(http://www.enchant.com)

REST API

Introduction

Enchant is a fully hosted solution to manage email and twitter based customer support. We provide a REST (http://en.wikipedia.org/wiki/Representational_state_transfer) API built on pragmatic RESTful design (http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api) principles.

Our API uses resource-oriented URLs that leverage built in features of HTTP, like authentication, verbs and response codes. All request and response bodies are JSON (http://en.wikipedia.org/wiki/JSON) encoded, including error responses. Any off-the-shelf HTTP client can be used to communicate with the API.

We believe an API is a user interface for a developer - accordingly, we've made sure our API can be easily explored from the browser!

Changes

This is a versionless API. Advance notification of breaking changes will be available in this document and will be sent by email to our developer mailing list.

Authentication

This API is authenticated using HTTP Basic Auth (http://en.wikipedia.org/wiki/Basic_access_authentication) over HTTPS (http://en.wikipedia.org/wiki/HTTP_Secure). Any requests over plain HTTP will fail.

A user's email address and password can be provided as auth credentials:

```
$ curl -u email:password https://site.enchant.com/api/v1/tickets
```

All requests are associated with a specific user in Enchant and permissions are limited to that user's capabilities.

Requests

The base URL of the API is `https://site.enchant.com/api/v1` where `site` should be replaced with your help desk identifier.

JSON Bodies

All `POST`, `PUT`, `PATCH` requests are JSON (http://en.wikipedia.org/wiki/JSON) encoded and must have content type of `application/json`, or the API will return a `415 Unsupported Media Type` status code.

```
$ curl -u email:password https://site.enchant.com/api/v1/users/543abc \
    -X PATCH \
    -H 'Content-Type: application/json' \
    -d '{"first_name":"John"}'
```

HTTP Verbs

We use standard HTTP verbs to indicate intent of a request:

- `GET` - To retrieve a resource or a collection of resources
- `POST` - To create a resource
- `PATCH` - To modify a resource
- `PUT` - To set a resource
- `DELETE` - To delete a resource

Limited HTTP Clients

If you are using an HTTP client that doesn't support `PUT`, `PATCH` or `DELETE` requests, send a `POST` request with an `X-HTTP-Method-Override` header specifying the desired verb.

```
$ curl -u email:password https://site.enchant.com/api/v1/users/543abc \
    -X POST \
    -H "X-HTTP-Method-Override: DELETE"
```

Responses

All response bodies are JSON (http://en.wikipedia.org/wiki/JSON) encoded.

A single resource is represented as a JSON object:

```
{
  "field1": "value",
  "field2": true,
  "field3": []
}
```

A collection of resources is represented as a JSON array of objects:

```
[
  {
    "field1": "value",
    "field2": true,
    "field3": []
  },
  {
    "field1": "another value",
    "field2": false,
    "field3": []
  }
]
```

Timestamps are in UTC (http://en.wikipedia.org/wiki/Coordinated_Universal_Time) and formatted as ISO8601 (http://en.wikipedia.org/wiki/ISO_8601).

Unset fields will be represented as a `null` instead of not being present. If the field is an array, it will be represented as an empty array - ie `[]`.

HTTP Status Codes

We use HTTP status codes to indicate success or failure of a request.

Success codes:

- `200 OK` - Request succeeded. Response included
- `201 Created` - Resource created. URL to new resource in Location header
- `204 No Content` - Request succeeded, but no response body

Error codes:

- `400 Bad Request` - Could not parse request
- `401 Unauthorized` - No authentication credentials provided or authentication failed
- `403 Forbidden` - Authenticated user does not have access
- `404 Not Found` - Resource not found
- `415 Unsupported Media Type` - POST/PUT/PATCH request occurred without a `application/json` content type
- `422 Unprocessable Entry` - A request to modify or create a resource failed due to a validation error
- `429 Too Many Requests` - Request rejected due to rate limiting
- `500, 501, 502, 503, etc` - An internal server error occured

Errors

All 400 series errors (400, 401, 403, etc) will be returned with a JSON object in the body and a `application/json` content type.

```
{
   "message": "Not Found"
}
```

500 series error codes (500, 501, 502, etc) do not return JSON bodies.

Validation Errors

In case of validation errors on a POST/PUT/PATCH request, a `422 Unprocessable Entry` status code will be returned. The JSON response body will include an array of error messages.

```
{
  "message": "Validation Failed",
  "errors": [
    {
      "message": "Field is not valid"
    },
    {
      "message": "OtherField is already used"
    }
  ]
}
```

Rate Limiting

The API is rate limited to 100 credits per minute for an entire help desk, across all end points, users and tokens. A request is typically worth 1 credit. However, embedding and counting can increase the the amount of required credits for a request. All responses include headers describing the current rate limit status:

```
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
```

- `Rate-Limit-Limit` - Total credit for current period
- `Rate-Limit-Remaining` - Remaining credit for current period
- `Rate-Limit-Used` - Number of credits used for this request
- `Rate-Limit-Reset` - Number of seconds until the the credit count resets

If the rate limit is hit, the API will return a `429 Too Many Requests` status code. In this situation, your application should not send any further requests until `Rate-Limit-Reset` seconds have elapsed.

Field Filtering

All responses from the API can limit fields to only the fields you need. Just pass in a `fields` query parameter with a comma separated list of fields you need

For example:

```
GET /api/v1/users?fields=id,first_name
```

Would have the following response body:

```
[
  {
    "id": "543abc",
    "first_name:": "John"
  },
  {
    "id": "543add",
    "first_name:": "Bob"
  }
]
```

Embedding

Many endpoints support embedding related resources to minimize the number of required API round trips.

Embedding is triggered by passing in an `embed` query parameter, which takes a comma separated list of endpoint types.

```
GET /api/v1/tickets/543abc?embed=labels
```

```
{
  "id": "543add",
  "type": "email",
  "label_ids": [ "123abc", "234bcd" ],
  "labels": [
    {
      "id": "123abc",
      "name": "Refund"
    },
    {
      "id": "234bcd",
      "name": "VIP"
    }
  ],
  ... other ticket fields ...
}
```

Only certain resource types can be embedded from certain endpoints. The endpoint documentation specifies which ones can be embedded.

Each embedded type uses an additional rate limit credit.

Counting

All endpoints that return a collection can provide a count of the total number of results. To request a count, just include a `count=true` as a query parameter. The count will be returned in a header `Total-Count`.

```
GET /api/v1/tickets?count=true
```

```
200 OK
Total-Count: 135
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 98
Rate-Limit-Used: 2
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
[
  ... results ...
]
```

Note that the count represents the total amount of available results, not the amount returned as part of the current response.

Counting uses an additional rate limit credit.

Enveloping

If your HTTP client makes it difficult to read status codes or headers, we can package everything neatly into the response body. Just include `envelope=true` as a request parameter and the API will always return a 200 HTTP status code. The real status, headers and response will be within the body.

```
GET /api/v1/users/does-not-exist?envelope=true
```

```
200 OK
```

```
{
  "status": 404,
  "headers": {
    "Rate-Limit-Limit": 100,
    "Rate-Limit-Remaining": 50,
    "Rate-Limit-Used": 0,
    "Rate-Limit-Reset": 25
  },
  "response": {
    "message": "Not Found"
  }
}
```

Pagination

Requests for collections can return between 0 and 100 results, controlled using the `per_page` and `page` query parameters. All end points are limited to 10 results by default.

```
GET /api/v1/tickets?per_page=15&page=2
```

Not all endpoints support pagination. If they do, it will be mentioned in their documentation.

Sorting

Some endpoints offer result sorting, triggered using the `sort` query parameter. The value of sort is a comma separated list of fields to sort by. You can specify descending sort by prepending `-` to a field. Not all fields can be sorted on. The endpoint documentation will list supported sort options.

The default sort for all endpoints is descending order of creation.

To get recently updated tickets, sorted in descending order of updated_at:

```
GET /api/v1/tickets?sort=-updated_at
```

Tickets

All requests are tracked as tickets. A ticket contains one or more messages.

Field	Type	Notes
`id`	string	
`number`	integer	Friendly ticket number, unique across the help desk.
`customer_id`	string	Associated customer.
`user_id`	string	Assigned user. Can be null.
`group_id`	string	Associated inbox. A ticket always belongs to an inbox.
`label_ids`	array of strings	
`state`	string	One of: `open`, `hold`, `closed`
`subject`	string	
`type`	string	One of: `email`, `twitter`
`reply_to`	string	Default To: field of a new reply.
`reply_cc`	string	Default Cc: field of a new reply.
`spam`	boolean	
`trash`	boolean	

Field	Type	Notes
updated_at	timestamp	
created_at	timestamp	

Listing tickets

```
GET /api/v1/tickets
```

Query Parameter	Notes
id	Comma separated list of ticket ids
group_id	Comma separated list of inbox ids
state	Comma separated list of states: `open`, `hold`, `closed`
user_id	Comma separated list of user ids
label_id	Comma separated list of label ids
spam	One of: `true`, `false`
trash	One of: `true`, `false`
since_created_at	Any tickets created since specified time (should be in UTC and ISO8601 formatted) eg: 2013-08-02T13:41:10Z
since_updated_at	Any tickets updated since specified time (should be in UTC and ISO8601 formatted) eg: 2013-08-02T13:41:10Z
page	Page number, starting from 1
per_page	Any integer between 0 and 100, inclusive
sort	One of: `-updated_at`, `-created_at`, `user_id,-updated_at`
fields	See Field Filtering
count	See Counting
envelope	See Enveloping

Sample Request for unassigned tickets:

```
GET /api/v1/tickets?user_id=null&sort=-updated_at&count=true&per_page=1
```

```
200 OK
Total-Count: 23
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 98
Rate-Limit-Used: 2
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
[
  {
    "id": "51e6e9",
    "number": 2209,
    "user_id": "501efc",
    "state": "closed",
    "subject": "help, it didn't work!",
    "labels_ids": [
      "41e3dc"
    ],
    "customer_id": "51b3e9",
    "type": "email",
    "reply_to": "michael@example.com",
    "reply_cc": "",
    "group_id": "51e3f9",
    "spam": false,
    "trash": false,
    "updated_at": "2013-07-25T12:19:33Z",
    "created_at": "2013-07-17T18:58:44Z",
    "summary": "I was trying to do something and it just didn't work.  Can you help?"
  }
]
```


Getting a single ticket

```
GET /api/v1/tickets/ticket_id
```

Query Parameter	Notes
`embed`	Comma separated list of embeddables: `user`, `group`, `customer`, `labels`, `messages`
`fields`	See Field Filtering
`envelope`	See Enveloping

Sample Request:

```
GET /api/v1/tickets/51e6e9&embed=customer
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 96
Rate-Limit-Used: 2
Rate-Limit-Reset: 18
Content-Type: application/json
```

```
{
  "id": "51e6e9",
  "number": 2209,
  "user_id": "501efc",
  "state": "closed",
  "subject": "help, it didn't work!",
  "label_ids": [
    "41e3dc"
  ],
  "customer_id": "51b3e9",
  "customer": {
    "id": "51b3e9",
    "first_name": "Michael",
    "last_name": "Smith",
    "summary": null,
    "contacts": [
      {
        "id": "5149d9",
        "type": "email",
        "value": "michael.smith@example.com",
      }
    ]
  },
  "type": "email",
  "group_id": "51e3f9",
  "spam": false,
  "trash": false,
  "updated_at": "2013-07-25T12:19:33Z",
  "created_at": "2013-07-17T18:58:44Z",
  "summary": "I was trying to do something and it just didn't work.  Can you help?"
}
```


Updating a ticket

```
PATCH /api/v1/tickets/ticket_id
```

Field	Notes
`user_id`	
`group_id`	This is the inbox id
`state`	One of: `open`, `hold`, `closed`
`label_ids`	Array of label id strings to set the ticket to. An Empty array will remove all labels from the ticket.
`spam`	One of: `true`, `false`. spam and trash cannot both be true at the same time.
`trash`	One of: `true`, `false`. spam and trash cannot both be true at the same time.
`subject`	

Closing a ticket:

```
PATCH /api/v1/tickets/51b3e9
Content-Type: application/json
```

```
{
  "state": "closed"
}
```

Assigning to a specific user:

```
PATCH /api/v1/tickets/51b3e9
Content-Type: application/json
```

```
{
  "user_id": "555acc"
}
```

Sending to trash:

```
PATCH /api/v1/tickets/51b3e9
Content-Type: application/json
```

```
{
  "trash": true
}
```

Setting labels:

```
PATCH /api/v1/tickets/51b3e9
Content-Type: application/json
```

```
{
  "label_ids": [
    "41e3dc",
    "423ccc"
  ]
}
```

Adding or removing labels

An additional endpoint is available to add or remove labels without impacting existing labels on the ticket.

Adding a label:

```
PUT /api/v1/tickets/ticket_id/labels/label_id
Content-Type: application/json
```

Adding multiple labels:

```
PUT /api/v1/tickets/ticket_id/labels/label_id1,label_id2
Content-Type: application/json
```

Removing a label:

```
DELETE /api/v1/tickets/ticket_id/labels/label_id
```

Removing multiple labels:

```
DELETE /api/v1/tickets/ticket_id/labels/label_id1,label_id2
```

Creating a new ticket

```
POST /api/v1/tickets
```

Field	Required	Notes
`type`	yes	Must be set to `email`. Twitter tickets cannot be created from the API.
`subject`	yes	Subject of the ticket
`customer_id`	yes	Associated customer
`reply_to`	yes	Default To: address for replies
`reply_cc`	no	Default Cc: address for replies
`user_id`	no	
`group_id`	no	

Sample Request:

```
POST /api/v1/tickets
Content-Type: application/json
```

```
{
  "customer_id": "4332ca",
  "reply_to": "john@example.com",
  "type": "email",
  "user_id": "522aaa",
  "group_id": "533bcd",
  "subject": "Oh no, it happened again!"
}
```

```
201 Created
Location: https://site.enchant.com/api/v1/tickets/51f6e9
Content-Type: application/json
```

```
{
  "id": "51f6e9",
  "number": 2210,
  "user_id": "522aaa",
  "state": "open",
  "subject": "Oh no, it happened again!",
  "label_ids": [
  ],
  "customer_id": "4332ca",
  "type": "email",
  "group_id": "533bcd",
  "spam": false,
  "trash": false,
  "updated_at": "2013-07-17T18:58:44Z",
  "created_at": "2013-07-17T18:58:44Z",
  "summary": null
}
```


Messages

Replies and notes are all represented as messages on a ticket.

Field	Type	Notes
`id`	string	
`from_name`	string	display name associated with message
`body`	string	
`htmlized`	boolean	true if the body is html
`attachments`	array of attachments	
`user_id`	string	id of user who created the message. null if a customer
`type`	string	One of: `reply` or `note`
`created_at`	timestamp	

Additional fields for email reply messages:

Field	Type	Notes
`direction`	string	one of: `in` - an inbound message from a customer `out` - an outbound message to a customer
`from`	string	`From` header
`to`	string	`To` header
`cc`	string	`Cc` header
`bcc`	string	`Bcc` header
`reply_to`	string	`Reply-To` header

Field	Type	Notes
`subject`	string	`Subject` header
`more_body`	string	contains signature if removed from body.

Additional fields for twitter reply messages:

Field	Type	Notes
`screen_name`	string	
`tweet_id`	string	Twitter Status ID
`direction`	string	one of: `in`, `out`

Listing messages

Messages can be retrieved as embedded resources on a ticket.

```
GET /api/v1/tickets/ticket_id?embed=messages
```

Sample Request:

```
GET /api/v1/tickets/513e9c?embed=messages
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 98
Rate-Limit-Used: 2
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
{
  "id": "513e9c",
  "number": 224,
  "messages": [
    {
      "id": "51f613",
      "type": "reply",
      "direction": "in",
      "from_name": "Chris Johnson",
      "body": "<p>I was trying to do something and it just didn't work.</p><p>Can you help?</p>",
      "htmlized": true,
      "from": "Chris Johnson <chris@example.com>",
      "to": "support@example.com",
      "cc": null,
      "bcc": null,
      "reply_to": null,
      "subject": "i need some help",
      "user_id": null,
      "attachments": [
      ],
      "created_at": "2013-07-29T07:03:10Z"
    },
    {
      "id": "51f677",
      "type": "reply",
      "direction": "out",
      "from_name": "Vinay Sahni",
      "body": "Hi Chris,<br><br>Ofcourse I can help!<br><br>Cheers,<br>Vinay",
      "htmlized": true,
      "user_id": "501efc",
      "from": null,
      "to": "chris@example.com",
      "cc": null,
      "bcc": null,
      "reply_to": null,
      "subject": null,
      "attachments": [
      ],
      "created_at": "2013-07-29T14:10:58Z"
    }
  ],
  "user_id": "501efc",
  "state": "closed",
  "subject": "i need some help",
  "label_ids": [
  ],
  "customer_id": "51f613",
  "type": "email",
  "reply_to": "chris@example.com",
  "reply_cc": "",
  "group_id": "513210",
  "updated_at": "2013-07-29T14:10:59Z",
  "created_at": "2013-07-29T07:03:10Z",
  "spam": false,
  "trash": false,
  "summary": "I was trying to do something and it just didn't work. Can you help?"
}
```


Creating a new note

```
POST /api/v1/tickets/ticket_id/messages
```

Field	Type	Required	Notes
`type`	string	yes	Must be set to: `note`
`user_id`	string	yes	id of user to associate with the message.
`body`	string	yes	
`htmlized`	boolean	yes	`true` if the body is html
`attachment_ids`	array of strings	no	See attachments

Creating a new inbound email reply

```
POST /api/v1/tickets/ticket_id/messages
```

Field	Type	Required	Notes
`type`	string	yes	Must be set to: `reply`
`direction`	string	yes	Must be set to: `in`
`body`	string	yes	
`htmlized`	boolean	yes	`true` if the body is html
`from`	string	yes	
`to`	string	yes	
`cc`	string	no	
`subject`	string	no	
`reply_to`	string	no	
`attachment_ids`	array of strings	no	See attachments

Creating a new outbound email reply

```
POST /api/v1/tickets/ticket_id/messages
```

Field	Type	Required	Notes
`type`	string	yes	Must be set to: `reply`
`direction`	string	yes	Must be set to: `out`
`body`	string	yes	
`htmlized`	boolean	yes	`true` if the body is html
`user_id`	string	yes	id of user to associate with the message.
`to`	string	yes	

Field	Type	Required	Notes
cc	string	no	
bcc	string	no	
attachment_ids	array of strings	no	See attachments

Sample request to add a reply:

```
POST /api/v1/tickets/513ecc/messages
Content-Type: application/json
```

```
{
  "direction": "out",
  "type": "reply",
  "body": "Hi Chris,<br><br>Ofcourse I can help!<br><br>Cheers,<br>Vinay",
  "htmlized": true,
  "user_id": "501efc"
}
```

Attachments

Attachments are associated with messages. After uploading an attachment, a message must be created using the attachment id. An attachment can only be associated with one message. Any unassociated attachments will be deleted.

Field	Type	Notes
id	string	
name	string	file name
size	integer	size in bytes
type	string	mime type

Getting an attachment

```
GET /api/v1/attachments/attachment_id
```

Sample Request:

```
GET /api/v1/attachments/54318c
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
{
  "id": "54318c",
  "name": "MyFile.zip",
  "size": 134345,
  "type": "application/zip"
}
```

Creating an attachment

```
POST /api/v1/attachments
```

Field	Type	Required	Notes
`name`	string	yes	file name
`type`	string	yes	mime type
`data`	string	yes	Base64 encoded file data

Sample attachment, including message creation:

```
POST /api/v1/attachments
Content-Type: application/json
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
```

```
{
  "name": "MyFile.zip",
  "type": "application/zip",
  "data": "UEsDBAoAAAAAAFmu/kLPx4wKGwAAABsAAAAKABwAcmVhZG1lLnR4dFVUCQAD
           OjX4UTo1+FF1eAsAAQToAwAABOgDAABNYXkgdGhlIGZvcmNlIGJlIHdpdGgg
           eW91LgpQSwECHgMKAAAAAABZrv5Cz8eMChsAAAAbAAAACgAYAAAAAAAAAAAA
           tIEAAAAAcmVhZG1lLnR4dFVUBQADOjX4UXV4CwABBOgDAAAE6AMAAFBLBQYA
           AAAAAQABAFAAAABfAAAAAA="
}
```

```
201 Created
Location: https://site.enchant.com/api/v1/attachments/5338df
Content-Type: application/json
```

```
{
  "id": "5338df",
  "name": "MyFile.zip",
  "type": "application/zip",
  "size": 197
}
```

Now, a message needs to be created using the attachment:

```
POST /api/v1/tickets/513ecc/messages
Content-Type: application/json
```

```
{
  "type": "note",
  "body": "Attaching the file!",
  "htmlized": false,
  "user_id": "501efc",
  "attachment_ids": [
    "5338df"
  ]
}
```

Users

Field	Type	Notes
id	string	
first_name	string	
last_name	string	
email	string	

Listing users

```
GET /api/v1/users
```

Sample Request:

```
GET /api/v1/users
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
[
  {
    "id": "501efc",
    "first_name": "John",
    "last_name": "Smith",
    "email": "john@example.com"
  },
  {
    "id": "501edd",
    "first_name": "Michael",
    "last_name": "Jones",
    "email": "michael@example.com"
  }
]
```

Customers

Field	Type	Notes
id	string	
first_name	string	
last_name	string	
summary	string	
contacts	array of contacts	

Listing customers

```
GET /api/v1/customers
```

Query Parameter	Notes
contacts.type	One of: `email`, `twitter`
contacts.value	Email address or twitter screen name
page	Page number, starting from 1
per_page	Any integer between 0 and 100, inclusive
fields	See Field Filtering
count	See Counting
envelope	See Enveloping

Sample Request:

```
GET /api/v1/customers
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
[
  {
    "id": "51f846",
    "first_name": "Ruby",
    "last_name": "Miller",
    "summary": null,
    "contacts": [
      {
        "id": "51f8aa",
        "type": "email",
        "value": "ruby@example.com"
      }
    ]
  },
  {
    "id": "51f7a7",
    "first_name": "Maggie",
    "last_name": "Miller",
    "summary": null,
    "contacts": [
      {
        "id": "51f7a9",
        "type": "email",
        "value": "maggie@example.com"
      }
    ]
  }
]
```


Find a customer by email address

```
GET /api/v1/customers?contacts.type=email&contacts.value=email_address
```

Sample Request:

```
GET /api/v1/customers?contacts.type=email&contacts.value=john@example.com
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
[
  {
    "id": "51f232",
    "first_name": "John",
    "last_name": "Smith",
    "summary": null,
    "contacts": [
      {
        "id": "51f352",
        "type": "email",
        "value": "john@example.com"
      }
    ]
  }
]
```

Get a single customer

```
GET /api/v1/customers/customer_id
```

Sample Request:

```
GET /api/v1/customers/51f7a7
```

```
200 OK
Rate-Limit-Limit: 100
Rate-Limit-Remaining: 99
Rate-Limit-Used: 1
Rate-Limit-Reset: 20
Content-Type: application/json
```

```
{
  "id": "51f7a7",
  "first_name": "Maggie",
  "last_name": "Miller",
  "summary": null,
  "contacts": [
    {
      "id": "51f7a9",
      "type": "email",
      "value": "maggie@example.com"
    }
  ]
}
```

Updating a customer

```
PATCH /api/v1/customers/customer_id
```

Field	Required	Notes
first_name	no	
last_name	no	

Field	Required	Notes
summary	no	

Updating a customer's name:

```
PATCH /api/v1/customers/51f7a7
Content-Type: application/json
```

```
{
  "first_name": "Maggy",
  "last_name": "Miller"
}
```

Adding custom summary info:

```
PATCH /api/v1/customers/51f7a7
Content-Type: application/json
```

```
{
  "summary": "VIP customer"
}
```

Creating a new customer

```
POST /api/v1/customers
```

Field	Required	Notes
first_name	no	
last_name	no	
summary	no	

Sample Request:

```
POST /api/v1/customers
Content-Type: application/json
```

```
{
  "first_name": "John",
  "last_name": "Smith"
}
```

```
201 Created
Location: https://site.enchant.com/api/v1/customers/51c3d1
Content-Type: application/json
```

```
{
  "id": "51c3d1",
  "first_name": "John",
  "last_name": "Smith",
  "summary": null
}
```

Contacts

Email addresses and twitter accounts are represented as contacts on a customer. They are automatically loaded with customer records

Field	Type	Notes
id	string	
type	string	One of: `email`, `twitter`
value	string	

Creating a contact

```
POST /api/v1/customers/customer_id/contacts
```

Field	Required	Notes
type	yes	One of: `email`, `twitter`
value	yes	

Sample Request:

```
POST /api/v1/customers/51f846
Content-Type: application/json
```

```
{
  "type": "email",
  "value": "john@example.com"
}
```

```
201 Created
Location: https://site.enchant.com/api/v1/customers/514846/contacts/5197cc
Content-Type: application/json
```

```
{
  "id": "5197cc",
  "type": "email",
  "value": "john@example.com"
}
```