**Chef Infra Server API**

The Chef Infra Server API is a REST API that provides access to objects on the Chef Infra Server.

The objects which can be accessed are listed below:

1. Nodes

2. Environments

3. Roles

4. Cookbooks (and cookbook versions)

5. And also to manage an API client list and the associated RSA public key-pairs.

**Requirements**

The Chef Infra Server API has the following requirements:

* The Accept header must be set to application/json.
* For PUT and POST requests, the Content-Type header must be set to application/json.
* The X-Chef-Version header must be set to the version of the Chef Infra Server API that is being used.
* A request must be signed using Mixlib::Authentication.
* A request must be well-formatted. The easiest way to ensure a well-formatted request is to use the Chef::REST library.

## Authentication Headers

Authentication to the Chef Infra Server occurs when a specific set of HTTP headers are signed using a private key that is associated with the machine from which the request is made. The request is authorized if the Chef Infra Server can verify the signature using the public key. Only authorized actions are allowed.

### Header Format

By default, all hashing is done using SHA-1 and encoded in Base64. Base64 encoding should have line breaks every 60 characters.

**Example:**

Method:HTTP\_METHOD

Hashed Path:HASHED\_PATH

X-Ops-Content-Hash:HASHED\_BODY

X-Ops-Timestamp:TIME

X-Ops-UserId:USERID

where:

* HTTP\_METHOD is the method used in the API request (GET, POST, and so on)
* HASHED\_PATH is the path of the request: /organizations/NAME/name\_of\_endpoint. The HASHED\_PATH must be hashed using SHA-1 and encoded using Base64, must not have repeated forward slashes (/), must not end in a forward slash (unless the path is /), and must not include a query string.
* The private key must be an RSA key in the SSL .pem file format. This signature is then broken into character strings (of not more than 60 characters per line) and placed in the header.

The Chef Infra Server decrypts this header and ensures its content matches the content of the non-encrypted headers that were in the request. The timestamp of the message is checked to ensure the request was received within a reasonable amount of time. One approach generating the signed headers is to use [mixlib-authentication](https://github.com/chef/mixlib-authentication), which is a class-based header signing authentication object similar to the one used by the Chef Infra Client.

## Global Endpoints

A global endpoint may be used to access all of the organizations on the Chef Infra Server.

### /organizations

The Chef Infra Server may contain multiple organizations.

The /organizations endpoint has the following methods: GET and POST.

#### GET

The GET method is used to get a list of organizations on the Chef Infra Server.

This method has no parameters.

**Request**

GET /organizations

**Response**

The response is similar to:

{

"org\_name1": https://url/for/org\_name1",

"org\_name2": https://url/for/org\_name2"

}

#### GET

The GET method is used to get the details for the named organization.

This method has no parameters.

**Request**

GET /organizations/NAME

**Response**

The response is similar to:

{

"name": "chef",

"full\_name": "Chef Software, Inc",

"guid": "f980d1asdfda0331235s00ff36862"

...

}

### \_status

The /\_status endpoint can be used to check the status of communications between the front and back end servers. This endpoint is located at /\_status on the front end servers.

**Request**

api.get("https://chef\_server.front\_end.url/\_status")

This method has no request body.

**Response**

The response will return something like the following:

{

"status": "pong",

"upstreams":

{

"service\_name": "pong",

"service\_name": "pong",

...

}

}

#### GET

The GET method is used to get a list of users on the Chef Infra Server.

This method has no parameters.

**Request**

GET /users

**Response**

The response is similar to:

{

"user1": "https://url/for/user1",

"user2": "https://url/for/user2"

}