

# Slicehost API

Version 1.3  
May 22, 2008

<b>Slicehost API</b>	<b>3</b>
<b>Abstract</b>	<b>3</b>
<b>Introduction</b>	<b>3</b>
<b>Getting Started</b>	<b>3</b>
<i>Authentication</i>	<i>3</i>
<i>ActiveResource</i>	<i>3</i>
<i>Ruby Example</i>	<i>4</i>
<i>Python Example</i>	<i>6</i>
<b>Services</b>	<b>8</b>
<b>Slices</b>	<b>8</b>
<i>Flavor</i>	<i>8</i>
<i>Image</i>	<i>8</i>
<i>Slice</i>	<i>9</i>
<i>Examples in Ruby (using the ActiveResource classes defined above)</i>	<i>9</i>
<b>DNS</b>	<b>10</b>
<i>Zone</i>	<i>10</i>
<i>Record</i>	<i>10</i>
<b>Retrieving API Information</b>	<b>11</b>
<b>Common Errors</b>	<b>11</b>

# Slicehost API

## Abstract

The Slicehost API is an interface to Slicehost services, allowing users to automate tasks as needed. The current iteration allows access to DNS and Slices (read more about limited accessibility below).

## Introduction

To use the Slicehost API, you must have a Slicehost account. You may enable or disable API access from this account, and you may re-generate your API password as you see fit.

This API follows a standard ActiveRecord pattern as seen in Ruby on Rails. The examples in this document are in Ruby, using the Ruby on Rails ActiveRecord library.

## Getting Started

### Authentication

Before you can access the API service, you must enable the service in the SliceManager under the Account tab. Note that this automatically generates a unique API password for you. Should you need to update the password, you can generate a new one from the same page at any time.

Authentication for the API uses standard HTTP Authentication which uses your unique password as part of the URL.

Here's an example:

```
https://3da541559918a808c2402bba5012f6c60b27661c@api.slicehost.com/
```

### ActiveResource

To learn about how to use ActiveRecord within Ruby (or to develop a way to do it in another language), we suggest reading the [ActiveResource README](#).

## Ruby Example

Below is an example of using one's API password inActiveResource classes:

```
require 'activeresource'

API_PASSWORD = "3da541559918a808c2402bba5012f6c60b27661c"

class Slice < ActiveResource::Base
  self.site = "https://#{API_PASSWORD}@api.slicehost.com/"
end

class Zone < ActiveResource::Base
  self.site = "https://#{API_PASSWORD}@api.slicehost.com/"
end

class Record < ActiveResource::Base
  self.site = "https://#{API_PASSWORD}@api.slicehost.com/"
end

# Creating a new Slice
slice = Slice.new(:image_id => 1, :flavor_id => 1, :name => 'example')
slice.save

# Updating the name of the Slice
slice.name = 'example.org'
slice.save

# Creating a new Zone
myzone = Zone.new(:origin => 'example.com', :ttl => 3000)
myzone.save

# Creating a record for that Zone
myrecord = Record.new(:record_type => 'A', :zone_id => 12345,
                     :name => 'www', :data => '127.0.0.1')
myrecord.save

# Updating the record
myrecord.ttl = 55000
myrecord.save

# Deleting the record
myrecord.destroy

# Back to our Zone
zid = myzone.id # The ID of the new Zone
              # Let's use this to re-retrieve the Zone

myzone = nil
```

```
myzone = Zone.find(zid) # Retrieving the same Zone we just created
myzone.ttl = 8000
myzone.save # Updating the TTL

myzone.destroy # Destroying the Zone
```

## Python Example

For Python, our own Jared Kuolt has created a library called [pyactiveresource](#), which we'll use in this example:

```
from pyactiveresource import ActiveResource

api_password = '3da541559918a808c2402bba5012f6c60b27661c'
api_site = 'https://%s@api.slicehost.com/' % api_password

class Slice(ActiveResource):
    class Meta:
        site = api_site

class Zone(ActiveResource):
    class Meta:
        site = api_site

class Record(ActiveResource):
    class Meta:
        site = api_site

# Creating a new Slice
slice = Slice(image_id=1, flavor_id=1, name='example')
slice.save()

# Updating the name of the Slice
slice.name = 'example.org'
slice.save()

# Creating a new Zone
myzone = Zone(origin='example.com', ttl=3000)
myzone.save()

# Creating a record for that Zone
myrecord = Record(record_type='A', zone_id=12345,
                  name='www', data='127.0.0.1')
myrecord.save()

# Updating the record
myrecord.ttl = 55000
myrecord.save()

# Deleting the record
myrecord.destroy

# Back to our Zone
zid = myzone.id # The ID of the new Zone
                # Let's use this to re-retrieve the Zone
```

```
myzone = None
myzone = Zone.find(id=zid)[0] # Retrieving the Zone we just created
myzone.ttl = 8000
myzone.save() # Updating the TTL

myzone.destroy() # Destroying the Zone
```

# Services

Not all of the services Slicehost provides are accessible via this API, however, we plan to offer as many API services as possible. Fields in bold are required.

**Note:** In addition to all fields outlined below, the `id` field is the identifier of an object which may not be changed.

## Slices

Slices, as a resource, are a special case. At this time we allow for the following actions in the API:

- Creating
- Renaming
- Rebuilding
- Rebooting (soft and hard)

There are two read-only resources you need to access to create a Slice: flavors and images. Flavors are the plans that we offer, e.g. 256 Slice. Images are the Linux distributions we offer.

### Flavor

Field	Access	Notes
name	r	Verbose name for the flavor, e.g. "256 slice"
price	r	The price as an integer of cents. For example: 2000 equals \$20.00. Note that all prices are in USD
ram	r	The amount of RAM (in Megabytes) included with the plan

### Image

Field	Access	Notes
name	r	Example: Ubuntu 8.04 LTS (hardy)

## Slice

Field	Access	Notes
name	rw	A string to identify the slice
flavor_id	rw	Flavor (see above) *
image_id	rw	Image (see above) *
ip_address	r	Slice's initial IP address (the API does not currently show multiple IPs)
root_password	r	Shown only on creation

\* This field may only be set on creation

In addition to creating and modifying Slices, you may currently reboot, hard reboot, and rebuild a slice. Each of these uses an ActiveResource “custom method”, which is simply a PUT request to the action:

```
https://apikey@api.slicehost.com/slices/xxxx/hard_reboot.xml
https://apikey@api.slicehost.com/slices/xxxx/reboot.xml
https://apikey@api.slicehost.com/slices/xxxx/rebuild.xml?image_id=x
```

**Note:** rebuilding requires the image\_id parameter.

### Examples in Ruby

```
s = Slice.find(xxx)
s.put(:reboot)
s.put(:hard_reboot)
s.put(:rebuild, :image_id => x)
```

### Examples in Python

**Note:** this requires pyactiveresource >= 0.2.3.

```
s = Slice.find(xxx)
s.perform_action('reboot', 'PUT')
s.perform_action('hard_reboot', 'PUT')
s.perform_action('rebuild', 'PUT', {'image_id': x})
```

# DNS

DNS is split into two sets of resources: Zones and Records. Zones represent a domain, whereas Records represent the records within a Zone.

## Zone

Field	Access	Notes
origin	rw	The origin field is the domain name, e.g. "example.com."
ttl	rw	Must be an integer and above 60
active	rw	"Y" or "N" indicating whether Zone is active

## Record

Field	Access	Notes
record_type	rw	Must be a valid record type: A, CNAME, MX, NS, SRV, TXT, AAAA, PTR
zone_id	rw	Must be an id of a Zone owned by the same customer. <b>Note:</b> this may not be changed after creation.
name	rw	Must be a valid name
data	rw	Must be valid data per the record_type
ttl	rw	Must be an integer and above 60
active	rw	"Y" or "N" indicating whether Record is active
aux	rw	Auxiliary info for the record

## Retrieving API Information

You may retrieve API information at any time from the URL `/api.xml`. This information includes the API version number, release date, and link to the documentation. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<api>
  <version>1.0</version>
  <date>2008-01-01</date>
  <documentation>http://articles.slicehost.com/api</documentation>
</api>
```

**Note:** the API documentation will correspond in the major.minor version number. For example, API doc 1.2 corresponds to API version 1.2, and API doc 1.3.x corresponds to API version 1.3.

## Common Errors

The following are common error messages and what they mean:

HTTP Code	Meaning
503	The service is temporarily disabled. This will happen when we perform maintenance; The API service will likely be back online shortly.
422	Unprocessable Entity; The request, or parameters in the request, were malformed. The most common reason for this is an invalid value, e.g. "A" for a Zone TTL.
401	Unauthorized; Your API Password is invalid.