Basics

A Lattigo Enterprise subscription is required to generate keys to make use of the API.

All API access is through HTTPS and is accessed through the root service URL: https://api.lattigo.com

Data is sent to the API through the query string for GET requests, and as JSON body data for all other types of requests.

Authentication

A key and secret pair are needed to access the API. Up to five key pairs can be created at one time for each account, and they can be generated and managed in your customer portal at https://portal.lattigo.com/api_keys.

Authentication is handled using HTTP Basic Authentication. The username used for authentication is the key_name of an API key in your account, and the password is the api_key.

Most languages and tooling support HTTP Basic Authentication. Should you need to set the header yourself, you must assemble it. The key_name and api_key pair are first concatenated using a single colon and then base64 encoded:

```
# Using the GNU base64 utility
/> printf <your key_name>:<your api_key> | base64
#  Base64 encoded output. Your output will be different
U29tZUFQSUtleU5hbWU6U29tZUFQSUtleVZhbHVl
```

The base64 encoded value is then used in the Authorization header:

```
Authorization: Basic U29tZUFQSUtleU5hbWU6U29tZUFQSUtleVZhbHVl
```

The curl utility lets you set the header via the command line, and does the encoding automatically. The examples all use curl.

```
# Pass the key_name and api_key as colon separated values, using the -u switch
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices?details=t
```

Return Codes

Most API endpoints accept one device, or an array of devices. The return codes map to an appropriate RESTful response. Endpoints may return additional information in the response body on error, and success.

- 200 The call succeeded
- 400 Missing, or incorrect parameters. The call failed to provide the necessary parameters.
- 401 Authentication or Authorization failure. Credentials are missing or incorrect.
- 403 Authorization failure. Attempted access to a resource you do not own.
- 404 The resource could not be found. Not a valid endpoint, possibly a typo.
- 405 Method not allowed. The endpoint does not accept this method, perhaps a different method was intended.
- 429 Too many requests. Lattigo may limit calls to endpoints to prevent abuse and maintain service for all API users.

Pagination

Endpoints that return a lot of data accept extra query string parameters for pagination. These parameters are:

- page: The page to retrieve.
- per_page: Number of records per page.
- offset: Offset of the record to start from.

The response will include pagination headers, e.g:

```
X-Total: 42
X-Total-Pages: 5
X-Page: 3
X-Per-Page: 10
X-Next-Page: 4
X-Prev-Page: 2
X-Offset: 10
```

Pagination indices start at '1'. The first page by default is '1' if you do not specify differently.

Time Format

All endpoints return time in an ISO 8601 format expressed as UTC. All input times are expected to be in ISO 8601 format. The API tolerates some non-standard formats. Times not in ISO 8601 format, or lacking a timezone, will be assumed to

be expressed in UTC.

Input time formats not defined by ISO 8601 are not guaranteed to return the correct results.

Rate Limiting

Lattigo may limit calls to endpoints to prevent abuse and maintain service for all API users. Reduce the rate of calls to any endpoint that returns a 429 HTTP error code. If you have futher questions, please contact support.

Single Device Endpoints

v2/devices

Returns an array of 'device_names,' identifying the devices associated with the account. An empty array is returned if there are no devices found. A 'device_name' is a required parameter for API endpoints below v2/devices

Verb: GET

Parameters:

- details: optional, set to true to retrieve more detailed information about all devices
- state: optional, filter the device list to only devices in this state

Optional pagination parameters:

- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices'
# outputs JSON data
[ <device a name>, <device b name>, <device c name>, ...]
```

Example with 'details' parameter

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices?details=t
# outputs an array of device objects
```

```
[ { <device a>}, {<device b>}, ... ]
```

'device' object fields:

```
:id
:device_name
:ip_address
:ipv6
:note
:state
:mdn
:meid
:imsi
:imei
:iccid
:esn
:msisdn
:min
:connected
:last_connection_date
:cost_center
:last_activation_date
:latitude
:longitude
:address
:bytes_used
:sms_used
:note1
:note2
:note3
```

v2/devices/:device_name

Get a a particular device object, by device name.

Verb: GET

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type
- id_type: optional, id type if different from device_name, e.g. iccid

Example:

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/<a valid d
```

```
# outputs a device object
{ <device a> }
```

v2/devices/:device_name/get_sms

Returns an array of SMS MT status objects.

This is only supported for VDF devices.

Verb: GET

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type
- start_date: optional, ISO 8601 formatted date, inclusive. The beginning of search window. The API accepts ISO 8601 format, but only the year, month, and date are used. Any time component will be ignored. It may be used to convert the passed value to UTC.
- end_date: optional, ISO 8601 formatted date, inclusive. The end of the search window. The API accepts ISO 8601 format, but only the year, month, and date are used. Any time component will not be passed. It may be used to convert the passed value to UTC.
- direction: optional. A string, one of 'all,' 'received,' 'sent'
- id_type: optional, id type if different from device_name, e.g. iccid

Example

```
/> curl -X GET -u '<your key_name>:<your api_key>' https://api.lattigo.com/v2/devices/<a
```

Return

```
[ { < SMS MT Object > } , ... ]
```

SMS MT Object fields:

```
:event_id
:event_status
:event_timestamp
:event_type
:imsi
:message_body
:msisdn
:report_timestamp
:failure_reason
```

v2/devices/:device_name/calamp_messages

Returns an array of 'calamp_message' objects for the device_name, for the date range specified. An empty array is returned if there are no messages found.

Messages are not stored indefinitely. Messages older than 3 months might not be available.

Verb: GET

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type
- start_date: ISO 8601 formatted date, inclusive.
- end_date: ISO 8601 formatted date, inclusive.

Optional parameters:

- event_code: only return messages with the given event code
- id_type: optional, id type if different from device_name, e.g. iccid
- page: page to retrieve
- per_page: how many records in a page
- offset: the offset to start from

Example:

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/36524/cala
# outputs JSON data
[{ <message> }, { <message> }, ...]
```

Message fields:

```
:message_id
:mobile_id
:latitude
:longitude
:altitude
:speed
:heading
:satellites
:fix_status
:event_index
:event_code
:gps_hdop
:communication_state
:rssi
:network_id
:inputs
```

```
:unit_status
:sequence_number
:report_generated_at
:created_at
:time_of_fix
:accumulator_count
:accumulator0
:accumulator1
:accumulator2
:accumulator3
:accumulator4
:accumulator5
:accumulator6
:accumulator7
:accumulator8
:accumulator9
:accumulator10
:accumulator11
:accumulator12
:accumulator13
:accumulator14
:accumulator15
```

v2/devices/:device_name/last_calamp_message

Returns the latest 'message' object for the device name, for the date range specified. An empty array is returned if there are no message for the device

Verb: GET

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type
- id_type: optional, id type if different from device_name, e.g. iccid

Example:

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/36524/last
# {"message_id":281,"mobile_id":"2424476102","latitude":"731.09", ...
```

'message' fields are the same as above

v2/devices/:device_name/note

Create or modify a note field on a device. Returns a simple JSON object, if successful

Verb: PUT

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type
- field_name: must be one of 'note', 'note1', 'note2', 'note3'
- note: a string, containing the note to store. Length is restricted, and notes longer than 100 characters may be truncated arbitrarily by the platform
- id_type: optional, id type if different from device_name, e.g. iccid

Example:

```
/> curl -X POST -u '<your key_name>:<your api_key>' -d 'field_name=note' -d 'note=Meaning
```

Example using JSON:

```
/> curl -X POST -H "Content-Type: application/json" -u '<your key_name>:<your api_key>' -
```

Return

```
{ "<the device_name>" : "Success" }
```

v2/devices/:device_name/device_reports

Get gevice reports for the named device. Returns a simple JSON object, if successful

Verb: GET

Parameters:

- device_name: as returned by 'v2/devices' or indicated by id_type

Optional parameters:

- start_date: a date or time, defaults to one month ago (2023-02-23)
- end_date: a date or time, defaults to now
- limit: an integer, the maximum number of results to return, defaults to 1
- id_type: optional, id type if different from device_name, e.g. iccid
- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -X GET -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/365
```

Example using JSON:

```
/> curl -X GET -H "Content-Type: application/json" -u '<your key_name>:<your api_key>' 'ht
```

Message fields:

```
:id
:latitude
:longitude
:speed
:heading
:altitude
:gps_quality
:unit_status
:time_of_fix
:battery
:voltage
:event_code
:temperature
:created_at
:type
:fuel_level
:distance
:rssi
```

Return

```
# An array of device report hashes
[ {device_report_1}, {device_report_2} ...]
```

v2/devices/:device_name/daily_data_usages

Get Daily Data Usages for the named device. Returns a simple JSON object, if successful

Verb: GET

Optional parameters:

- start_date: a date or time, defaults to one month ago (2023-02-23)
- end_date: a date or time, defaults to now
- limit: an integer, the maximum number of results to return, defaults to 1
- id_type: optional, id type if different from device_name, e.g. iccid

- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -X GET -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/365
```

Example using JSON:

```
/> curl -X GET -H "Content-Type: application/json" -u '<your key_name>:<your api_key>' 'ht
```

Message fields:

```
:id
:date_for
:bytes_used
:sms_used
```

Return

```
# An array of data usage hashes report hashes
[ {daily_data_usage_1}, {daily_data_usage_1} ...]
```

v2/devices/:device_name/monthly_data_usages

Get Monthly Data Usages for the named device. Returns a simple JSON object, if successful

Verb: GET

Optional parameters:

- start_date: a date or time, defaults to one month ago (2023-02-23)
- end_date: a date or time, defaults to now
- limit: an integer, the maximum number of results to return, defaults to 1
- id_type: optional, id type if different from device_name, e.g. iccid
- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -X GET -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/365
```

Example using JSON:

```
/> curl -X GET -H "Content-Type: application/json" -u '<your key_name>:<your api_key>' 'ht
```

Message fields:

```
:id
:date_for
:bytes_used
:sms_used
```

Return

```
# An array of data usage hashes report hashes
[ {monthly_data_usage_1}, {monthly_data_usage_1} ...]
```

v2/devices/:device_name/cellular_locate

Request location triangulation from carrier if supported, charges apply per call

Verb: POST

Optional parameters:

- id_type: optional, id type if different from device_name, e.g. iccid

Example:

```
/> curl -X POST -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/devices/3
```

Example using JSON:

```
/> curl -X POST -H "Content-Type: application/json" -u '<your key_name>:<your api_key>' 'h
```

Return

```
{ "<the device_name>": "<carrier_response>" }
```

Device Operations Endpoints

v2/device_operations/activation_request

Request activation for specified devices.

Request Body Spec

Parameter Name	Value Type	Required	Description
rate_plan_name	string	yes	Name of the customer's billing plan to activate under (Available options determined by contract. Contact PWS support to confirm available choices or spelling if needed.)
device_sku	string	no	Can be used instead of an IMEI for customers who only activate on a single hardware model. Note SKUs are not publicly available so contact PWS support if interested.
region_code	string	no	For rate plans with geographically determined APNs, this allows customers to select the device's home region. Within the United States, the value is a 5-digit zipcode.
imei	string	yes	Valid IMEI number
iccid	string	yes	Valid ICCID number
note	string, up to 250 standard alphabet characters and limited symbols	no	Note fields are for customer use and will be stored with the device records. They can be modifed later but will never be modified by PWS.
note1	string, up to 250 standard alphabet characters and limited symbols	no	Additional note field.
note2	string, up to 250 standard alphabet characters and limited symbols	no	Additional note field.
note3	string, up to 250 standard alphabet characters and limited symbols	no	Additional note field.

Below is an example of the body you could submit with an activation request

```
{
  "rate_plan_name" : "ACME Co 5mb Rate Plan",
  "activation_requests" : [
    {
      "imei" : "111111111111111",
      "iccid" : "89222222222222222222"
    },
    {
      "imei" : "333333333333333",
      "iccid" : "89444444444444444444",
      "note"  : "Notes are stored with the device record and can be used for sorting, filt
    },
    {
```

```
      "imei" : "555555555555555",
      "iccid" : "89666666666666666666",
      "note3" : "Note fields do not need to be filled in any particular order."
    }
  ]
}
```

Verb: POST

Example using curl with the above request body as properly JSON formatted

```
/> curl -X POST -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d "{ \"rate
```

Another example, requesting a single device activation on a different plan "ACME Co 10MB Plan"

```
/> curl -X POST -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d "{ \"rate
```

Response Body Spec

HTTP return status codes will either be

- HTTP 200 - Ok (indicating the request was accepted and will be worked on, not that the request has been completed)
- HTTP 422 - Unprocessable Entity
- HTTP 500 - Internal Server Error

Parameter Name	Value Type	Description
processed	array of hashes	Each entry is a record of a successfully processed request which is being activated
unprocessed	array	Each entry is a record of an unsuccessful request which is not being activated, a
device_index	integer	Index of the device in submitted request which did not get processed (starting from 0)
error	JSON encoded hash of arrays	Each hash key references something that had an issue, each entry in the subarrays details what the issue was
input	hash	A copy of the data that failed activation submission

Example POST response body. This response would have HTTP status code: 422 "Unprocessable Entity"

```
{
  "processed": [
    {
```

```
      "iccid": "89222222222222222222",
      "imei": "111111111111111",
      "note": "",
      "note1": "",
      "note2": "",
      "note3": "",
      "request_type": "activate"
    },
    {
      "iccid": "89666666666666666666",
      "imei": "555555555555555",
      "note": "",
      "note1": "",
      "note2": "",
      "note3": "Note fields do not need to be filled in any particular order.",
      "request_type": "activate"
    }
  ],
  "unprocessed": [
    {
      "device_index": 1,
      "error": "{\"iccid\":[\"Error messages would be here about an issue with the ICCID\"
      "input": {
          "iccid": "89444444444444444444",
          "imei": "333333333333333",
          "note": "Notes are stored with the device record and can be used for sorting, fi
          "note1": "",
          "note2": "",
          "note3": ""
      }
    }
  ]
}
```


v2/device_operations/suspend_no_billing_request

Request suspension without billing for specified devices. This does not directly suspend the requested ICCID/ IMEI pairs, but queues a request. The HTTP response codes indicate success or failure to queue the request. Suspension may take some time.

Parameters:

- suspension_requests: hash comprising an array of IMEI & ICCID tuples named 'devices', structured as shown. Each suspend_no_billing_request object must include the IMEI and the ICCID.

```
{
  "suspension_requests" :
    [
      { "imei" : "123" , "iccid" : "456" },
      { "imei" : "902902930", "iccid" : "789" }
    ]
}
```

Verb: POST

Example using JSON

```
/> curl -X POST -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d '{ "suspe
```

v2/device_operations/change_cost_center

Update the cost center associated with each device. Returns a 'change_status' object, with change status for each device_name. This replaces an earlier API endpoint used to change the cost center associated with the device.

Parameters:

- device_names: an array of device_name, as returned by v2/devices
- cost_center: A string, applied to all specified device_names

Verb: PUT

Example

```
/> curl -X PUT -u '<key_name>:<api_key>' -d 'device_names[]=123' -d 'device_names[]=456' -
```

Example using JSON

```
/> curl -X PUT -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d '{"device_
```

change_status fields:

```
{
  change_status: {
    <device_name> : <Success|Error: Device does not exist.>,
    <device_name> : <Success|Error: Device does not exist.>,
    ....
  }
```

```
}
```

v2/device_operations/change_simstate

Update the simstate for each device. Returns 'change_status' object, with change status for each device_name.

Parameters:

- device_names: an array device_names or IDs of id_type if specified.
- state: the SIM state, this will be applied to all specified device_names. Must be one of 'suspend|restore'
- id_type: optional, id type if different from device_name, e.g. iccid

Verb: PUT

Example

```
/> curl -X PUT -u '<key_name>:<api_key>' -d 'device_names[]=123' -d 'device_names[]=456' -
```

Example using JSON

```
/> curl -X PUT -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d '{"device_
```

change_status fields:

```
{
  change_status: {
    <device_name> : <Success|Error: Device does not exist.>,
    <device_name> : <Success|Error: Device does not exist.>,
    ....
  }
}
```

v2/device_operations/remove_usage_limit

Remove monthly usage limit from the specified devices.

Parameters:

- device_names: array of device_names to udpate

Verb: POST

Example

```
/> curl -X POST -u '<key_name>:<api_key>' -d 'device_names[]=123' -d 'device_names[]=456'
```

Using JSON

```
/> curl -X POST -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d '{"device
```

status fields:

```
{
  status: {
    <device_name> : <Success: Device data limit removed.|Error: Device does not exist.>,
    <device_name> : <Success: Device data limit removed.|Error: Device does not exist.>,
    ....
  }
}
```

v2/device_operations/send_sms

Queue a request to send SMS for specified devices.

Parameters:

- device_names: an array of device_names to which to send an SMS message
- sms_text: the SMS text to send

Verb: POST

Example

```
/> curl -X POST -u '<key_name>:<api_key>' -d 'device_names[]=123' -d 'device_names[]=456'
```

status fields:

```
{
  status: {
    <device_name> : <Success: Attempting SMS Send.|Error: Device does not exist, or does r
    <device_name> : <Success: Attempting SMS Send.|Error: Device does not exist, or does r
    ....
  }
}
```

v2/device_operations/set_usage_limit

Set usage limits on specified devices. Returns a 'status' object, with change status for each device_name specified.

Once the usage limit is reached, the device is suspended. The device state will not change without further intervention.

Parameters:

- device_names: array of device_names to udpate
- kb_limit: Integer, specifying the kb_limit for monthly bandwidth for all specified devices.
- action: May be 'suspend,' the action to take when the device reaches its monthly bandwidth allotment. At least one of 'suspend,' or 'email' must be specified. Suspensions will also send email upon triggering
- usernames: An array of usernames, as returned by '/v2/users'. Each user will be sent an email alert when the device reaches its limit. Invalid usernames may be silently ignored.

Verb: POST

Example

```
/> curl -X POST -u '<key_name>:<api_key>' -d 'device_names[]=123' -d 'device_names[]=456'
```

Example with JSON data

```
/> curl -X POST -H "Content-Type: application/json" -u '<key_name>:<api_key>' -d '{"device
```

status fields:

```
{
  status: {
    <device_name> : <Success: Device data limit is set.|Error: Device does not exist.>,
    <device_name> : <Success: Device data limit is set.|Error: Device does not exist.>
    ....
  }
}
```

v2/device_operations/upload_request

Request upload of IMEI(s) to carrier device database.

Request Body Spec

Parameter Name	Value Type	Required	Description
upload_email	string	yes	Carrier-registered email address of the device vendor
upload_sku	string	yes	Carrier-registered vendor sku
upload_ids	array	yes	Array of device identifiers
imei	string	yes	Valid IMEI number

Below is an example of the body you could submit with an upload request

```
{
  "upload_email" : "name@vendor_company.com",
  "upload_sku" : "XYZ000000012345",
  "upload_ids" :
    [
      { "imei" : "111111111111111" },
      { "imei" : "333333333333333" },
      { "imei" : "555555555555555" }
    ]
}
```

Verb: POST

Example using curl with the above request body as properly JSON formatted

```
/> curl -X POST -H "Content-Type: application/json" -u "<key_name>:<api_key>" -d "{ \"uplo
```

Response Body Spec

HTTP return status codes will either be

- HTTP 200 - Ok
- HTTP 422 - Unprocessable Entity
- HTTP 500 - Internal Server Error

Parameter Name	Value Type	Description
processed	array of hashes	Each entry is a record of a successfully processed request which is being activated
unprocessed	array	Each entry is a record of an unsuccessful request which is not being activated, a
device_index	integer	Index of the device in submitted request which did not get processed (starting from 0)
error	JSON encoded hash of arrays	Each hash key references something that had an issue, each entry in the subarrays details what the issue was
input	hash	A copy of the data that failed activation submission

Example POST response body. This response would have HTTP status code: 422 "Unprocessable Entity"

```
{
  "processed": [
    {
```

```
      "imei": "111111111111111",
      "sku": "XYZ0000000012345",
      "email":"name@vendor_email.com"
    },
    {
      "imei": "555555555555555",
      "sku": "XYZ0000000012345",
      "email":"name@vendor_email.com"
    }
  ],
  "unprocessed": [
    {
      "device_index": 1,
      "error": "{\"imei\":[\"Error messages would be here about an issue with the IMEI\"]}",
      "input": {
        "imei": "333333333333333"
      }
    }
  ]
}
```

v2/device_operations/calamp_events

Returns an array of 'calamp_message' objects with a specific event code, across all devices, for the date range specified. An empty array is returned if there are no messages found.

Messages are not stored indefinitely. Messages older than 3 months might not be available.

Verb: GET

Parameters:

- event_code: message event code to search for
- start_date: ISO 8601 formatted date, inclusive.
- end_date: ISO 8601 formatted date, inclusive.

Optional parameters:

- state: filter for devices in a given state
- page: page to retrieve
- per_page: how many records in a page
- offset: the offset to start from

Example:

```
/> curl -u '<your key_name>:<your api_key>' 'https://api.lattigo.com/v2/device_operations/
# outputs JSON data
[{ <message> }, { <message> }, ...]
```

Message fields:

```
:message_id
:mobile_id
:latitude
:longitude
:altitude
:speed
:heading
:satellites
:fix_status
:event_index
:event_code
:gps_hdop
:communication_state
:rssi
:network_id
:inputs
:unit_status
:sequence_number
:report_generated_at
:created_at
:time_of_fix
:accumulator_count
:accumulator0
:accumulator1
:accumulator2
:accumulator3
:accumulator4
:accumulator5
:accumulator6
:accumulator7
:accumulator8
:accumulator9
:accumulator10
:accumulator11
:accumulator12
:accumulator13
:accumulator14
```

```
:accumulator15
```

Rate Plan Endpoints

v2/rate_plans

Get a list of rate plans associated with the account.

Verb: GET

Optional pagination parameters:

- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -u '<your key_name>:<your api_key>' https://api.lattigo.com/v2/rate_plans
[<rate_plan 1>, <rate_plan 2>, ... ]
```

Rate plan object fields:

```
:name
:days_in_term
:activation_fee
:termination_fee
:monthly_data_plan_cost
:monthly_data_allowance_bytes
:overage_data_unit_bytes
:overage_data_unit_cost
:monthly_sms_plan_cost
:monthly_sms_allowance
:overage_sms_unit
:overage_sms_unit_cost
```

User Endpoints

v2/users

Get a list of usernames associated with the account.

Verb: GET

Optional pagination parameters:

- page: your current page
- per_page: how many to record in a page
- offset: the offset to start from

Example:

```
/> curl -u '<your key_name>:<your api_key>' https://api.lattigo.com/v2/users
[<username 1>, <username 2>, ... ]
```