Using the REST API

Mobile Backend Services (MBS) provides a REST API accessible from any networked client device for creating, querying, updating, and deleting MBS objects.

Each MBS object supports a set of methods, which are documented in the API reference.

REST API basics

Each of the REST API methods has its own URL and HTTP method (GET, POST, PUT, or DELETE).

To make an API call, you make an HTTP request. Method parameters are passed in the URL query string or in the message body, depending on the HTTP method.

For GET and DELETE requests, send the parameters in the URL itself as part of the URL query string. For example:

```
https://api.cloud.appcelerator.com/v1/checkins/show.json?key=
<YOUR APP APP KEY>&checkin_id=4d8bc645d0afbe0363000013
```

For POST and PUT requests, you send an HTTP request with the `multipart/form-data` media type, where each parameter is sent as a separate form field.

API responses are returned as JSON objects. In most cases, the response JSON includes two objects:

- `meta` - An object containing response metadata, including the response status code and error message, if any.
- `response` - An object containing the actual data for the request. The `response` object is omitted for some requests, such as delete requests, that return no data.

Authentication

All API calls must contain a valid App Key or 2-Legged OAuth signature and request header for the MBS server to process and respond to them. See the authentication page for more information.

User sessions and cookies

To create a user and perform actions which require a logged-in user, the `session_id` cookie must be saved and reused with each API call.

To get a session ID, use the `users/login.json` method to login into the application. If the API call is successful, the `session_id` field is returned in the `meta` object of the response. For example:

```
curl -F "login=admin" -F "password=admin"
"https://api.cloud.appcelerator.com/v1/users/login.json?key=<API_KEY>"
{
  "meta": {
    "code": 200,
    "status": "ok",
    "method_name": "loginUser",
    "session_id": "sjuvQqSEYTD3DekMCrUHcCTf7GU"
  },
  "response": {
    "users": [
      {
        "id": "526edc0d294e712499000004",
        "created_at": "2013-10-28T21:50:05+0000",
        "updated_at": "2013-12-05T01:02:32+0000",
        "external_accounts": [
        ],
        "confirmed_at": "2013-10-28T21:50:05+0000",
        "username": "admin",
        "role": "",
        "admin": "true",
      }
    ]
  }
}
```

Pass the `session_id` value to the `_session_id` parameter in the URL, for example:

```
https://api.cloud.appcelerator.com/v1/reviews/create.json?key=<API_KEY>&_session_id=<S
ESSION_ID>
```

With the cURL command, use the `-b` and `-c` options to read and write cookies to store your session ID. Many of the REST examples in the documentation use these options.

User login sessions expire after they have been unused for three months. If the application saves and uses a persistent reference to the user login session, and the user session expires, any MBS call that requires a user login will return a 404 error. Your application needs to handle an invalid user session error, such as prompting the user to log in.

Testing with cURL and wget

cURL and wget are both excellent tools for quickly testing MBS API calls from the command line. Using these commands can help you determine what calls to make and show you the JSON output that your app receives. cURL is included with OS X, and can easily be used from the Terminal application:

```
$ curl --verbose -b cookies.txt -c cookies.txt
https://api.cloud.appcelerator.com/v1/places/search.json?key=vvCNPSh1cd0Gb5A6tWZAIC4Mn
gO95mGs
```

Use the `-b cookies.txt` and `-c cookies.txt` options to save and reuse the `_session_id` cookie sent from the MBS server. The `--verbose` option is useful for seeing all of the HTTP header and cookie information sent and received by the MBS server.

```
```

Expand

source

```
$ curl --verbose -b cookies.txt -c cookies.txt -F "login=mike@appcelerator.com" -F
"password=food"
https://api.cloud.appcelerator.com/v1/users/login.json?key=vjCQ6KRqplmkektlpbEjiDQ2nYR
eubkP
* About to connect() to host (#0)
*   Trying 1.2.3.4... connected
* Connected to host (1.2.3.4) (#0)
> POST /v1/users/login.json?key=vjCQ6KRqplmkektlpbEjiDQ2nYReubkP HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8l
zlib/1.2.3
> Host: <%= "#{@api_url}"%>
> Accept: */*
> Content-Length: 259
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=----------------------------e6e42e31228c
>
* Done waiting for 100-continue
< HTTP/1.1 200 OK
< X-Ua-Compatible: IE=Edge
< Access-Control-Allow-Headers: x-requested-with
< Etag: "9eeb8ecc4fd905ab6b340c290f24ea0f"
< Access-Control-Allow-Methods: POST, GET, PUT, DELETE, OPTIONS
< Connection: Keep-Alive
< Content-Type: application/json; charset=utf-8
< Date: Mon, 16 May 2011 04:56:58 GMT
< Server: WEBrick/1.3.1 (Ruby/1.8.7/2009-06-12)
< X-Runtime: 1.681994
< Content-Length: 480
< Cache-Control: max-age=0, private, must-revalidate
< Access-Control-Allow-Credentials: true
* Added cookie _session_id="4dd0ae9ad0afbe183300001c" for domain localhost, path /,
expire 0
< Set-Cookie: _session_id=4dd0ae9ad0afbe183300001c; path=/; HttpOnly
<
{
  "meta": {
    "stat":"ok",
    "code":200,
    "method":"loginUser",
    "session_id":"4dd0ae9ad0afbe183300001c"
  },
  "response": {
    "users": [
      {
        "id":"4dc6334fd0afbe3700000001",
        "first_name":"Mike",
        "last_name":"Goff",
        "created_at":"2011-05-08T06:08:15+0000",
        "updated_at":"2011-05-16T04:56:58+0000",
        "facebook_authorized":false,
        "email":"mike@appcelerator.com"
      }
    ]
  }
}
```

Uploading photos

The `create` (POST) and `update` (PUT) methods for many objects such as `Users`, `Checkins`, and `Photos` take an optional `photo` or `file` par ameter to send a photo. The binary data must be sent in a form with Content-Type `multipart/form-data`, and the content type of the `photo` o r `file` must be`image/jpeg`, `image/png`, or `image/gif`.

When using cURL, uploading a photo can be done easily by using `@` in front of the filename, such as `@photo.jpg`, to specify that the file should be attached.

```
$ curl --verbose -b cookies.txt -c cookies.txt -F "photo=@photo.jpg" -F "message=At
the beach"
https://api.cloud.appcelerator.com/v1/statuses/create.json?key=<API_KEY>
```

Object IDs

All returned data objects contain unique IDs which are 24-digit hexadecimal strings. These IDs may be used to efficiently return data for a single object:

Expand source

```
GET
https://api.cloud.appcelerator.com/v1/places/show/**4d6f13e96f70953608000012**.json?ke
y=<API_KEY>

{
  "meta": {
    "stat":"ok",
    "code":200,
    "method":"showPlace"
  },
  "response": {
    "places": [
      {
          "id":"4d6f13e96f70953608000012",
          "name":"Maya Restaurant",
          "created_at":"2011-03-03T04:07:05+0000",
          "updated_at":"2011-03-03T04:07:05+0000",
          "address":"303 2nd Street",
          "city":"San Francisco",
          "state":"CA",
          "country":"United States",
          "phone":"(415) 543-2928",
          "lat":37.784732,
          "lng":-122.395441
      }
    ]
  }
}
```

Response paging

⚠ For query operations, the `page` and `per_page` paging mechanism described below only applies to applications created before Mobile Backend Services 1.1.5. For applications created with Mobile Backend Services 1.1.5 and later, you must use range-based queries, as discussed in Query Pagination.

API calls which return arrays of objects take optional `page` and `per_page` arguments to specify the number of objects to return. By default, ten

objects are returned on each page, and the request may specify up to 20 results per page. Page numbers start at 1; if unspecified, the page defaults to page 1.

Data about the current page is included in the `meta` object. For paged responses, the `meta` object includes the `page`, `per_page`, `total_pages`, and `total_results` keys:

```
"meta": {
  "status":"ok",
  "code":200,
  "method_name":"showThreadMessages",
  "page":1,
  "per_page":10,
  "total_pages":1,
  "total_results":3
},
```