Example API documentation version 1

http://example.com/1

Welcome

Welcome to the Example Documentation. The Example API allows you to do stuff. See also example.com.

```
var raml2html = require('raml2html');

// Using the default templates:
// source can either be a filename, file contents (string) or parsed RAML object
raml2html.parse(source, onSuccess, onError);

// Using your own templates:
// - config should be an object with at least an `template` property
// - config can also include `helpers` and `partials`
// - the config object will be accessible from your handlebars templates
raml2html.parseWithConfig(source, config, onSuccess, onError);
```

Chapter two

More content here. Including **bold** text!

Small table:

A	B	C
1	2	3

Done

ACCOUNTS

This is the top level description for /account.

- One
- Two
- Three

/account

POST

POST /account

Creates a new account. Some **bold** text here. More text. Need to fill the line, so make it longer still. Hooray! Line two

Paragraph two

Request

Body
Type: application/json

Example:

```
{
  "email": "john@example.com",
  "password": "super_secret",
  "name": "John Doe"
}
```

Response

HTTP status code 200

Account was created and user is now logged in

/account/find

GET

GET /account/find

find an account

Request

Query Parameters

- **name**: *required (string)*
  name on account

  Example:

  ```
  Naruto Uzumaki
  ```

- **gender**: *required (one of male, female)*
- **number**: *(integer - default: 42)*

/account/{id}

GET PUT DELETE

GET /account/{id}

Request

URI Parameters

- **id**: *required (string - minLength: 1 - maxLength: 10)*
  account identifier

Headers

- **Authorization**: *(string)*
  Basic authentication header

  Example:

  ```
  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
  ```

PUT /account/{id}

Update the account

Request

URI Parameters

- **id**: *required (string - minLength: 1 - maxLength: 10)*
  account identifier

Body
Type: application/x-www-form-urlencoded

Form Parameters

- **name**: *(string)*
  name on account

  Example:

  ```
  Naruto Uzumaki
  ```

- **gender**: *(one of male, female)*

DELETE /account/{id}

Delete the account

Request

URI Parameters

- **id**: *required (string - minLength: 1 - maxLength: 10)*
  account identifier

/account/login

POST

POST /account/login

Login with email and password

Request

Body
Type: application/json

Example:

```
{
  "email": "john@example.com",
  "password": "super_secret"
}
```

Response

HTTP status code 200

Login was correct

Body
Type: text/xml

Example:

```
<test>This is a test</test>
```

HTTP status code 400

Login was incorrect, please try again

HTTP status code 401

Not authorized

Headers

- **WWW-Authenticate**: *(string)*
  user was not authorized

  Example:

  ```
  WWW-Authenticate: Basic realm="raml2html"
  ```

/account/forgot

POST

POST /account/forgot

Sends an email to the user with a link to set a new password

Response

HTTP status code 200

Test

Body
Type: text/xml

Example:

```
<test>This is a test</test>
```

/account/session

GET DELETE

GET /account/session

Gets the sessions

DELETE /account/session

Deletes the session, logging out the user

Forecasts

The very top resource - displays OK

/forecasts/{geoposition}

GET

Overview endpoint to assemble and access forecast data in various timely resolutions - THIS IS NOT DISPLAYED ANYWHERE WITH RAML2HTML :/

GET /forecasts/{geoposition}

Provides an overview of the available data - display OK

Request

URI Parameters

- **geoposition**: *required (string)*
  A geoposition aquired by calling /geoposition/search - displays OK

/forecasts/test

No methods here, but it does have a description

/conversations

This is the top level description for /conversations.

/conversations

GET POST

GET /conversations

Get a list of conversation for the current user

POST /conversations

Create a new conversions. The currently logged in user doesn't need to be supplied in the members list, it's implied.

Request

Body
Type: application/json

Example:

```
{
  "content": "My message!",
  "members": [1, 2, 3]
}
```

Response

HTTP status code 200

A conversation with these members already existed, the message was added to that one

HTTP status code 201

The conversation was created and the message added to it

/conversations/{convId}

GET PUT

GET /conversations/{convId}

Get a single conversation including its messages

Request

URI Parameters

- **convId**: *required (string)*

PUT /conversations/{convId}

Update a conversation (change members)

Request

URI Parameters

- **convId**: *required (string)*

/conversations/{convId}/messages

GET POST

GET /conversations/{convId}/messages

Get the messages for the conversation

Request

URI Parameters

- **convId**: *required (string)*

Headers

- **TESTING**: *(string)*
  does a trait render its headers?

Query Parameters

- **page_size**: *(number - default: 20)*
  The number of items per page
- **page**: *(number - default: 0)*
  The page to return

POST /conversations/{convId}/messages

Add a new message to a conversation

Request

URI Parameters

- **convId**: *required (string)*

/conversations/{convId}/messages/{messageId}

PUT DELETE

PUT /conversations/{convId}/messages/{messageId}

Update the message

Request

URI Parameters

- **convId**: *required (string)*
- **messageId**: *required (string)*

DELETE /conversations/{convId}/messages/{messageId}

Delete the message

Request

URI Parameters

- **convId**: *required (string)*
- **messageId**: *required (string)*

/users

/users

GET POST

GET /users

Get a list of all users

Request

Headers

- **TESTING**: *(string)*
  does a trait render its headers?

Query Parameters

- **page_size**: *(number - default: 20)*
  The number of items per page
- **page**: *(number - default: 0)*
  The page to return
- **from**: *(string - pattern: ^[a-zA-Z].+$)*
  Limit results to those created after from.

  Example:

```
2014-12-31T00:00:00.000Z
```

POST /users

Creates a new user

Request

Body
Type: application/xml

Example:

```
<h1>Hello!</h1>
```

/users/{userId}

GET PUT DELETE

GET /users/{userId}

Get the details of a user including a list of groups he belongs to

Request

URI Parameters

- **userId**: *required (string)*

PUT /users/{userId}

Update a user

Request

URI Parameters

- **userId**: *required (string)*

DELETE /users/{userId}

Deletes a user

Request

URI Parameters

- **userId**: *required (string)*

/groups

/groups

GET POST

GET /groups

Get a list of all the groups

POST /groups

Create a new group

Request

Body
Type: application/json

Example:

```
{
  "name": "Cool people",
  "members": [1, 2, 3]
}
```

/groups/{groupId}

GET PUT DELETE

GET /groups/{groupId}

Get the details of a group, including the member list

Request

URI Parameters

- **groupId**: *required (string)*

PUT /groups/{groupId}

Update the group, **optionally** supplying the new list of members (overwrites current list)

Request

URI Parameters

- **groupId**: *required (string)*

Body
Type: application/json

Example:

```
{
  "name": "Cool people",
  "members": [1, 2, 3]
}
```

DELETE /groups/{groupId}

Removes the group

Request

URI Parameters

- **groupId**: *required (string)*

/groups/{groupId}/users **POST**

POST /groups/{groupId}/users

Adds a user to a group

Request

URI Parameters

- **groupId**: *required (string)*

Body
Type: application/json

Example:

```
{
  "user_id": 4,
}
```

/groups/{groupId}/users/{userId} **DELETE**

DELETE /groups/{groupId}/users/{userId}

Removes a user from a group

Request

URI Parameters

- **groupId**: *required (string)*
- **userId**: *required (string)*