you submitted. When you use that smart URL in a mobile message, social application, mobile phone app, web or mobile web page, etc, the Mogreet platform detects the device type which is requesting the content, selects the most appropriate version of the content for that device, and delivers the correct data from a global CDN. All you need to do is ingest the content via the media.upload API, the Mogreet platform does all the transcoding, hosting and serving for you transparently.

media types supported

You can upload Audio, Video, and Images into the platform. When POST Upload, the mime types are used to validate the media prior to upload. The following mime types are accepted:

Video	Audio	Image
video/quicktime	audio/basic	image/jpeg
video/x-msvideo	audio/x-au	

video/x-ms-wmv	audio/mpeg	more coming soon!
video/avi	audio/x-mpeg	
video/mp4	audio/mp3	
video/mp4v-es	audio/x-mp3	
video/mpeg	audio/mpeg3	
video/x-m4v	audio/x-mpeg3	
video/3gpp	audio/mpg	
application/x-3gp	audio/x-mpg	
	audio/x-mpegaudio	
	audio/x-wav	
	audio/mp4	
	audio/x-mp4	
	audio/mp4a-latm	
	audio/wav	
	audio/mp4a-latm	
	audio/mp4a-latm	
	audio/x-ms-wmv	
	audio/x-ms-wma	

When you ingest content with a direct post, the data arrives as an octet-stream, and cannot be checked for supported media until we attempt transcoding. At that point, we support 99% or better of the LibAVCodecs found here:

http://en.wikipedia.org/wiki/Libavcodec

Failures during this phase of the upload can be detected with callbacks, see below.

callbacks

You can also specify an optional callback URL in the API call. When you specify a callback URL, the status of the upload, ingest and transcode will be passed back to that URL when the Mogreet platform completes the work. This is useful for confirming that the content is ready for use, and for catching failures and errors in transcoding that might be caused by a corrupt file, unsupported type, or other unknown problem.

Request:

https://api.mogreet.com/cm/media.upload

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
type	the media type of the content file, one of: audio, image, video The API uses this to pre-validate the file extension prior to ingesting it. The observed filetype of the file (.png, .jpg, .mov, etc) should be appropriate for the type described in this field. If there is a mismatch, an error will be returned (see below).
name	A descriptive name to be used in the media.list and other areas to describe the media content. This can be different than the filename.
file (optional)	The actual data file payload to ingest: the image file, audio file or video file. This field is used when POSTing multi-part form data. It can be omitted if you are using the URL ingestion method.
url (optional)	When you wish to ingest a media file which is already hosted somewhere on the internet, you can use this option instead of the POST multi-part form method. The URL should identify a file hosted at a fully qualified domain, and be publically available without login or other security.
callback_url (optional)	If you specify a callback_url, a message indicating success or failure will be delivered to the location specified, as an XML document. See below for examples.

A successful call will return an HTTP 200 with XML response as detailed below. Example call:

```
https://api.mogreet.com/cm/media.upload?client_id=849&token=e15b9a70b18c38fa275496836eab1cb0&url=http://www.mysite.com/video/nice_car.mp4&type=video&name=VideoFromUrl
```

Name	Description
complete response	<?xml version="1.0" encoding="UTF-8"?> <response code="1" status="success"> <message>media correctly uploaded</message> <media> <name>lancia_037</name> <content_id>154570</content_id> <smart_url>http://m.mogreet.com/oc/iw9l2mf59</smart_url> </media> </response>
name	The name of the content item given at the time the content was ingested

smart_url	a URL which will resolve to the correct version of the content for any media device. You can test a smart URL by pasting it into a web browser. http://m.mogreet.com/oc/iw9l2mf59

Here is an example call using the multi-part/form encoding, using curl. Curl is a great way to test the upload command while in development.

```
curl -F "file=@myvideo.mp4"
"https://api.mogreet.com/cm/media.upload?client_id=849&token=e15b9a70b18c38fa275496836
eab1cb0&type=video&name=VideoPostFromCurl"
```

Here is an example API command, using curl, with the optional callback_url specified. That URL will receive an XML document describing the status and result when the upload and transcode is complete.

```
curl -F "file=@myvideo.mp4"
"https://api.mogreet.com/cm/media.upload?client_id=849&token=e15b9a70b18c38fa275496836
eab1cb0&type=video&name=VideoCallback&callback_url=http://labs.mogreet.com/api_callbac
k"
```

Here are some examples of errors you might encounter:

Request	Response
an improper filetype	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="462"> <message> <![CDATA[Your video is in an unsupported format.]]> </message> </response>
the file was larger than the upload size limit of 150 MB.	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="461"> <message> <![CDATA[Maximum file size exceeded. Please keep the file under 150MB.]]> </message> </response>
invalid media name too long!)	<?xml version="1.0" encoding="UTF-8"?> <response status="error" code="463"> <message> <![CDATA[Name is too long (maximum is 50 characters)]]> </message> </response>

media.destroy

The media.destroy API command takes a content_id, and then flags it for deletion. Content deletion happens daily, at which time all versions will be purged from the CDN and hosting services, and the records of the versions will be deleted. Up until deletion occurs the content will still be available via the smart URL. After deletion, any request to that smart URL will receive a special 'not available' content item.

Request:

https://api.mogreet.com/cm/media.destroy

With the following parameters:

Name	Description
client_id	Your client id.
token	Your token.
content_id	The content id of the content item you want removed from the system. After deletion that content item will no longer be available. There is no undo.

Other Response examples:

Request	Response
A successful media.destroy API call	<?xml version="1.0" encoding="UTF-8"?> <response code="1" status="success"> <message>media correctly deleted</message> <media> <name>lancia_037</name> <content_id>154570</content_id> <hash>iw9l2mf59</hash> </media> </response>

Meta Store Methods

The MetaStore APIs are similar to the Stats APIs. These APIs are still RESTful web calls, with a reduced level of required security, made with HTTP rather than HTTPS (although HTTPS will still work), and they do not require a client_id and token.

Store

The purpose of the Store API method is to record arbitrary key/value pairs of data with a known MSISDN (cell number).

The response is provided in the JSON format, to make it easy to consume and parse in a client side browser. A typical use case would be where a web form collects additional information about a mobile subscriber, such as shoe size, eye color, etc. That data could be stored with the subscriber MSISDN, and then at a later date, a list of recipients could be filtered to display and use only those that meet a specific requirement.

These API calls have a destination URL that contains the identifier 'store' in the path, like this:

http://api.mogreet.com/store/userdata?phone=2061112223333 ....

With the following parameters:

Name	Description
phone	The 'phone' field is required.
key/value pairs	any number of key/value pairs can be included in a single API call. You may also make several different API calls to associate different groups of key/value pairs with the same MSISDN.

The result set returned is in JSON format. For the following API call:

http://api.mogreet.com/store/userdata?phone=2061112223333&eyes=blue&gender=male

here is a representative JSON result confirming that the data is set:

```
{"attributes":
       {"eyes":"blue","gender":"male"},
       "tags":{},
```

```
"phone":"2061112223333",
"status":"success"
}
```


There is no API to retrieve the key/value pairs for a given MSISDN. The metastore data is used only to filter lists in the Mogreet Campaign Manager.

Appendix A: Media Types and Formats

The MoMS allows content to be ingested and transcoded for delivery on the fly. Images, Audio files, and video files are all supported content types. The developer includes a URL pointing to the publicly accessible location of the content file in a transaction.send or blast.send API call to initiate the automatic on-demand ingestion process. Only the HTTP and HTTPS protocols are supported for ingestion.

In general, the content needs to have a valid http content type (applies when using the content_url parameter) or a valid file extension. Here are the supported content types and file extensions for the media we support.

```
##
# Maximum content length allowed.
MAX_CONTENT_LENGTH = 104_857_600 # Set to 100 MBs.

##
# Supported video content-types.
SUPPORTED_VIDEO_TYPES = [
  'video/mpeg',
  'video/quicktime',
  'video/mp4',
  'video/x-msvideo',
  'video/x-ms-wmv',
  'video/3gpp',
  'video/3gpp2',
  'video/x-m4v',
  'video/x-flv',
]
##
# Supported video extensions.
SUPPORTED_VIDEO_EXTENSIONS = [
  '.mpeg',
  '.mpg',
  '.mov',
  '.qt',
  '.mp4',
  '.m4v',
  '.avi',
  '.wmv',
  '.3gp',
  '.3gpp',
  '.3g2',
  '.flv',
]
##
# Supported image content-types.
SUPPORTED_IMAGE_TYPES = [
  'image/gif',
  'image/jpeg',
  'image/png',
  'image/tiff',
]
##
# Supported image extensions.
SUPPORTED_IMAGE_EXTENSIONS = [
  '.gif',
  '.jpg',
  '.jpeg',
  '.png',
```

```
  '.tif',
]
##
# Supported audio content-types.
SUPPORTED_AUDIO_TYPES = [
  'audio/amr',
  'audio/basic',
  'audio/mid',
  'audio/mpeg',
  'audio/x-aiff',
  'audio/x-mpegurl',
  'audio/x-wav',
]
##
# Supported audio extensions.
SUPPORTED_AUDIO_EXTENSIONS = [
  '.aif',
  '.aiff',
  '.au',
  '.m3u',
  '.mid',
  '.mp3',
  '.mp4',
  '.wav',
]
```

Appendix B: Supported Mobile Carriers

The MoMS has connectivity to and from almost all the carriers in the US. There may be a few rural small carriers which have no connectivity via SMS or MMS. Here is a table of carrier connectivity and whether they support SMS and MMS via the MoMS.

ACS/Alaska	SMS ONLY
Alltel	MMS AND SMS
AT&T	MMS AND SMS
Bluegrass Cellular	SMS ONLY
Boost Mobile	SMS ONLY
Cello	SMS ONLY
Cellular One of N.E. Pennsylvania	SMS ONLY
Cellular South	SMS ONLY
Chat Mobility-NW Missouri Cellular	SMS ONLY
Cincinnati Bell	SMS ONLY
Cricket	MMS AND SMS
Cross Wireless	SMS ONLY
Duet IP	SMS ONLY
East Kentucky Wireless	SMS ONLY
ECIT	SMS ONLY
Elegant Mobile	SMS ONLY
Epic Touch	SMS ONLY
GCI	SMS ONLY
Golden State Cellular	SMS ONLY
Illinois Valley Cellular	SMS ONLY
Immix/Keystone Wireless	SMS ONLY
Inland Cellular	SMS ONLY
IWireless	MMS AND SMS
Metro PCS	SMS ONLY
Mobi PCS	SMS ONLY
Mosaic Telecom	SMS ONLY
MTPCS Cellular One	SMS ONLY
Nex-Tech Wireless	SMS ONLY
Nextel	SMS ONLY
nTelos	SMS ONLY
Panhandle Wireless	SMS ONLY
Peoples Wireless	SMS ONLY

Pioneer Cellular	SMS ONLY
Plateau Wireless	SMS ONLY
Revol Wireless	SMS ONLY
RINA (10 Carriers)	SMS ONLY
Rural Cellular Corp	SMS ONLY
Simmetry Wireless	SMS ONLY
Sprint	MMS AND SMS
T-Mobile	MMS AND SMS
Thumb Cellular	SMS ONLY
Tracfone (AT&T)	SMS ONLY
Union telephone	SMS ONLY
United Wireless	SMS ONLY
US Cellular	MMS AND SMS
Verizon	MMS AND SMS