Protocol

Dycapo Protocol is an open protocol for sharing trip data among dynamic transit services. It is currently in the first stages of development.

It is an JSON RESTful protocol, heavily inspired by OpenTrip (http://opentrip.info/) Core protocol.

Contents

Introduction

Dycapo Protocol is an application-level protocol for enabling communication between Dynamic Carpooling servers and clients, using HTTP [RFC2616] and JSON [RFC4627].

Where to start

To better understand OpenTrip Dynamic, you should first read the OpenTrip Core (http://opentrip.info/wiki/OpenTrip_Core) specification, as we are inspiring from it. After that, we may summarize Dycapo Protocol as “OpenTrip entities extended and encoded in JSON”. That is, we took all OpenTrip Core entities as described in the draft, created a convenient UML class diagram for developing Dycapo and extended the entities to suit our needs. That is how Dycapo Protocol is coming out.

Specification

Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 (http://tools.ietf.org/html/rfc2119) .

Each attribute requirement is specified when submitting objects to the system, not when returning them to the clients. Some attributes MAY be hidden when Resources are accessed because of privacy issues, e.g. the actual position of a user. This goes beyond the scopes of this Protocol and depends on the implementation of the server system.

JSON-Related Conventions

Dycapo Protocol Document formats are specified in terms of the JavaScript Object Notation [rfc4627].

Every data type is a JSON data type, as described in JSON specs (http://www.ietf.org/rfc/rfc4627.txt) .

Other Data-Type Conventions

The protocol makes use of GeoRSS Simple Point (http://georss.org/GeoRSS_Simple#Point) notation for expressing Geographical coordinates, encapsulated in a JSON string.

Date/Time objects are expressed using JSON string data type, using [ISO 8601] Convention. The following is the preferred format: "YYYY-MM-DD HH:MM:SS", as example: "2010-08-21 21:36:23"

Terminology

For convenience, this protocol can be referred to as the "Dycapo Protocol" or "DycapoP". The following terminology is used by this specification:

- URI - A Uniform Resource Identifier as defined in [RFC3986]. In this specification, the phrase "the URI of a document" is shorthand for "a URI which, when dereferenced, is expected to produce that document as a representation".
- IRI - An Internationalized Resource Identifier as defined in [RFC3987]. Before an IRI found in a document is used by HTTP, the IRI is first converted to a URI. See Section 4.1.
- Resource - A network-accessible data object or service identified by an IRI, as defined in [RFC2616]. See [REC-webarch] for further discussion on Resources.
- Object - An unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, boolean, null, object, or array. Defined in [rfc4627]
- Array - An ordered sequence of zero or more values, as in [rfc4627]
- Driver - the role assumed by a user when he/she offers a Trip and drives a vehicle.
- Passenger - the role assumed by a user when he/she searches for a ride. A user which is not a Driver is automatically considered a Passenger.
- Trip - a single journey or course of travel taken as part of one's duty, work, etc. A Driver offers Trips. In DycapoP, a Trip is composed by some simple attributes described below plus a mode, a preferences, more than two locations.
- Location - a place of settlement, activity, or residence.
- Mode - a description about the mode of transportation being used by the Driver when performing a Trip.
- Preferences - a description about the preferences of a Driver when performing a Trip.
- Participation - the fact of taking part, as in some action or attempt, in a Trip. Both a Driver and Passengers participate in a Trip.

Protocol Model

DycapoP specifies operations for publishing, editing and deleting specific Resources using HTTP. It uses JSON-formatted representations to describe the state and metadata of those Resources.

Objects as inner Properties of other Objects

Some DycapoP objects can be obtained as Objects alone or be included in other objects.

As example, a Person object has a *location* attribute, that holds the current Person position.

When a DycapoP object is returned as an inner property of another object, just the *href* attribute MUST be included. All the other properties MAY be included.

The href attribute

Each Protocol object MUST include an attribute called **href** when returned as a resource. The type of this attribute is a string and its value MUST be the URL uniquely identifying the object.

For each Object a URL structure is proposed for sake of comprehension. It SHOULD NOT be followed as it is against REST principles.

The author attribute

Some Protocol objects have an attribute called **author**. This attribute specifies the Person that created the resource. The requirement of this attribute is always a MAY because its presence depends on the internal implementation of the security systems of the server implementing the protocol. Therefore, its value is usually filled in by the server after an authentication happened.

Initial URI

Dycapo Protocol attempts to be as more REST as possible. Therefore, it defines resources as hypertext driven. Clients SHOULD use an inital URI, here defined as **http://example.com/api". Each other access SHOULD be performed using the** href **attribute of each object.**

A GET request to the initial URI should return a list of accessible resources. As example:

```
{
    "searches": {
        "href": "http://example.com/api/searches/"
    },
    "persons": {
        "href": "http://example.com/api/persons/"
    },
    "trips": {
        "href": "http://example.com/api/trips/"
    }
}
```

Elements

Location

Represents a single location. See OpenTrip_Core#Location_Constructs (http://opentrip.info/wiki/OpenTrip_Core#Location_Constructs) for more info.

Attribute	Type	Requirement
label	string	MAY
street	string	MUST*
point	string	MUST
country	string	MAY
region	string	MAY
town	string	MUST*
postcode	number	MUST*
subregion	string	MAY
georss_point	string	MUST*
offset	number	SHOULD
recurs	string	MAY
days	string	MAY
leaves	string (see Dates)	MUST
href	string	MUST NOT

- Either **georss_point** OR all from set {**street**,**town**,**postcode**} MUST be specified
- **point** value MUST be any from the set {**orig**, **dest**, **wayp**, **posi**}.

See OpenTrip_Core#Attributes (http://opentrip.info/wiki/OpenTrip_Core#Attributes) for more info. **posi** is an extension and is for indicating that the Location represents the current position of a Person.

Data Representation Example

The following is a valid Location object:

```
{
                "town": "Bolzano",
                "point": "orig",
                "country": "",
                "region": "",
                "subregion": "",
                "days": "",
                "label": "Work",
                "street": "Rom Strasse",
                "postcode": 39100,
                "offset": 150,
                "leaves": "2010-09-02 13:32:34",
                "recurs": "",
                "georss_point": "46.490200 11.342294"
}
```

Operations

URL	http://example.com/persons/[username]/location/
Method	GET
Description	Returns a Person position

URL	http://example.com/persons/[username]/location/
Method	POST
Request Body	Location
Description	Updates (or creates) a Person's position

URL	http://example.com/persons/[username]/location/
Method	PUT
Request Body	Location
Description	Updates (or creates) a Person's position

URL	http://example.com/trips/[id]/locations/
Method	GET
Description	Returns the Locations involved in a Trip

URL	http://example.com/trips/[trip_id]/locations/[location_id]/
Method	GET
Description	Returns a Location involved in a Trip

Person

Represents a Person as described on OpenTrip_Core#Person_Constructs (http://opentrip.info/wiki/OpenTrip_Core#Person_Constructs)

Attribute	Type	Requirement
username	string	MUST
email	string	MUST
first_name	string	SHOULD
last_name	string	SHOULD
uri	string	MAY
phone	string	SHOULD
location	object (Location)	MUST NOT
age	number	SHOULD
gender	string	SHOULD
smoker	boolean	MAY
blind	boolean	SHOULD
deaf	boolean	SHOULD
dog	boolean	SHOULD
href	string	MUST NOT

Data Representation Example

The following is a valid Dycapo Protocol Person:

```
{
    "username": "driver1",
    "gender": "M",
    "phone": "123456",
    "email": "driver@drivers.com"
}
```

Operations

URL	http://example.com/persons/
Method	GET
Description	Retrieves a collection of Persons

URL	http://example.com/persons/
Method	POST
Request Body	Person
Description	Creates a Person resource

URL	http://example.com/persons/[username]/
Method	GET
Description	Retrieves a Person

URL	http://example.com/persons/[username]/
Method	PUT
Request Body	Person
Description	Updates a Person object

Modality

Represents additional information about the mode of transportation being used. See OpenTrip_Core#Mode_Constructs (http://opentrip.info/wiki/OpenTrip_Core#Mode_Constructs) for more info.

Attribute	Type	Requirement
kind	string	MUST
capacity	number	MUST
vacancy	number	MUST
make	string	MUST
model_name	string	MUST
year	string	MAY
color	string	SHOULD
lic	string	SHOULD
cost	number	SHOULD
href	string	MUST NOT

- Please use as **capacity** the total capacity of your car MINUS the driver. E.G. If a car has a capacity of 5 seats, use 4 as value for capacity.

Data Representation Example

The following is a valid DycapoP Modality object:

```
{
        "kind": "auto",
        "capacity": 4,
        "lic": "",
        "color": "",
        "make": "Ford",
        "vacancy": 4,
        "cost": 0.0,
        "year": 0,
        "model_name": "Fiesta",
}
```

Operations

URL	http://example.com/trips/[trip_id]/modality/
Method	GET
Description	Returns the Modality object of that Trip

Preferences

Stores the preferences of a Trip set by the Person who creates it. See OpenTrip_Core#Preference_Constructs (http://opentrip.info/wiki/OpenTrip_Core#Preference_Constructs) for more info. We kept drive and ride attributes just for compatibility reasons: in OpenTrip Dynamic just a driver should be the author of a Trip.

Attribute	Type	Requirement
age	string	MAY
nonsmoking	boolean	MAY
gender	string	MAY
drive	boolean	MAY
ride	boolean	MAY
href	string	MUST NOT

- Even if all attributes of Prefs objects are optional, objects of type Prefs MUST be provided when doing an operation that involves this object. In case of zero attributes provided, an empty object MUST be provided
- **gender** MUST be any of the values {'M', 'F', 'B'}, meaning 'male', 'female', 'both'

Data Representation Example

The following is a valid DycapoP Prefs object:

```
{
        "ride": false,
        "gender": "",
        "age": "18-30",
        "drive": false,
        "nonsmoking": false
}
```

Operations

URL	http://example.com/trips/[trip_id]/preferences/
Method	GET
Description	Returns the Preferences of the Trip

Trip

Represents a Trip. See OpenTrip_Core#Entry_Elements (http://opentrip.info/wiki/OpenTrip_Core#Entry_Elements) for more info.

Attribute	**Type**	**Requirement**
published	string (Date)	MUST NOT
active	boolean	MUST
updated	string (Date)	MUST NOT
expires	string (Date)	MUST
author	object (Person)	MAY
locations	array (Location)	MUST
mode	object (Mode)	MUST
preferences	object (Preferences)	MUST
href	string	MUST NOT
participations	array (Participation)	MUST NOT

Data Representation Example

The following is a complete DycapoP Trip object, containing the other Entities used as example in the rest of the document

```
{
    "preferences": {
        "nonsmoking": false,
        "gender": "",
        "ride": false,
        "drive": false,
        "age": "18-30"
    },
    "expires": "2010-09-05 13:33:08",
    "locations": [
        {
            "town": "Bolzano",
            "point": "orig",
            "country": "",
            "region": "",
            "subregion": "",
            "days": "",
            "label": "Work",
            "street": "Rom Strasse",
            "postcode": 39100,
            "offset": 150,
            "leaves": "2010-09-02 13:32:34",
            "recurs": "",
            "georss_point": "46.490200 11.342294"
        },
        {
            "town": "Bolzano",
            "point": "dest",
            "country": "",
            "region": "",
            "subregion": "",
            "days": "",
            "label": "Work",
            "street": "Piazza della Vittoria, 1",
            "postcode": 39100,
            "offset": 150,
            "leaves": "2010-09-02 13:32:34",
            "recurs": "",
            "georss_point": "46.500740 11.345073"
        }
    ],
    "modality": {
        "kind": "auto",
        "capacity": 4,
        "lic": "",
        "color": "",
        "make": "Ford",
        "id": 4,
        "vacancy": 4,
        "cost": 0.0,
        "year": 0,
        "model_name": "Fiesta"
    }
}
```

Operations

URL	http://example.com/trips/
Method	GET
Description	Retrieves a collection of Trips

URL	http://example.com/trips/
Method	POST
Request Body	Trip
Description	Creates a Trip object

URL	http://example.com/trips/[trip_id]/
Method	GET
Request Body	Person
Description	Returns a Trip Resource

URL	http://example.com/trips/[trip_id]/
Method	PUT
Request Body	Trip
Description	Updates a Trip object

URL	http://example.com/trips/[trip_id]/
Method	DELETE
Request Body	Trip
Description	Deletes a Trip object

Participation

Represents a Participation in a Trip.

Attribute	Type	Requirement
author	object (Person)	MAY
status	string	MUST
href	string	MUST

- status attribute value MUST be from the set {"**request**","**accept**","**start**","**finish**"} and represents the current Participation status of a user

Data Representation Example

The following is a valid Dycapo Protocol Participation:

```
{
        "status": "accept",
        "person": {
            "username": "driver1",
            "href": "http://example.com/api/persons/rider1/",
            "location": {
                "href": "http://example.com/api/persons/rider1/location/"
            }
        },
        "href": "http://example.com/api/trips/4/participations/rider1/"
    }
```

Operations

URL	http://example.com/trips/[trip_id]/participations/
Method	GET
Description	Retrieves the Participations of the Trip

URL	http://example.com/trips/[trip_id]/participations/
Method	POST
Request Body	Participation
Description	Creates a Participation object related to the Trip (Requesting a Ride)

URL	http://example.com/trips/[trip_id]/participations/[username]/
Method	PUT
Request Body	Person
Description	Updates a Participation object related to the Trip (Accepting a Ride request, Starting a Ride, Finishing a Ride)

URL	http://example.com/trips/[trip_id]/participations/[username]/
Method	DELETE
Description	Deletes a Participation resource (Refusing a Ride request, Deleting a Ride request).

Search

This resource represents a search between Trips. Due to the complexity of the objects involved when searching a Trip, there is the necessity of creating state-ful Search objects, accessible each time a search is needed.

Attribute	Type	Requirement
origin	object (Location)	MUST
destination	object (Location)	MUST
author	object (Person)	MAY
trips	array (Trip)	MUST NOT

Data Representation Example

The following is a valid Search object:

```
{
    "origin": {
        "town": "Bolzano",
        "point": "posi",
        "href": "http://example.com/api/searches/37/",
        "country": "",
        "region": "",
        "subregion": "",
        "days": "",
        "label": "Work",
        "street": "Drususallee, 43/a",
        "postcode": 39100,
        "offset": 150,
        "id": 140,
        "leaves": "2010-09-02 13:32:34",
        "recurs": "",
        "georss_point": "46.494957  11.340239"
    },
    "author": {
        "username": "rider1",
        "href": "http://example.com/api/persons/rider1/"
    },
    "destination": {
        "town": "Bolzano",
        "point": "dest",
        "href": "http://example.com/api/searches/37/",
        "country": "",
        "region": "",
        "subregion": "",
        "days": "",
        "label": "Work",
        "street": "Piazza della Vittoria, 1",
        "postcode": 39100,
        "offset": 150,
        "id": 141,
        "leaves": "2010-09-02 13:32:34",
        "recurs": "",
        "georss_point": "46.500891  11.344306"
    }
}
```

Operations

URL	http://example.com/searches/
Method	POST
Request Body	Search
Description	Creates a Search object

URL	http://example.com/searches/[search_id]/
Method	GET
Description	Retrieves the Search object and the results, if any

Use of HTTP Response Codes

The Dycapo Protocol uses the response status codes defined in HTTP to indicate the success or failure of an operation. Consult the HTTP specification [RFC2616] for detailed definitions of each status code. In detail, DycapoP makes use of the following codes:

Code	Name	Description
200	OK	Denotes a successful operation, an entity containing additional information SHOULD be provided
201	Created	Denotes the creation of a resource. A representation of the Resource SHOULD be provided
204	No Content	Denotes the deletion of a resource. A representation of the Resource SHOULD NOT be provided
401	Unauthorized	The client provided wrong (or did not provide) credentials
403	Forbidden	The client does not have the rights for perform the request
404	Not Found	No Resource has been found at the given URI
415	Unsupported Media Type	A Protocol Error Occurred. A description of the error SHOULD be provided

License