# Note

This file has been created by turning the markdown source into a human friendly format.This file may be out of date with the current versions in the source repository.As a result, internal links in the document will not work correctly.Instead the links documents have been compiled into separate documents

|  |  |
| --- | --- |
| Document | Description |
| protocol.docx | The compiled protocol files |
| scenarios.docx | The compiled scenarios files |
| charter.docx | The compiled charter files |
| review.docx | The compiled review guide files |

This document has been compiled with pandoc. docx is used because the PDF version doesn't get the tables correctly.The authoritative source for these documents is online at GitHub which is the preferred way for you to view and comment on them<https://github.com/wufm/osdi-docs>

For questions, please contact [joshco@gmail.com](mailto:joshco@gmail.com)

# Open Supporter Data Interface

The Open Supporter Data Interface (OSDI) effort seeks to define an API and data structures for interoperability among products in the cause-based, campaign and non-profit marketplace. The existence of a common API will reduce customer costs related to moving data between different systems, lower integration costs and enhance the ability of innovators to create products for the marketplace.

The API will define interfaces including but not limited to resources representing people, donations, questions, tags, and events. The group will determine the order in which to define resource models and which version of the API to include them in.

[Read the full charter](charter.md)

[Read our scenarios document](scenarios/scenarios.md)

[Experiment with our prototype](http://api.opensupporter.org)

Please give us feedback on our work. [Read the Review Guide](review_guide.md) to learn what kind of feedback we're looking for and how to provide it.

# Authors

* Tim Anderegg, New Organizing Institute (NOI)
* Topper Bowers, Amicus
* Josh Cohen, Washington United For Marriage (Editor)
* Charles Parsons, Salsa Labs
* Ray Suelzer, UFCW International Union

This effort is currently in an exploratory phase to determine if consensus on a common API can be achieved. The involvement of a person or company does not reflect a commitment to implement this API.

# API Data Model

* [API Entry Point](aep.md)
* [People and Addresses](people.md)
* [Survey Questions and Answers](questions.md)
* Events
* Donations
* [Canvassing Interactions](interactions.md)

# Basic Resource Access

## API Entry Point and linking

All access through OSDI starts at the API Entry Point (AEP). The AEP is a resource that acts like a directory of the types of resources available on a server. It also includes capability information like the maximum query pagesize.

Some servers may support some or all of the different resource collections. For example, a peer to peer donation system might support Donations and People but not events. In order to find out what resources are available and what URIs to use to access them, do a GET on the AEP URI.

Your service provider can tell you what the AEP URI is for your account.

For the purposes of example, assume your provider has given you an AEP URI of

[<http://api.opensupporter.org/api/v1>](http://api.opensupporter.org/api/v1)

Note: you can explore the AEP with a user friendly interface by visiting our [prototype endpoint](http://api.opensupporter.org)

In order to determine the available resources on the server the client should perform an HTTP GET request to this URI.

Within the response will be a collection of links to the resource collections available on the server.

Request

GET /api/v1 HTTP/1.1Host: api.opensupporter.org

Response

200 OKContent-Type: application/json{ "motd": "Welcome to the ACME Action Platform OSDI API endpoint!!", "\_links": { "people": { "href": "/api/v1/people", "title": "The collection of people in the system" }, "addresses": { "href": "/api/v1/addresses", "title": "The collection of addresses in the system" }, "questions": { "href": "http://api.opensupporter.org/api/v1/questions", "title": "The collection of questions in the system" }, "question\_answers": { "href": "http://api.opensupporter.org/api/v1/question\_answers", "title": "The collection of question answers in the system" }, "self": { "href": "/api/v1", "title": "The root API Entry Point (Your are here)" }, "docs": { "href": "https://github.com/wufm/osdi-docs", "title": "Documentation:", "name": "Docs", "index": "index" } }}

Given the above example response, let's fetch the people collection on this server.Notice the "\_links" collection. Find the object in the links collection with key "people". That object has an attribute "href" which contains the URI to use to access the people collection.

This is for example purpose only. The official definition of the person schema is [People and Addresses](people.md)

Request

GET /api/v1/people HTTP/1.1

Response

{ "total\_records": 80, "total\_pages": 16, "page" : 2, "\_links": { "next" : { "href" : "http://api.opensupporter.org/api/v1/people?page=3&per\_page=5" }, "previous" : { "href" : "http://api.opensupporter.org/api/v1/people?page=1&per\_page=5" } }, "\_embedded": { "people": [ { "first\_name": "Edwin", "last\_name": "Labadie", "middle\_name": "Marques", "email": "test-3@example.com", "gender": "Male", "sex": "Female", "party": "Democrat", "source": "sed", "source\_details": "Delectus rerum autem mollitia sit asperiores odit hic cum.", "twitter\_handle": "@Edwin\_Labadie", "guid": "c1d9c510-b562-0130-dc7c-168c51e904de", "\_embedded": { "primary\_address": { "address1": "935 Ed Lock", "city": "New Dudley", "state": "MN", "postal\_code": "17678", "country\_code": "RU", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses/46" }, "person": { "href": "http://api.opensupporter.org/api/v1/people/23" } } }, "addresses": [ { "address1": "28160 Wiegand Divide", "city": "Lake Amarimouth", "state": "GA", "postal\_code": "27585-7257", "country\_code": "US", "address\_type": "Work", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": false, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses/45" }, "person": { "href": "http://api.opensupporter.org/api/v1/people/23" } } }, { "address1": "935 Ed Lock", "city": "New Dudley", "state": "MN", "postal\_code": "17678", "country\_code": "RU", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses/46" }, "person": { "href": "http://api.opensupporter.org/api/v1/people/23" } } } ] }, "\_links": { "addresses": { "href": "http://api.opensupporter.org/api/v1/people/23/addresses" }, "question\_answers": { "href": "http://api.opensupporter.org/api/v1/people/23/question\_answers" }, "self": { "href": "http://api.opensupporter.org/api/v1/people/23" } } } .... other person records follow } }

In the last example message, the server returns a list of people. For brevity this document only shows the first one. Within each person object, there is also a "\_links" collection just like in the AEP. This will show up in most objects in OSDI. The links collection lets the client know what other resources and resource collections are associated with a given object.

In this example, the link shown is "addresses". The href attribute of the "addresses" link contains the URI of the address collection *for this person*.

"\_links": { "addresses": { "href": "http://api.opensupporter.org/api/v1/people/23/addresses" }

A client can send a GET request to this URI to retrieve a list of addresses associated with this person.

GET /api/v1/people/23/addresses200 OKContent-Type: application/json{ "total\_pages": 1, "page": 1, "total\_records": 2, "\_embedded": { "addresses": [ { "address1": "28160 Wiegand Divide", "city": "Lake Amarimouth", "state": "GA", "postal\_code": "27585-7257", "country\_code": "US", "address\_type": "Work", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": false, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses/45" }, "person": { "href": "http://api.opensupporter.org/api/v1/people/23" } } }, { "address1": "935 Ed Lock", "city": "New Dudley", "state": "MN", "postal\_code": "17678", "country\_code": "RU", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses/46" }, "person": { "href": "http://api.opensupporter.org/api/v1/people/23" } } } ] }, "\_links": { "self": { "href": "http://api.opensupporter.org/api/v1/addresses" } }}

Note that this pattern can be applied to other associated collections including but not limited to donations or question\_answers.

## HAL

OSDI has embraced the [JSON+HAL spec](http://tools.ietf.org/html/draft-kelly-json-hal-05). JSON+HAL specifies a simple way to embed linking into APIs. The combination of linking and a specification allows generic clients to be written and, indeed, [many languages have HAL clients](http://stateless.co/hal_specification.html). Linking itself makes it easier to both reason about and write clients for an API.

By default, server responses should expand first level instances unless otherwise specified. For example, in a response for a collection of resources, those resources should be embedded.

## Common CRUD operations

### Creating a Resource

Creating a new resource involves adding a new item to a collection. To create a new resource, an HTTP POST message is sent to the URI for a collection.

POST <addURI> HTTP/1.1Host: ...Accept: application/...Content-Type: application/...<serialization of request to create a new resource>HTTP/1.1 201 CreatedLocation: ...Content-Type: application/...<serialization of new resource>

#### Insert or Update (Upsert)

In some cases, the client doesn't know if a resource exists or not. Instead of first having to query a resource to determine if it exists and then do an update via PUT, the client may use the upsert feature.

When used, the server will use a matching algorithm to determine if the input attributes match an existing record.

The algorithm used by the server to perform matching is vendor-specific. Contact your vendor for specifics.

To use the upsert feature, the $upsert query parameter is appended to the URI. Its value is either true or false

If the upsert parameter is not included, it defaults to TRUE. The server will attempt a match to an existing resource first, but if if cannot find one, a new resource will be created.

POST <addURI>?upsert=false HTTP/1.1Host: ...Accept: application/...Content-Type: application/...<serialization of potential new resource>

If the resource does not exist, then a 201 response is returned

HTTP/1.1 201 CreatedLocation: ...Content-Type: application/...<serialization of new resource>

If the resource does already exist, then a 200 response is returned

HTTP/1.1 200 OKLocation: ...Content-Type: application/...<serialization of existing resource>

### Retrieving a Resource

Retrieving a resource gets a representation of a resource instance or resource collection. The retrieval is performed with an HTTP GET sent to the URI of the resource.

Request

GET <ResourceURI> HTTP/1.1Host: ...Accept: application/...

Response

HTTP/1.1 200 OKContent-Type: application/...<serialization of resource>

#### Collection Responses

When retrieving collections, the response representation will include some common attributes.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| total\_pages | integer | The number of pages applicable to this query |
| total\_records | integer | The total number of resources matching this query |
| page | integer | The page number of this response |

##### Prev/next

Collection responses may include additional links for navigation to previous and next pages

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| next |  | the link for the next page of results |
| previous |  | the link for the previous page of results |

### Updating a Resource

Updating a resource instance is accomplished by the use of an HTTP PUT sent to the URI of a given resource. Due to the complexity of full-resource updates involving read-only properties, out-of-date data, and the need to know all properties (which one may not), this specification focuses on the ability to make partial updates to resources.

To make an update to a resource, the client sends an HTTP PUT to the URI of a resource instance. The body of the put contains a partial resource representation including the attributes to update. Missing attributes are left unchanged on the Provider side.

Clients may set an attribute to nil by including the attribute using ‘nil’ for JSON.

Updating Collections with PUT is not supported.

PUT <editURI> HTTP/1.1Host: ...Accept: application/...Content-Type: application/...<serialization of request to update a resource>HTTP/1.1 200 OKContent-Type: application/...<serialization of updated resource>The HTTP response body shall contain the serialization of the updated resource

## Selecting Results

### Filtering Collections

When retrieving representations of a collection, clients may include filters expressed as query parameters. The $filter query parameter is used for this purpose. The $filter parameter value shall contain an expression using the following operators:

'<', '<=', '=', '>=', ">', '!=' : Integer and date value/attribute '=', '!=' : String value/attribute

Example:Assuming a resource of ‘Person’ which has an attribute named ‘firstName’, the following filter would return resources with first name of ‘Jon’

$filter=first\_name=’Jon’

### Pagination

The parameters $per\_page and $page control pagination.

* $per\_page specifies how many results to return per page
* $page specifies the starting page to start with.

### Expand / Mixins

In order to optimize access, the $expand query parameter can be used to expand collections within resources. Normally when retrieving a resource instance, subordinate collections are returned as references.

When the $expand parameter is used, the collection corresponding to the parameter value shall be expanded to represent the collection of instances.

When this parameter is used in requests for Collections, the value applies to the resources enclosed in the collection rather than the collection itself.

Example

Assuming a resource of Person with a collection of addresses, $expand=addresses would cause the collection of instances of addresses to be returned rather than a reference.

## Notational Conventions

In this specification, when defining models, the following notational conventions are used.

|  |  |
| --- | --- |
| Convention | Description |
| type[] | A reference to a collection of resources of type 'type' |
| type\* | A reference to a single resource of type 'type' |
| string | A string |
| datetime | A date and time representation. In JSON this is a string. The contents of this attribute shall be ISO 8601 |

In the description of string types, sometimes the specification will list a set of acceptable values such as

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| gender | string | one of "Male", "Female", "Other" |

In these cases, the string value should conform to one of the choices unless specified otherwise

# API Entry Point

## Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| max\_pagesize | string | a number that defines how many records a server can return in a single query. |
| motd | string | an informational message from the server |
| people | Person[] | A reference to the Persons Collection |
| events | Event[] | A reference to the Events Collection |
| attendances | Attendance[] | A reference to the Attendance Collection |
| donations | Donation[] | A reference to the Donation collection |
| questions | Question[] | A reference to the Question collection |
| question\_answers | QuestionAnswer[] | A reference to the questionAnswer collection. |
| Interactions | Interaction[] | A reference to the Interaction collection |

# Retrieving the API endpoint

## Request

GET /api/v1 HTTP/1.1

## Response

200 OKContent-Length: nnContent-Type: application/json{ "motd": "Welcome to the ACME Action Platform OSDI API endpoint!!", "\_links": { "people": { "href": "/api/v1/people", "title": "The collection of people in the system" }, "addresses": { "href": "/api/v1/addresses", "title": "The collection of addresses in the system" }, "questions": { "href": "http://osdi-prototype.herokuapp.com/api/v1/questions", "title": "The collection of questions in the system" }, "question\_answers": { "href": "http://osdi-prototype.herokuapp.com/api/v1/question\_answers", "title": "The collection of question answers in the system" }, "self": { "href": "/api/v1", "title": "The root API Entry Point (Your are here)" }, "http://localhost:3000/Readme.md": { "href": "http://localhost:3000/Readme.md", "title": "Documentation:", "name": "Docs", "index": "index" } }}

# Person

## Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| last\_name | string | Last name |
| first\_name | string | First name |
| middle\_name | string | Middle name |
| sex | string | A person's biological sex, one of "Male", "Female", "Other" |
| gender | string | The gender with which a person identifies |
| guid | string | A globally unique identifier (GUID) for a person |
| party | string | Party affiliation "democrat", "republican", "independent", "none" |
| primary\_address | Address | A single instance of an address reflecting the person's primary address |
| primary\_phone | string | The person's primary phone number |
| primary\_email | string | A person's primary email address |
| source | string | Information about the source where this person record was acquired. Eg "Ref74" |

## Collections

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| addresses | Address[] | A collection of addresses associated with the person |
| emails | Email[] | A collection of email addresses associated with the person |
| phones | Phone[] |
| donations | Donation[] | A collection of donations associated with the person |
| question\_answers | QuestionAnswer[] | A collection of answers to questions from surveys |
| event\_attendance | Attendance[] | A collection of attendance records for a person |
| interactions | Interaction[] | A collection of outreach interactions for a person, eg Volunteer Joe called voter Sam F. Bar |

# Address

## Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| primary | boolean | Denotes if this is the primary address. A person can have only one primary address |
| address\_type | string | Type of address "Home","Work",'Mailing' |
| address1 | string | Address line 1 |
| address2 | string | Address line 2 |
| addressN | string | Additional address lines |
| city | string | City |
| state | string | State abbreviation according to ISO 3166-2 (Final 2 alpha digits) |
| postal\_code | string | Region specific postal code |
| country\_code | string | Country code according to ISO 3166-1 Alpha-2 |
| lat | float | Geolocation latitude |
| lng | float | Geolocation longitude |
| accuracy | string | One of "Rooftop", "Approximate" |
| address\_status | string | One of "Potential", "Verified", "Bad". |

### State and Country codes

Country Codes should conform to [ISO 3166-1 Alpha-2](http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2)

Examples:

|  |  |
| --- | --- |
| Country | Code |
| United States | US |
| Canada | CA |
| Cyprus | CY |

In the United States, state abbreviations should conform to [ISO 3166-1 Alpha-2](http://en.wikipedia.org/wiki/ISO_3166-2:US) but using only the final two alphanumeric characters

Examples:

|  |  |
| --- | --- |
| State | Code |
| New Jersey | NJ |
| California | CA |
| New York | NY |
| Washington | WA |

## Links

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| person | Person\* | A link to the person associated with the address |

# Scenarios

## Get a list of people with pagination

GET /api/v1/people?per\_page=2&page=1200 OKContent-Type: application/json{ "total\_pages": 1, "page": 1, "total\_records": 2, "\_embedded": { "people": [ { "first\_name": "Edwin", "last\_name": "Labadie", "middle\_name": "Marques", "email": "test-3@example.com", "gender": "Male", "sex": "Female", "party": "Democrat", "source": "sed", "source\_details": "Delectus rerum autem mollitia sit asperiores odit hic cum.", "twitter\_handle": "@Edwin\_Labadie", "guid": "c1d9c510-b562-0130-dc7c-168c51e904de", "\_embedded": { "primary\_address": { "address1": "935 Ed Lock", "city": "New Dudley", "state": "MN", "postal\_code": "17678", "country\_code": "RU", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/46" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23" } } }, "addresses": [ { "address1": "28160 Wiegand Divide", "city": "Lake Amarimouth", "state": "GA", "postal\_code": "27585-7257", "country\_code": "US", "address\_type": "Work", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": false, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/45" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23" } } }, { "address1": "935 Ed Lock", "city": "New Dudley", "state": "MN", "postal\_code": "17678", "country\_code": "RU", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/46" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23" } } } ] }, "\_links": { "addresses": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23/addresses" }, "question\_answers": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23/question\_answers" }, "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/23" } } }, { "first\_name": "Parker", "last\_name": "Walker", "middle\_name": "Jannie", "email": "test-4@example.com", "gender": "Male", "sex": "Female", "party": "Democrat", "source": "architecto", "source\_details": "Itaque et reprehenderit rerum ea quis.", "twitter\_handle": "@Parker\_Walker", "guid": "c1e1d0d0-b562-0130-dc7c-168c51e904de", "\_embedded": { "primary\_address": { "address1": "22184 Vernie Cove", "city": "Rowemouth", "state": "GA", "postal\_code": "74895", "country\_code": "JP", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/48" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24" } } }, "addresses": [ { "address1": "7485 Rashad Pine", "city": "Brandynview", "state": "PR", "postal\_code": "76221-3163", "country\_code": "US", "address\_type": "Work", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": false, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/47" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24" } } }, { "address1": "22184 Vernie Cove", "city": "Rowemouth", "state": "GA", "postal\_code": "74895", "country\_code": "JP", "address\_type": "Home", "lat": 44, "lng": 40, "accuracy": "Rooftop", "address\_status": "Verified", "primary": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses/48" }, "person": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24" } } } ] }, "\_links": { "addresses": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24/addresses" }, "question\_answers": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24/question\_answers" }, "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people/24" } } } ] }, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/people" }, "addresses": { "href": "http://osdi-prototype.herokuapp.com/api/v1/addresses" } }}

# Survey Questions and Answers

## Question

### Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| name | string | Name for the question |
| question | string | Human readable text of the question |
| question\_type | string | One of "MultiChoice" |
| responses | QuestionResponse[] | A collection of possible responses |

## QuestionResponse

A possible response or choice for a question.

### Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| name | string | Human readable text of the value |
| value | string | Actual value |
| default | boolean | True if this response should be the default response |

## QuestionAnswer

An answer to a question. An answer is chosen by the user from one of the QuestionResponse options

### Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| value | string | Human readable text of the value |
| person\_id | integer | ID of associated person object |
| question\_id | integer | ID of associated question object |
| question | Question\* | Reference to associated question |
| person | Person\* | Reference to associated person |

## Retrieving Available Questions

Request

GET /api/v1/questions HTTP/1.1

Response

200 OKContent-Type: application/json{ "\_embedded": { "questions": [ { "name": "Best Editor", "question": "What is the best editor?", "question\_type": "MultiChoice", "id": 1, "\_embedded": { "question\_responses": [ { "name": "Emacs", "value": "Emacs", "default": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/question\_responses/1" } } }, { "name": "Vi", "value": "Vi", "default": false, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/question\_responses/2" } } } ] }, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/questions/1" } } }, { "name": "Marriage Equality", "question": "Gays should have the right to Marry?", "question\_type": "MultiChoice", "id": 2, "\_embedded": { "question\_responses": [ { "name": "Strongly Support", "value": "5", "default": true, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/question\_responses/3" } } }, { "name": "Strongly Oppose", "value": "1", "default": false, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/question\_responses/4" } } } ] }, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/questions/2" } } }, { "name": "Bio", "question": "Tell us about yourself", "question\_type": "Paragraph", "id": 3, "\_embedded": { "question\_responses": [] }, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/questions/3" } } } ] }, "\_links": { "self": { "href": "http://osdi-prototype.herokuapp.com/api/v1/questions" } }}

## Creating a QuestionAnswer

Request

POST /api/v1/people/question\_answers HTTP/1.1Content-Type: application/json{ "value" : "Emacs", "question\_id" : 5}

Response

# Interactions

Interactions are instances of one person engaging another person. An example of this would be as follows. Alice makes a telephone call to Bob. On that call she asks Bob a question and records his response. The Interaction is primarily associated with Alice via the person attribute as she is the one who took action. Bob answered the question and is considered the prospect. The Interaction is secondarily associated with Bob via the prospect attribute.

# Attributes

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| canvasser | Person\* | Reference to the person who completed this interaction, eg the person who asked the question |
| prospect | Person\* | Reference to the person the canvasser interacted with. eg, the person who was asked the question |
| attempt\_type | string | One of "Call", "Email","Canvass","Web", "Other" |
| attempt\_result | string | One of "Completed", "Unreachable", "Left Message" |
| donations | Donation[] | Associated donations |
| attendances | Attendance[] | Associated event attendances |
| question\_answers | QuestionAnswer[] | Associated answered questions |
| foos | Foo[] | There may be additonal collections of objects associated with an interaction |