Introduction to REST APIs

Olivier Liechti & Laurent Prévost
COMEM Web Services

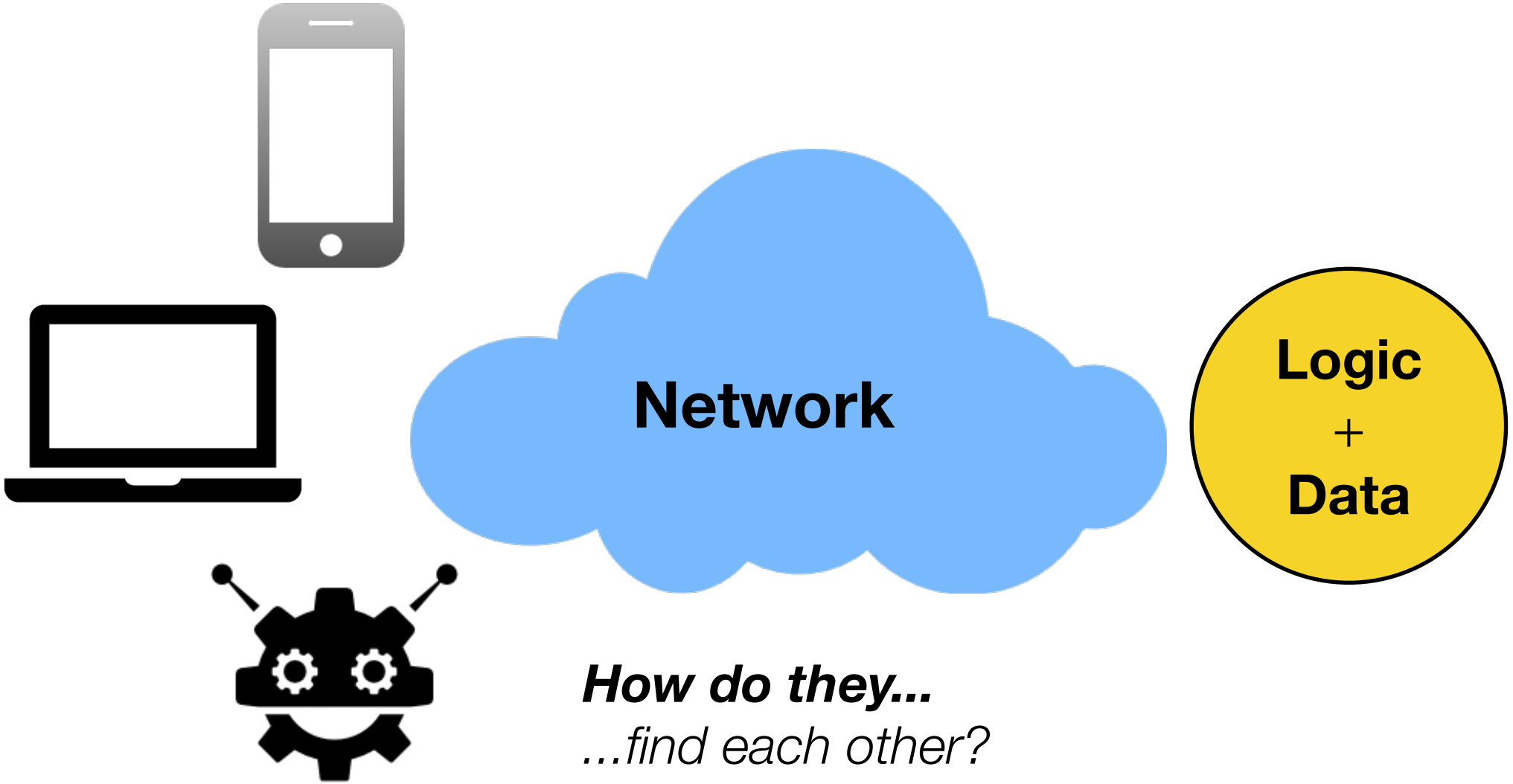
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Big Web Services vs REST

What is a Web Service?

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



How do they...

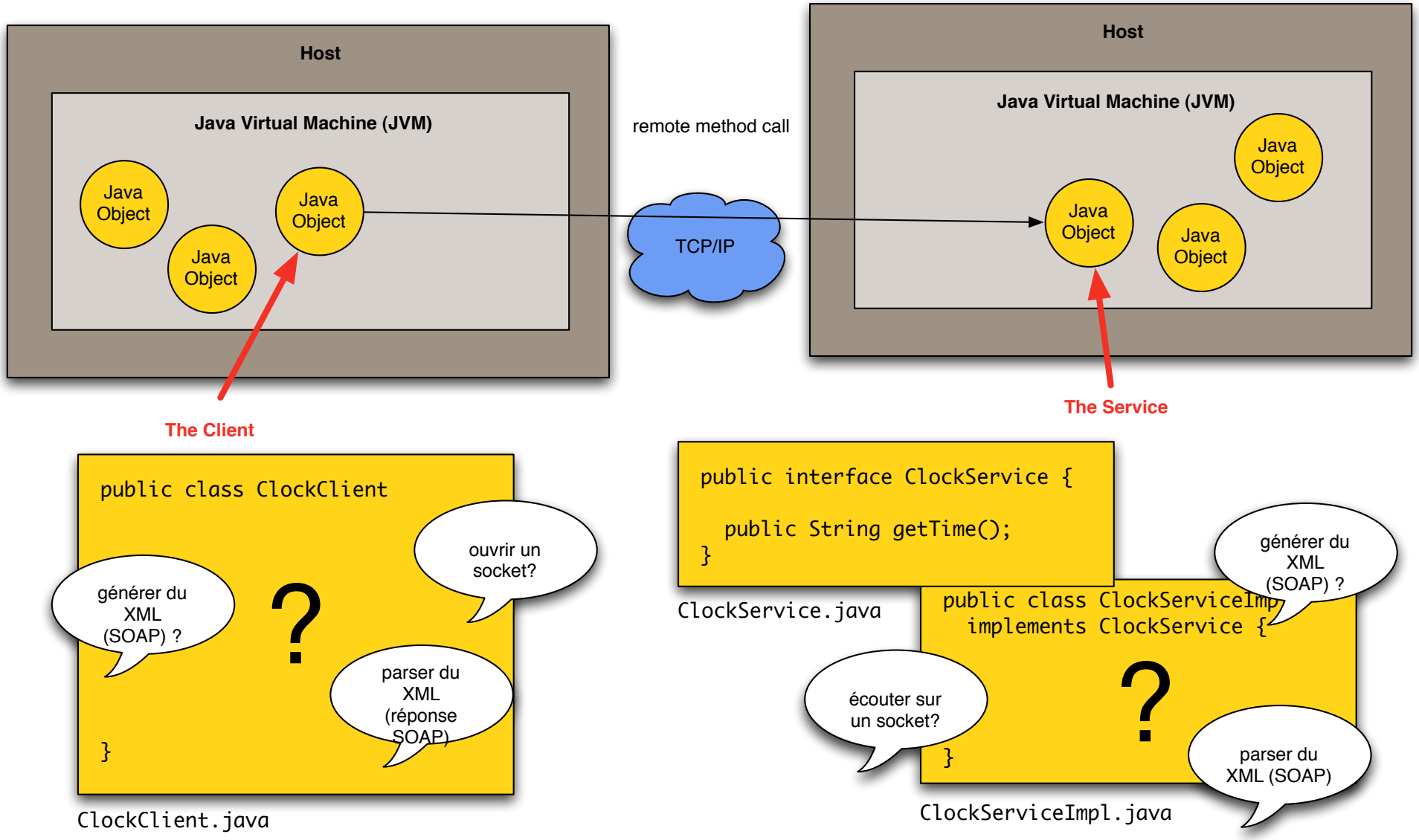
...find each other?

...know what logic can be invoked?

...talk to each other?

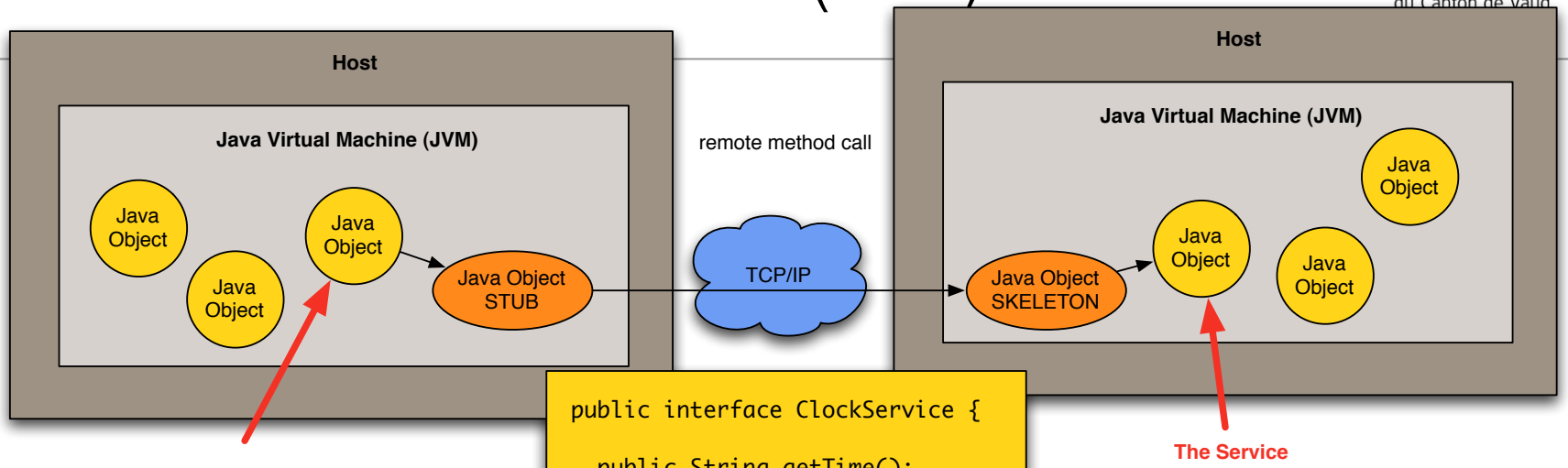
Remote services in Java (RMI)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



Remote services in Java (RMI)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud




```
public interface ClockService {

  public String getTime();
}
```

ClockService.java

```
public class ClockClient

  private ClockService service;

...

  public String showTime() {
    System.out.println("It is " +
      service.getTime());
  }
}
```

ClockClient.java

```
public class ClockServiceStub
  implements ClockService {

  public String getTime() {
    ... bla bla... SOCKET ...
    XML... bla bla bla... SOAP ...
    bla bla OUTPUTSTREAM... bla bla
    INPUTSTREAM....

    return time;
}
```

ClockServiceStub.java

```
public class ClockServiceSkeleton {

  private ClockService realService;

  private void setup() { ... }
  private void listen() { ... }
  private void processCall {
    realService.getTime();
  }
}
```

ClockServiceSkeleton.java

```
public class ClockServiceImpl
  implements ClockService {

  public String getTime() {
    return new java.util.Date();
  }
}
```

ClockServiceImpl.java

Generated automatically by special tools!

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

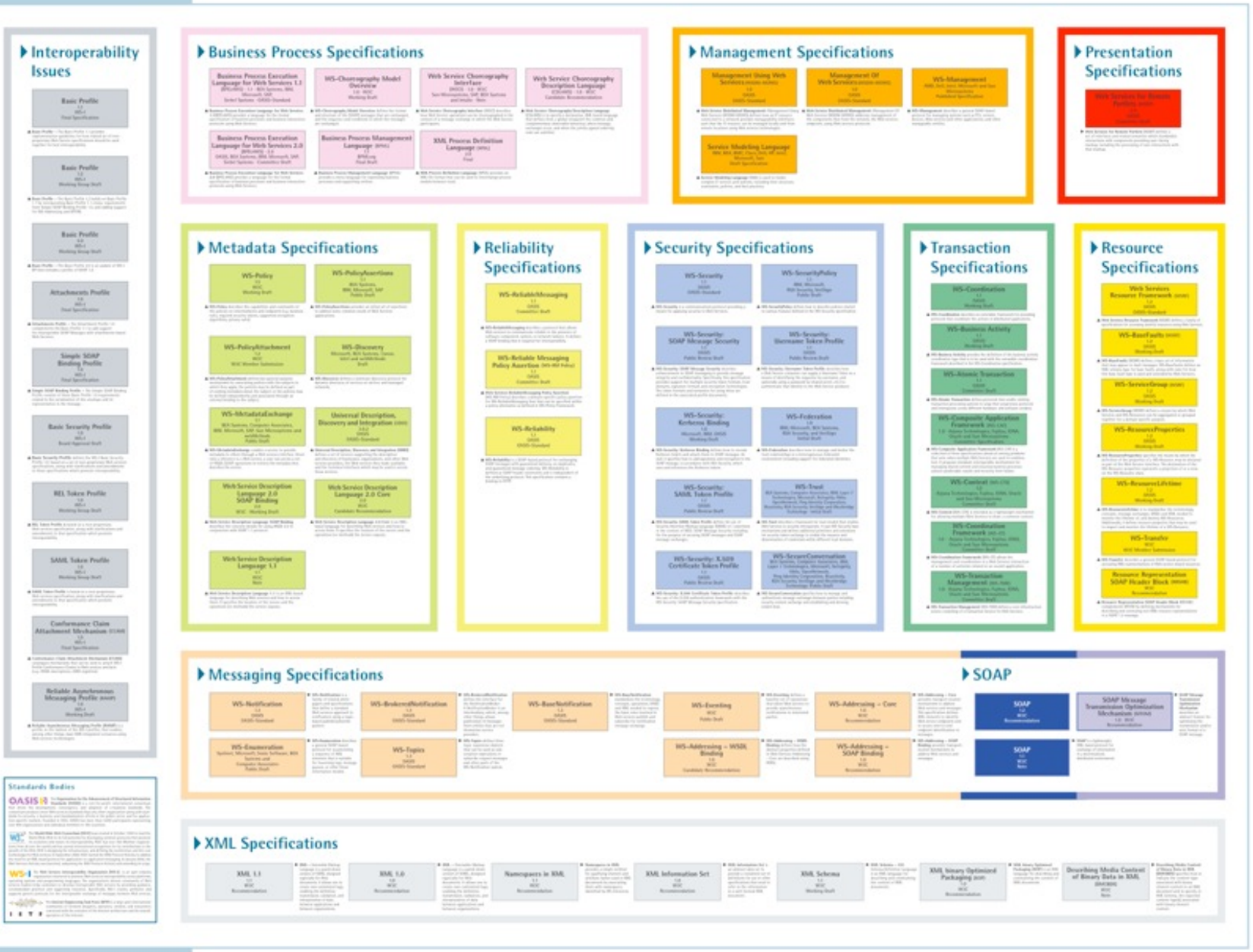
The “Big” Web Services Approach

Big Web Services

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

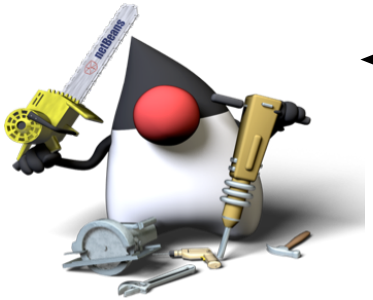
- **Approach**
  - Services are often designed and developed with a **RPC style** (even if Document-Oriented Services are possible).
- **Core Standards**
  - Simple Object Access Protocol (**SOAP**)
  - Web Services Description Language (**WSDL**)
- **Benefits**
  - **Very rich protocol stack** (support for security, transactions, reliable transfer, etc.)
- **Problem**
  - **Very rich protocol stack** (complexity, verbosity, incompatibility issues, theoretical human readability, etc.)

Web Services Standards Overview



http://www.innoq.com/soa/ws-standards/poster/
http://www.innoq.com/soa/ws-standards/

Web Services Standards Overview



Enterprise Platforms (Java EE, .NET, etc.) hide some of the complexity.

```
@Stateless
@WebService
public class Demo implements DemoLocal {

  @Override
  public String getTime() {
    return new Date().toString();
  }

  @Override
  public long computeSum(long v1, long v2) {
    return v1 + v2;
  }

}
```

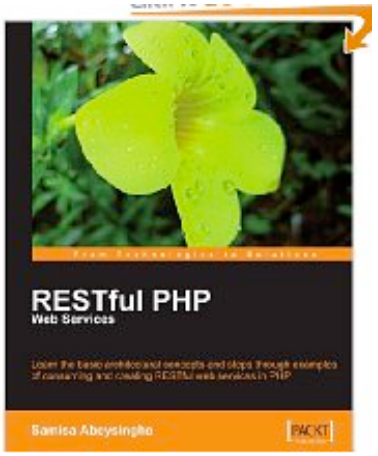
Adding a **@WebService** annotation does the magic. The application server takes care of all the gory details: generation of the WSDL interface, marshalling/unmarshalling of the SOAP messages. Still…

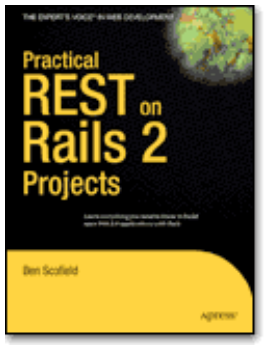
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

The REST Approach

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

RESTful Web Services





The REST Architectural Style

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- REST: **REpresentational State Transfer**
- REST is an **architectural style** for building distributed systems.
- REST has been introduced in **Roy Fielding's Ph.D. thesis** (Roy Fielding has been a contributor to the HTTP specification, to the apache server, to the apache community).
- The WWW is **one example** for a distributed system that exhibits the characteristics of a REST architecture.

Principles of a REST Architecture

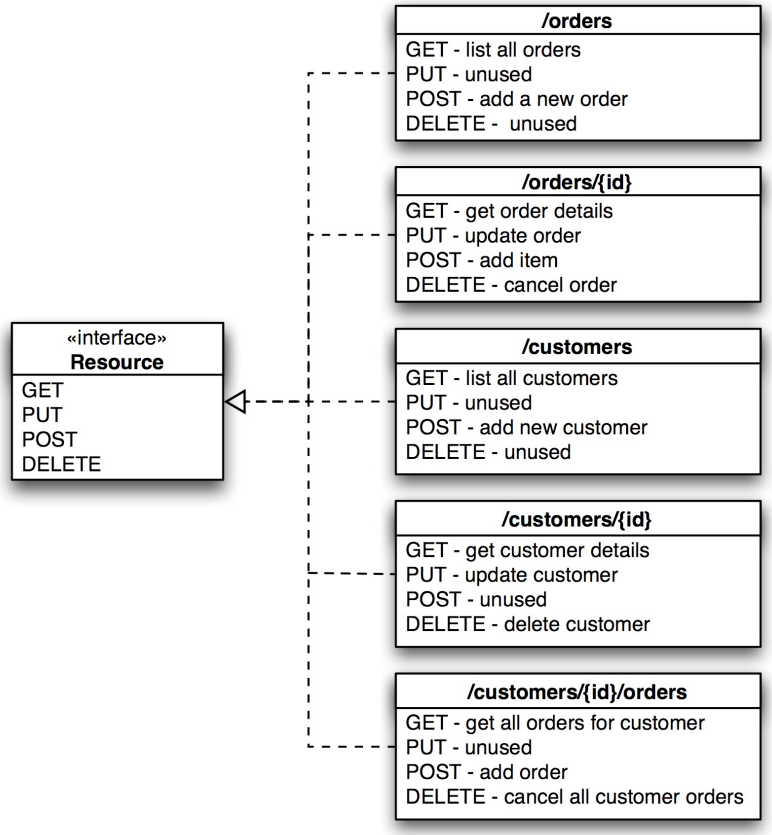
heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- The state of the application is captured in a **set of resources**
  - Users, photos, comments, tags, albums, etc.
- Every resource is **identified with a standard format** (e.g. URL)
- Every resource can have **several representations**
- There is one **unique interface for interacting** with resources (e.g. HTTP methods)

References

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- Very good article, with presentation of key concepts and illustrative examples:
  - http://www.infoq.com/articles/rest-introduction
- Suggestions for the design of "pragmatic APIs"
  - http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api



HTTP is a protocol for interacting with "**resources**"

What is a “**Resource**”

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- At first glance, one could think that a “resource” is a file on a web server:
  - an HTML document, an XML document, a PNG document
- That fits the vision of the “static content” web
- But of course, the web is now more than a huge library of hypermedia documents:
  - through the web, we interact with services and a lot of the content is dynamic.
  - more and more, through the web we interact with physical objects (machines, sensors, actuators)
  - We need a more generic definition for resources!

What is a “**Resource**”?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- A resource is "something" that can be named and uniquely identified:
  - Example 1: an article published in the "24 heures" newspaper
  - Example 2: the collection of articles published in the sport section of the newspaper
  - Example 3: a person’s resume
  - Example 4: the current price of the Nestlé stock quote
  - Example 5: the vending machine in the school hallway
  - Example 6: the list of grades of the student Jean Dupont
- URL (Uniform Resource Locator) is a mechanism for identifying resources
  - Exemple 1: http://www.24heures.ch/vaud/vaud/2008/08/04/trente-etudiants-partent-rencontre-patrons
  - Exemple 2: http://www.24heures.ch/articles/sport
  - Exemple 5: http://www.smart-machines.ch/customers/heig/machines/8272

Resource vs. **Representation**

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- A "resource" can be something intangible (stock quote) or tangible (vending machine)
- The HTTP protocol supports the exchange of data between a client and a server.
- Hence, what is exchanged between a client and a server is **not** the resource. It is a **representation** of a resource.
- Different representations of the same resource can be generated:
  - HTML representation
  - XML representation
  - PNG representation
  - WAV representation
- **HTTP provides the content negotiation mechanisms!!**

Universal interface

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

CRUD

Create Read Update Delete

CREATE RUD

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
POST /users/ HTTP/1.1
Host: www.myservice.com
Content-type: application/json

{
  “firstName” : “olivier”,
  “lastName” : “liechti”
}
```

```
HTTP/1.1 201 Created
Location: /users/7823
```

C **READ** U D

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
GET /users/ HTTP/1.1
Host: www.myservice.com
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-type: application/json

[
  { “id” : 7823 },
  {“id” : 7824}
]
```

C **READ** U D

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
GET /users/7823 HTTP/1.1
Host: www.myservice.com
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
  “firstName” : “olivier”,
  “lastName” : “liechti”
}
```

CR **UPDATE** D

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
PUT /users/7823 HTTP/1.1
Host: www.myservice.com
Content-type: application/json

{
  “firstName” : “Olivier”,
  “lastName” : “Liechti”
}
```

```
HTTP/1.1 204 No Content
```

CRU **DELETE**


```
DELETE /users/7823 HTTP/1.1
Host: www.myservice.com
```

```
HTTP/1.1 204 No Content
```

References in resource representations

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
GET /users/7823 HTTP/1.1
Host: www.myservice.com
```

```
HTTP/1.1 200 OK
Content-type: application/json

{
  'firstName' : 'olivier',
  'lastName' : 'liechti',
  'organizationId' : 892
}
```

ID

```
HTTP/1.1 200 OK
Content-type: application/json

{
  'firstName' : 'olivier',
  'lastName' : 'liechti',
  'organization' : '/organizations/892'
}
```

URL

Object graphs & URLs

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
GET /companies/8939/employees/192/office HTTP/1.1
```

VS

```
GET /office?occupantId=192 HTTP/1.1
```

```
GET /cars/882/wheels/frontLeft HTTP/1.1
```

VS

```
GET /wheels/89278927 HTTP/1.1
```

Filtering & Sorting in collections

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
GET /users/?filterByZipCode=1446&orderBy=lastName HTTP/1.1
Host: www.myservice.com
```

Pagination in collections

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
GET /users/ HTTP/1.1
Host: www.myservice.com
Accept: application/json
X-Page-Size: 10
```

```
HTTP/1.1 200 OK
Content-type: application/json
X-Page-Number: 1
X-Elements-Count: 23
X-Pages-Count: 3
X-Page-Size: 10

[{...},...]
```

```
GET /users/ HTTP/1.1
Host: www.myservice.com
Accept: application/json
X-Page-Size: 10
X-Page-Number: 2
```

Alternative: **link headers**

`http://tools.ietf.org/html/rfc5988#page-6`

https://developer.github.com/guides/traversing-with-pagination/

```
Link: <https://api.github.com/search/code?q=addClass+user%3Amozilla&page=15>; rel="next",
  <https://api.github.com/search/code?q=addClass+user%3Amozilla&page=34>; rel="last",
  <https://api.github.com/search/code?q=addClass+user%3Amozilla&page=1>; rel="first",
  <https://api.github.com/search/code?q=addClass+user%3Amozilla&page=13>; rel="prev"
```

URLs, actions & verbs

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
GET /cars/9383/startEngine HTTP/1.1
```

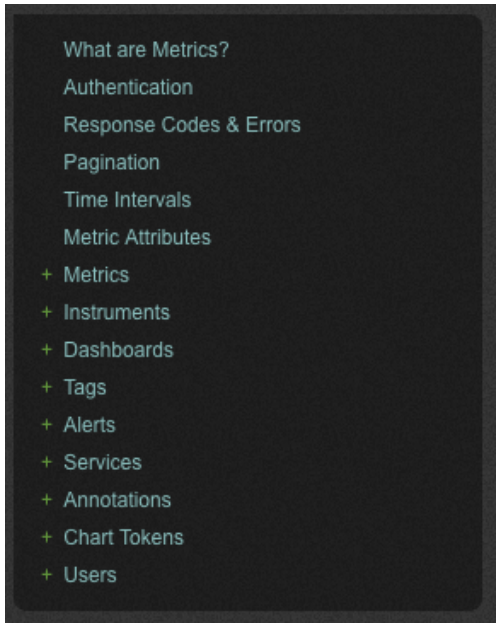
VS

```
POST /cars/9383/startEngine HTTP/1.1
```

VS

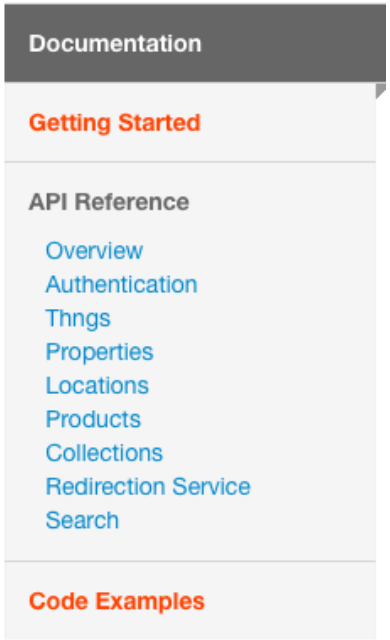
```
POST /cars/9383/commands HTTP/1.1
Content-type: application/json

{
  ‘command’ : ‘startEngine’,
  ‘params’ : [
    ‘when’ : ‘ASAP’,
    ‘how’ : ‘withTurbo’
  ]
}
```

Look at Some Examples



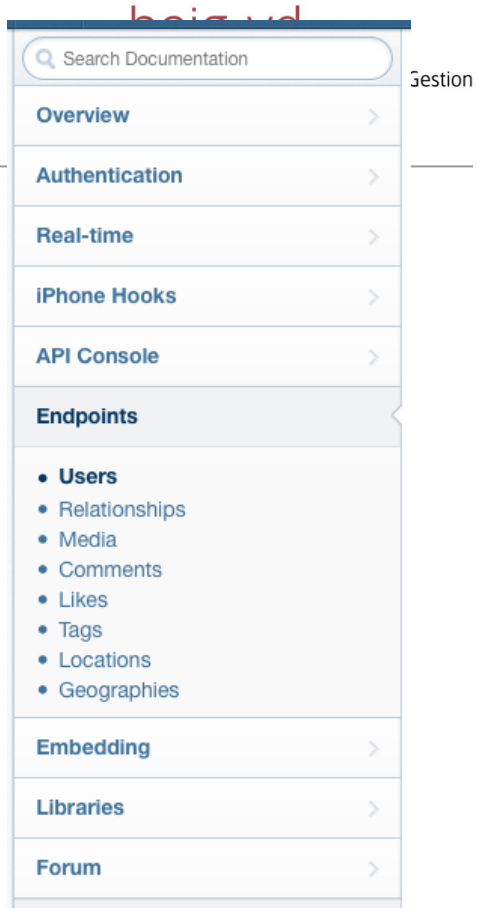
http://dev.librato.com/v1

librato



https://dev.evrythng.com/documentation/api

EVRYTHNG



http://instagram.com/developer/endpoints/

Instagram

Short description of the resource (domain model)

heig-vd

What Are Metrics?

Metrics are custom measurements stored in Librato's Metrics service. These measurements are created and may be accessed programatically through a set of RESTful API calls. There are currently two types of metrics that may be stored in Librato Metrics, `gauges` and `counters`.

Gauges

Gauges capture a series of measurements where each measurement represents the value under observation at one point in time. The value of a gauge typically varies between some known minimum and maximum. Examples of gauge measurements include the requests/second serviced by an application, the amount of available disk space, the current value of $AAPL, etc.

Counters

Counters track an increasing number of occurrences of some event. A counter is unbounded and always monotonically increasing in any given *run*. A new *run* is started anytime that counter is reset to zero. Examples of counter measurements include the number of connections made to an app, the number of visitors to a website, the number of a times a write operation failed, etc.

Metric Properties

Some common properties are supported across all types of metrics:

`name`

Each metric has a name that is unique to its class of metrics e.g. a gauge name must be uni
gauges. The name identifies a metric in subsequent API calls to store/query individual meas
can be up to 63 characters in length. Valid characters for metric names are 'A-Za-z0-9.:-_'.

`period`

The `period` of a metric is an integer value that describes (in seconds) the standard reportin
metric. Setting the period enables Metrics to detect abnormal interruptions in reporting and a

Examples & payload structure

Examples

Return the metric named *cpu_temp* with up to four measurements at resolution 60.

```
curl \
  -u <user>:<token> \
  -X GET \
  https://metrics-api.librato.com/v1/metrics/cpu_temp?resolution=60&count=4
```

Response Code

```
200 OK
```

Response Headers

```
** NOT APPLICABLE **
```

Response Body

```
{
  "type": "gauge",
  "display_name": "cpu_temp",
  "resolution": 60,

: {
e.com": [

 1,
_time": 1234567890,
 84.5

 1,
_time": 1234567950,
 86.7

 1,
_time": 1234568010,
 84.6

 1,
_time": 1234568070,
 89.7

{
": 0,
nsform": null,
ts_short": "&#176;F",
ua": "librato-metrics/0.7.4 (ruby; 1.9.3p194; x86_64-linux) direct-faraday/0.8.4
": null,
ts_long": "Fahrenheit",
cked": true

 "Current CPU temperature in Fahrenheit",
emp"
```

CRUD method description

GET /v1/metrics/:name

API VERSION 1.0

Description

Returns information for a specific metric. If time interval search parameters are specified will also include a set of metric measurements for the given time span.

URL

```
https://metrics-api.librato.com/v1/metrics/:name
```

Method

```
GET
```

Measurement Search Parameters

If optional time interval search parameters are specified, the response includes the set of metric measurements covered by the time interval. Measurements are listed by their originating *source* name if one was specified when the measurement was created. All measurements that were created without an explicit source name are listed with the source name `unassigned`.

`source`

Deprecated: Use `sources` with a single source name, e.g `[mysource]`.

`sources`

If `sources` is specified, the response is limited to measurements from those sources. The `sources` parameter should be specified as an array of source names. The response is limited to the set of sources specified in the array.

navigator

- What are Metrics?
- Authentication
- Response Codes & Errors
- Pagination
- Time Intervals
- Metric Attributes
- \- Metrics
  - GET /metrics
  - POST /metrics
  - DELETE /metrics
  - GET /metrics/:name
  - PUT /metrics/:name
  - DELETE /metrics/:name
- \+ Instruments
- \+ Dashboards
- \+ Tags
- \+ Alerts
- \+ Services
- \+ Annotations
- \+ Chart Tokens
- \+ Users

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Short description of the whole domain model

Overview

The central data structure in our engine are `Thngs`, which are data containers to store all the data generated by and about any physical object. Various `Properties` can be attached to any Thng, and the content of each property can be updated any time, while preserving the history of those changes. Thngs can be added to various `Collections` which makes it easier to share a set of Thngs with other `Users` within the engine.

`Thng`

An abstract notion of an object which has location & property data associated to it. Also called Active Digital Identities (ADIs), these resources can model real-world elements such as persons, places, cars, guitars, mobile phones, etc.

`Property`

A Thng has various properties: arbitrary key/value pairs to store any data. The values can be updated individually at any time, and can be retrieved historically (e.g. "Give me the values of property X between 10 am and 5 pm on the 16th August 2012").

`Location`

Each Thng also has a special type of Properties used to store snapshots of its geographic position over time (for now only GPS coordinates - latitude and longitude).

`User`

Each interaction with the EVRYTHNG back-end is authenticated and a user is associated with each action. This dictates security access.

`Collection`

A collection is a grouping of Thngs. Col one collection.

CRUD method description

Creating a new Product

To create a new `Product`, simply POST a JSON document that describes a product to the `/products` endpoint.

```
POST /products
Content-Type: application/json
Authorization: $EVRYTHNG_API_KEY
```

```
{
  *"fn": <String>,
  "description": <String>,
  "brand": <String>,
  "categories": [<String>, ...],
  "photos": [<String>, ...],
  "url": <String>,
  "identifiers": {
    <String>: <String>,
    ... },
  "properties": {
    <String>: <String>,
    ... },
  "tags": [<String>, ...]
}
```

Mandatory Parameters

`fn`

<String> The functional name of the product.

Optional Parameters

`description`

<String> An string that gives more details about the product, a short description.

More details about the Product resource (domain model) & payload structure

Products

Products are very similar to thngs, but instead of modeling an individual object instance, products are used to model a class of objects. Usually, they are used for general classes of thngs, usually a particular model with specific characteristics. Let's take for example a specific TV model (e.g. this one), which has various properties such as a model number, a description, a brand, a category, etc. Products are useful to captor the properties that are common to a set of thngs (so you don't replicate a property "model name" or "weight" for thousands of thngs that are individual instances of a same product category).

The Product document model used in our engine has been designed to be compatible with the hProduct microformat, therefore it can easily be integrated with the hProduct data model and applications supporting microformats.

The Product document model is as follows:

```
<Product>={
  "id": <String>,
  "createdAt": <timestamp>,
  "updatedAt": <timestamp>,
  "fn": <String>,
  "description": <String>,
  "brand": <String>,
  "categories": [<String>, ...],
  "photos": [<String>, ...],
  "url": <String>,
  "identifiers": {
    <String>: <String>,
    ... },
  "properties": {
    <String>: <String>,
    ... },
  "tags": [<String>, ...]
}
```

Cross-cutting concerns

Pagination

Requests that return multiple items will be paginated to 30 items by default. You can specify further pages with the `?page` parameter. You can also set a custom page size up to 100 with the `?per_page` parameter.

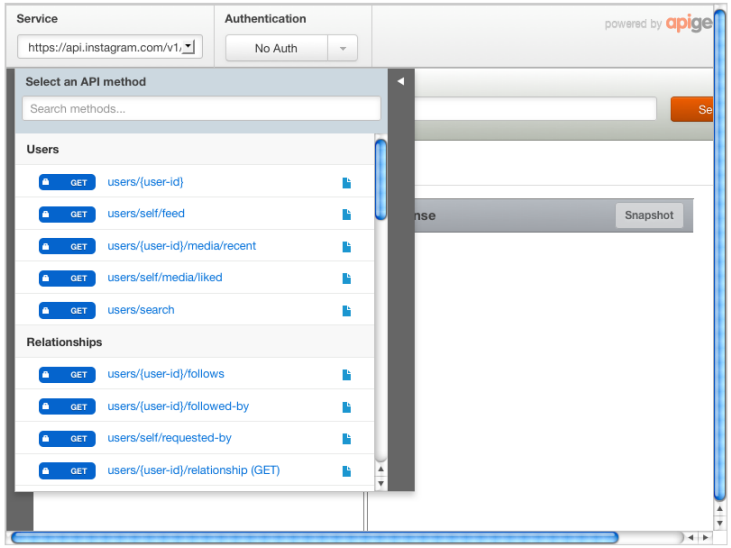
Authentication

Access to our API is done via HTTPS requests to the `https://api.evrythng.com` domain. Unencrypted HTTP requests are accepted (`http://api.evrythng.com` for low-power device without SSL support), but we **strongly** suggest to use only HTTPS if you store any valuable data in our engine. Every request to our API must include an API key using `Authorization` HTTP header to identify the user or application issuing the request and execute it if authorized.

Interactive test console

API Console

Our API console is provided by Apigee. Tap the Lock icon, select OAuth, and you can experiment with making requests to our API. See it in full screen →



List of supported CRUD methods for each resource (R, RW)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

User Endpoints

GET	/users/ **user-id**	Get basic information about a user.
GET	/users/self/feed	See the authenticated user's feed.
GET	/users/ **user-id** /media/recent	Get the most recent media published by a user.
GET	/users/self/media/liked	See the authenticated user's list of liked media.
GET	/users/search	Search for a user by name.

Comment Endpoints

GET	/media/ **media-id** /comments	Get a full list of comments on a media.
POST	/media/ **media-id** /comments	Create a comment on a media. Please email apide…
DEL	/media/ **media-id** /comments/ **comment-id**	Remove a comment.

GET /media/ **media-id** /comments

https://api.instagram.com/v1/media/555/comments?access_token=ACCESS-TOKEN

RESPONSE


```
{
    "meta": {
        "code": 200
    },
    "data": [
        {
            "created_time": "1280780324",
            "text": "Really amazing photo!",
            "from": {
                "username": "snoopdogg",
                "profile_picture": "http://images.instagram.com/profiles/profile_16_75sq_1305612434.jpg",
                "id": "1574083",
                "full_name": "Snoop Dogg"
            },
            "id": "420"
        },
        ...
    ]
}
```


Get a full list of comments on a media.

Required scope: comments

CRUD method description

Cross-cutting concerns

Limits

Be nice. If you're sending too many requests too quickly, we'll send back a 503 error code (server unavailable).

You are limited to 5000 requests per hour per access_token or client_id overall. Practically, this means you should (when possible) authenticate users so that limits are well outside the reach of a given user.

PAGINATION

Sometimes you just can't get enough. For this reason, we've provided a convenient way to access more data in any request for sequential data. Simply call the url in the next_url parameter and we'll respond with the next set of data.

The Envelope

Every response is contained by an envelope. That is, each response has a predictable set of keys with which you can expect to interact:


```
{
    "meta": {
        "code": 200
    },
    "data": {
        ...
    },
    "pagination": {
        "next_url": "...",
        "next_max_id": "13872296"
    }
}
```

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Tools & Demo