API

This document describes how to configure, connect to, and interact with the API. For information on expanding the API with your own custom code see Extending the API. All examples are presented using PHP.

Table of Contents

Configuration

Before you begin using the API you must first create a set of API credentials for the company you wish to connect to. For information on creating API credentials consult the System > API Access section of the User Manual.

Overview

Structure

The structure of an API request involves specifying the model, method, and format to be requested, along with any additional parameters the method may require. For example:

```
https://yourdomain.com/installpath/api/users/get.json?user_id=1
```

In the above example, **yourdomain.com** is your domain and **installpath** is the path to the Blesta installation. If your server does not support .htaccess include index.php in your URL (e.g. /installpath/index.php/api/users/get.json?user_id=1). **users** is the model to request, **get** is the method, and **json** is the format.

Requesting Models of Plugins

To request a specific model of a plugin format your request as /plugin.model/method.*format*

Formats

The API supports XML, JSON, and PHP serialization formats. By default JSON formatting is used. So if there is an error detecting the format of the request (due to a bad URL, for example) the error response will be returned in JSON format.

Errors

There are several types of errors that may be encountered when working with the API:

1. Sending invalid parameters will result in a 400 Bad Request response.

   ```
   HTTP/1.1 400 Bad Request
   Content-Length: 137

   {
    "message":"The request cannot be fulfilled due to bad syntax.",
    "errors":
    {
     "field":
     {
      "code":"Error message."
     }
    }
   }
   ```

2. Providing invalid credentials will result in a 401 Unauthorized response.

   ```
   HTTP/1.1 401 Unauthorized
   Content-Length: 67

   {"message":"The authorization details given appear to be invalid."}
   ```

3. Attempting to access a resource that is not callable will result in a 403 Forbidden response.

   ```
   HTTP/1.1 403 Forbidden
   Content-Length: 55

   {"message":"The requested resource is not accessible."}
   ```

4. Attempting to access a resource that does not exist will result in a 404 Not Found response.

```
HTTP/1.1 404 Not Found
Content-Length: 52

{"message":"The requested resource does not exist."}
```

5. Requesting a format that is not supported will result in a 415 Unsupported Media Type response.

```
HTTP/1.1 415 Unsupported Media Type
Content-Length: 66

{"message":"The format requested is not supported by the server."}
```

6. If an unexpected error occurs a 500 Internal Server Error will result. If this error is encountered consult the documentation for the method you are requesting.

```
HTTP/1.1 500 Internal Server Error
Content-Length: 42

{"message":"An unexpected error occured."}
```

7. If an unexpected error occurs that specifies that it "**Failed to retrieve the default value**" then you likely encounter this issue due to IonCube in Blesta. You can work-around this issue by ensuring that you specify all optional arguments to your API call.

```
HTTP/1.1 500 Internal Server Error
Content-Length: 108

{"message":"An unexpected error occured.","response":"Internal error: Failed to retrieve the default
value"}
```

8. When Blesta is under maintenance mode, the API will return a 503 Service Unavailable response.

```
HTTP/1.1 503 Service Unavailable
Content-Length: 81

{"message":"The requested resource is currently unavailable due to maintenance."}
```

All error response objects contain an array of input parameters that produced errors. From the example above "field" is the parameter name. Each field may contain one or more error codes along with a related message. Common codes are **empty**, **exists**, **format**, **length**, and **valid**, but many more are available. The error code is always the index of the error message, and is used primarily in identifying the type of error encountered.

Timestamps

The API returns all timestamps in UTC time using the following ISO 8601 format:

```
YYYY-MM-DD hh:mm:ss / 2021-12-31 12:00:00
```

The API expects timestamps in one of the following formats:

```
YYYY-MM-DD hh:mm:ss ±hh:mm / 2021-12-31 12:00:00 +00:00
YYYY-MM-DD hh:mm:ss UTC / 2021-12-31 12:00:00 UTC
YYYY-MM-DDThh:mm:ssZ / 2021-12-31T12:00:00Z
```

Always specify a timezone

If the timezone can not be determined from the timestamp, the system will assume the timestamp is in local time. For this reason you should always specify a timezone in your timestamps. For details on checking or setting the local timezone for the system check your localization settings.
Formatting dates with PHP

The easiest way to format a date into a suitable format using PHP is to use **date("c")**.

Authentication

The API supports Basic Authentication for web requests. When connecting to the API via a command line the API credentials must be passed as parameters. See Using the Command Line Interface for more information on connecting to the API via command line.

Running PHP in CGI or FCGId mode?

Update your .htaccess file to pass an environment variable to Blesta so it can capture the basic authentication details as per the following snippet:

.htaccess

```
RewriteEngine on
...
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

If running PHP-FPM, set CGIPassAuth On in your .htaccess, or within your httpd.conf like the example:

```
    <FilesMatch \.php$>
        CGIPassAuth On
        SetHandler proxy:unix:/var/php-fpm/166630281728629.sock|fcgi://127.0.0.1
    </FilesMatch>
```

Connecting

There are a number of ways to connect to the API. Choose the option that best fits your environment.

Remotely

To connect remotely, first determine the URL of the API for your installation. The default path is http://yourdomain.com/installpath/api/. Where yourdomain.com is the domain you've installed Blesta in, and installpath is the path to Blesta. If your server does not support .htaccess then the URL should instead appear as http://yourdomain.com/installpath/index.php/api/.

Safety First!

Because each request contains your API key, and may contain additional sensitive information, you should only process requests remotely over a secure connection (i.e. only use HTTPS)

Download the API SDK

The SDK includes an API processor that makes it super easy to work with the API.

Connecting Remotely with PHP

```
<?php
require_once "blesta_api.php";

$user = "username";
$key = "key";
$url = "https://yourdomain.com/installpath/api/";

$api = new BlestaApi($url, $user, $key);

$response = $api->get("users", "get", array('user_id'
=> 1));

print_r($response->response());
print_r($response->errors());
?>
```

Connecting Remotely with cURL

```
curl https://yourdomain.com/installpath/api/users/get.
json?user_id=1 -u username:key
```

Locally

Using the Command Line Interface

The API is available via a command line interface (CLI). The API supports the following command line parameters:

1. **-u** or **-user** The API user
2. **-k** or **-key** The API key
3. **-m** or **-method** The method for the request (GET, POST, PUT, DELETE)
4. **-p** or **-params** A URL encoded array of parameters as you would find in a URL string (e.g. first_name=John&last_name=Doe&company=Acme Co.)

Remember to quote parameters

If any of your command line parameters contain special characters or spaces you **must** wrap quote marks (') around the value.

Below is an example API request.

```
php index.php api/users/get.json -u USER -k KEY -m GET -p "user_id=1"
```

Within the Blesta environment

If you're working with or have created code within the Blesta environment, there's no need to use the API at all. All of the methods available in the API are first and foremost available to Blesta, in the form of models. To use these models, simply load the model within your environment.

Loading Models from a Controller

```
// from somewhere in your controller...
$this->uses(array("ModelName"));

// Now invoke it
$this->ModelName->someMethod($param1, $param2);
```

Loading Models elsewhere

```
// from any other class...
Loader::loadModels($this, array("ModelName"));

// Now invoke it
$this->ModelName->someMethod($param1, $param2);
```

HTTP Verbs

The API encourages the proper use of GET, POST, PUT, and DELETE when interacting with the API. For all POST requests the API will pass only post parameters to the requested resource. For PUT only put parameters will be passed. Similarly, for all GET and DELETE requests, the API will pass only get parameters to the requested resource.

Choose the Request Type Carefully

It is highly discouraged to use GET or DELETE for API requests where you are providing sensitive information. That sensitive information will be included as plain-text as get parameters in the URI. Instead, use POST or PUT to pass that sensitive information securely in the request body rather than the URI.

POST

Use POST requests when creating new records. For example, when adding a new user record via *api/users/add.json*.

GET

Use GET requests when retrieving record data. For example, when fetching a user record via *api/users/get.json*.

PUT

Use PUT requests when updating records. For example, when updating a user record via *api/users/edit.json*.

DELETE

Use DELETE requests when deleting records. For example, when deleting a user record via *api/users/delete.json*.

Requests

The API supports hundreds of requests, and many more through API extensions, so we can't document them all here. Instead, check out the source code documentation. All public model methods are callable through the API. To find documentation on a particular request pull up the related model and method from the source code documentation.

Examples

Below are a few basic examples to get you started.

API Request	Description	PHP	CURL

encryption/systemEncrypt	Encrypts the given value using 256-bit AES in CBC mode using a SHA-256 HMAC hash as the key, based on the system configured setting in Blesta.system_key	`<?php`
		`require_once "blesta_api.php";`
		
		`$user = "username";`
		`$key = "key";`
		`$url = "https://yourdomain.com/installpath/api/";`
		
		`$api = new BlestaApi($url, $user, $key);`
		
		`$response = $api->post("encryption", "systemEncrypt", array('value' => "my text"));`
		
		`print_r($response->response());`
		`print_r($response->errors());`
		`?>`
			`curl https://yourdomain.com/installpath/api/encryption/systemEncrypt.json -u username:key -d 'value=my text'`

encryption/systemDecrypt	Decrypts the given value using 256-bit AES in CBC mode using SHA-256 HMAC hash as the key, based on the system configured setting in Blesta.system_key	`<?php`
		`require_once "blesta_api.php";`
		
		`$user = "username";`
		`$key = "key";`
		`$url = "https://yourdomain.com/installpath/api/";`
		
		`$api = new BlestaApi($url, $user, $key);`
		
		`$response = $api->post("encryption", "systemDecrypt", array('value' => "b2J2aW91c2x5IG5vdCByZWFsbHkgZW5jcnlwdGVk"));`
		
		`print_r($response->response());`
		`print_r($response->errors());`
		`?>`
			`curl https://yourdomain.com/installpath/api/encryption/systemDecrypt.json -u username:key -d 'value=b2J2aW91c2x5IG5vdCByZWFsbHkgZW5jcnlwdGVk'`

invoices/add	Creates a new invoice using the given data	<?php
		require_once "blesta_api.php";
		
		$user = "username";
		$key = "key";
		$url = "https://yourdomain.com/installpath/api/";
		
		$api = new BlestaApi($url, $user, $key);
		
		$data = array(
		'vars' => array(
		'client_id' => 1,
		'date_billed' => date("c"),
		'date_due' => date("c"),
		'currency' => "USD",
		'lines' => array(
		array(
		'description' => "Line item #1",
		'amount' => "5.99"
		),
		array(
		'description' => "Line item #2",
		'amount' => "3.75",
		'qty' => 2
		)
		),
		'delivery' => array("email")
		)
		);
		$response = $api->post("invoices", "add", $data);
		
		print_r($response->response());
		print_r($response->errors());
		?>
			curl https://yourdomain.com/installpath/api/invoices/add.json -u username:key
			-d 'vars[client_id]=1'
			-d 'vars[date_billed]=2013-11-20T16:43:00-07:00'
			-d 'vars[date_due]=2013-11-20T16:43:00-07:00'
			-d 'vars[currency]=USD'
			-d 'vars[lines][0][description]=Line item #1'
			-d 'vars[lines][0][amount]=5.99'
			-d 'vars[lines][1][description]=Line item #2'
			-d 'vars[lines][1][amount]=3.75'
			-d 'vars[lines][1][qty]=2'
			-d 'vars[delivery][0]=email'

users/auth	Determines whether the user credentials are valid to be authenticated with Blesta	<?php
		require_once "blesta_api.php";
		
		$user = "username";
		$key = "key";
		$url = "https://yourdomain.com
		/installpath/api/";
		
		$api = new BlestaApi($url,
		$user, $key);
		
		$data = array(
		'username' => 'myuser',
		'vars' => array(
		'username' =>
		'myuser',
		'password' =>
		'mypassword'
		),
		'type' => 'any'
		);
		$response = $api->post("users",
		"auth", $data);
		
		print_r($response->response());
		print_r($response->errors());
		?>
			curl https://yourdomain.com
			/installpath/api/users/auth.json -u
			username:key
			-d 'username=myuser'
			-d 'vars[username]=myuser'
			-d 'vars[password]=mypassword'
			-d 'type=any'

How to Read the Source Docs

The source code documentation contains documentation on everything in Blesta, but the API only supports calling Model methods. You can find all core model methods under **blesta > app > models**, however plugin model methods are also callable through the API.