HTTP

Pedro Félix

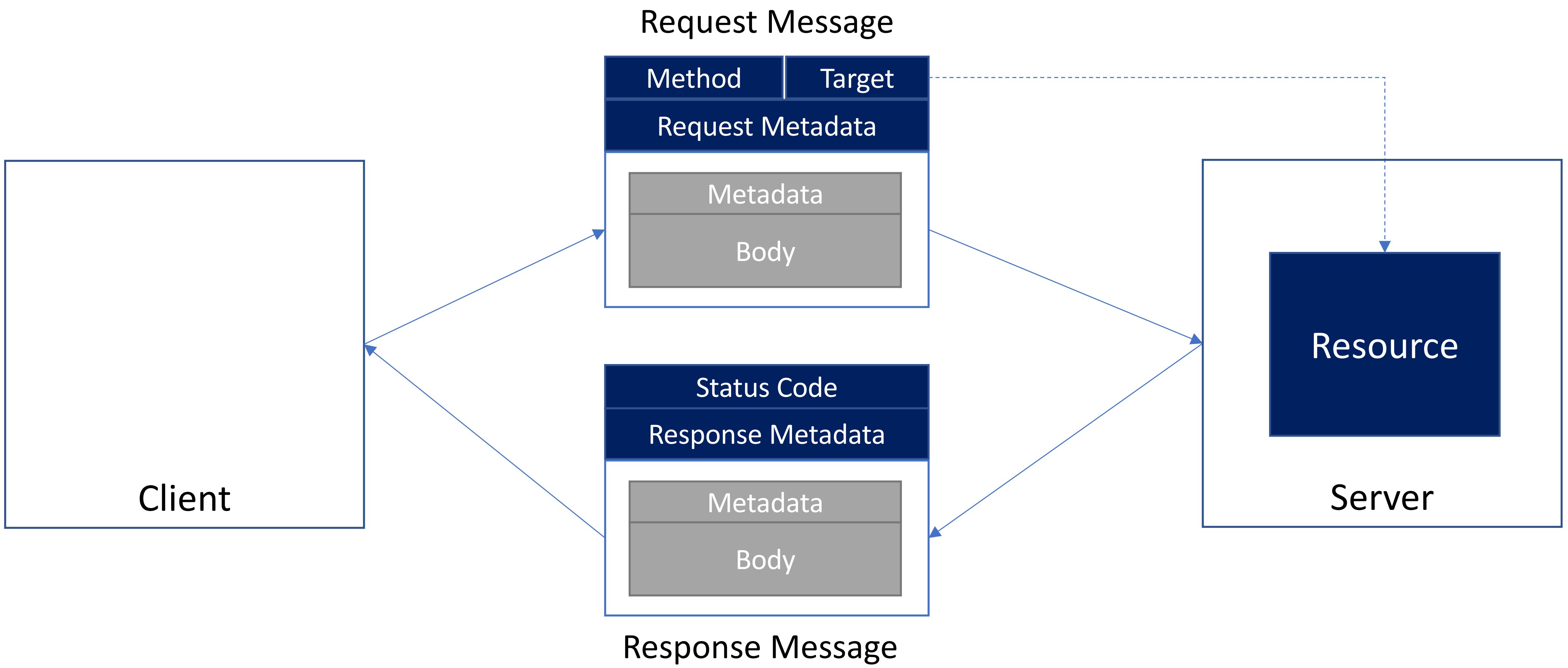
March 2020

“The Hypertext Transfer Protocol (HTTP) is a ***stateless application-level protocol*** *for* ***distributed****, collaborative, hypertext* ***information systems****. This document defines the semantics of HTTP/1.1* ***messages****, as expressed by* ***request methods****,* ***request header fields****,* ***response status codes****, and* ***response header fields****, along with the* ***payload of messages*** *(****metadata*** *and* ***body content****) and mechanisms for content negotiation.”*

In https://tools.ietf.org/html/rfc7231

… operates by exchanging messages …

*“HTTP is a stateless request/response protocol that **operates by exchanging messages** (…).*
*An HTTP "**client**" is a program that establishes a connection to a server for the purpose of **sending one or more HTTP requests**.*
*An HTTP "**server**" is a program that accepts connections in order **to service HTTP requests** by sending HTTP responses.”*



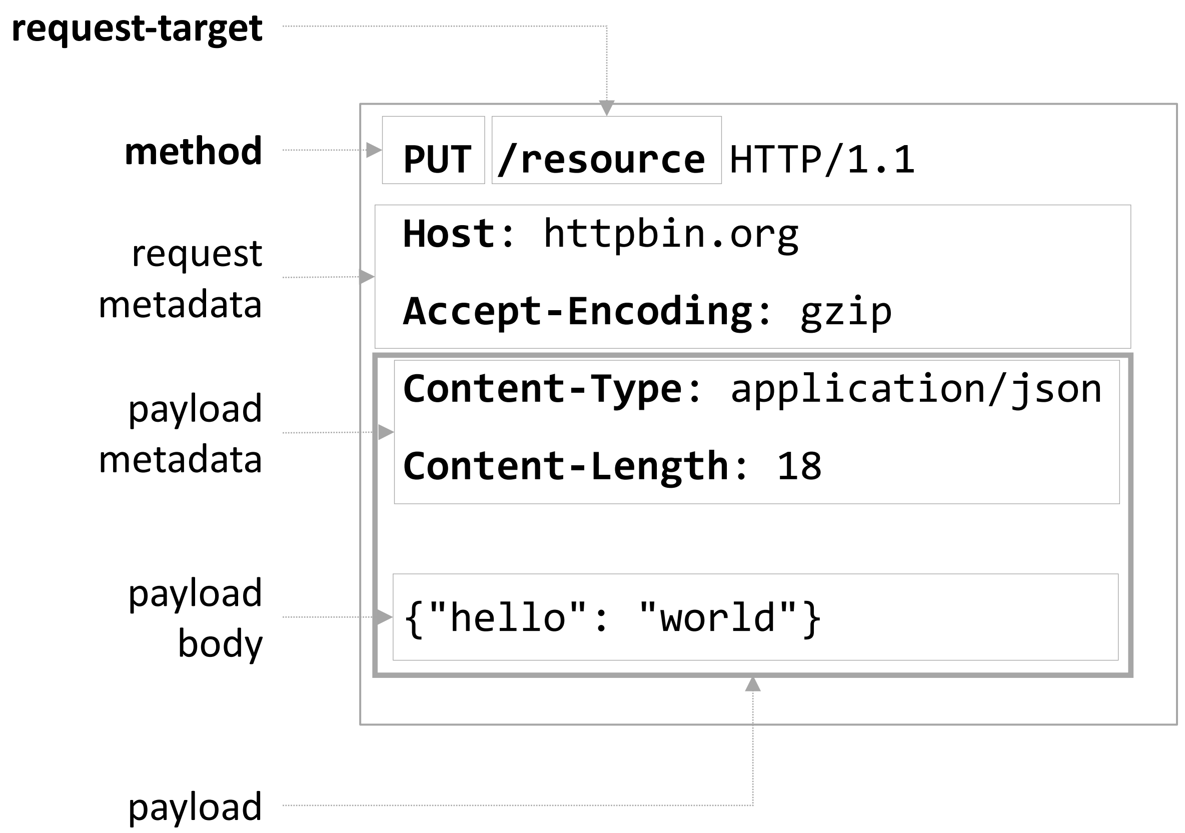
Request Message

*“The **request-target** identifies the **target resource** upon which to **apply the request**”*

*“The **request method** token is the **primary source of request semantics**; it **indicates the purpose for which the client has made this request** and **what is expected by the client as a successful result**.”*

*“A client sends **request header fields** to **provide more information** about the **request context**”*

“Some HTTP messages transfer a complete or partial representation as the message "payload
(…)
*The **purpose of a payload** in a request is **defined by the method semantics***
(…)
In a response, the payload's purpose is defined by both the request method and the response status code”



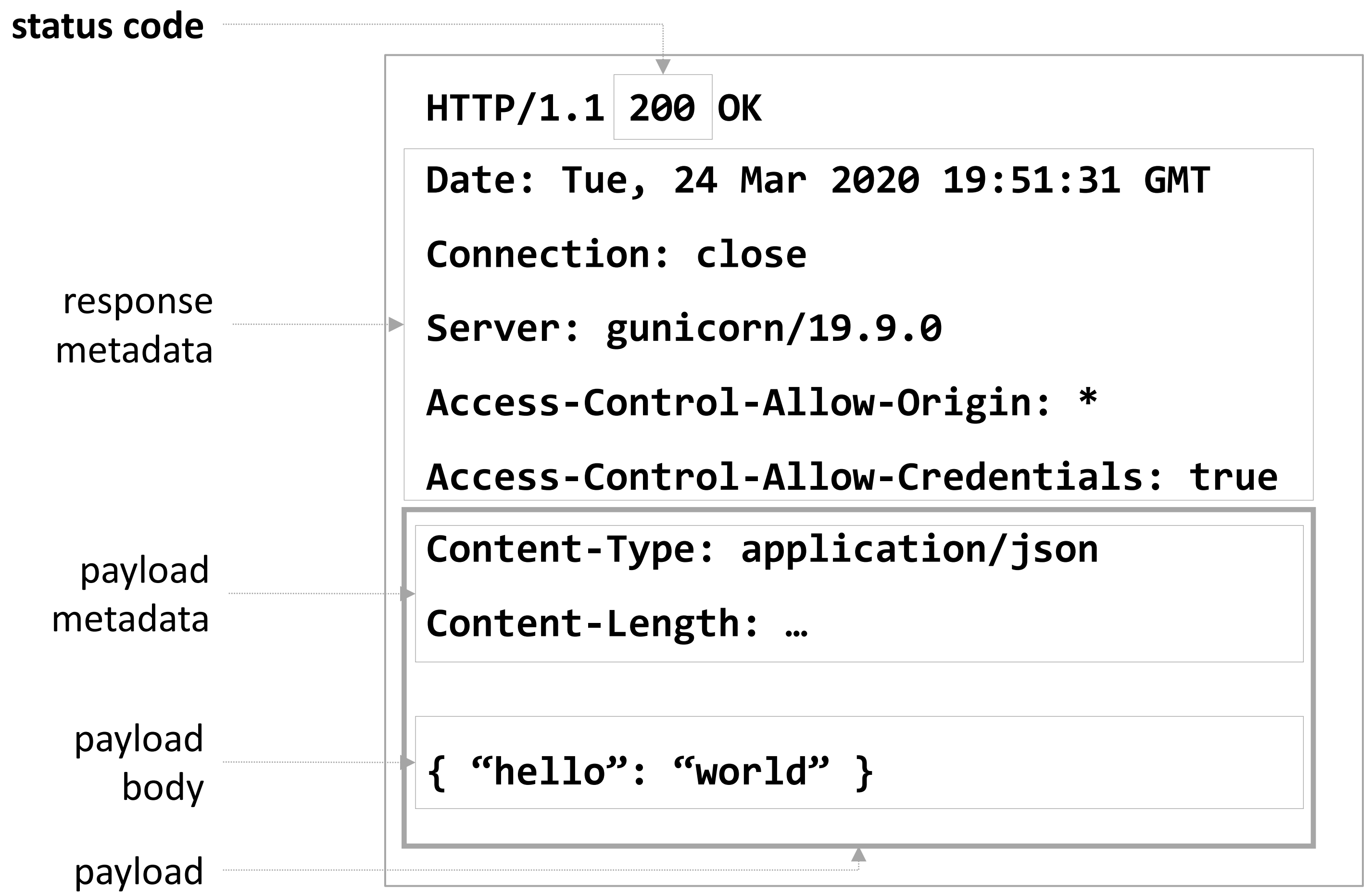
```
PUT /resource HTTP/1.1
Host: httpbin.org
Accept-Encoding: gzip
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
```

Response Message

*"The **status-code** element is a 3-digit integer code **describing the result of the server's attempt** to **understand** and **satisfy** the client's corresponding request. The rest of the **response message** is to be **interpreted in light of the semantics defined for that status code**"*

*"The **response header fields** allow the server to pass **additional information about the response** beyond what is placed in the status-line"*

*"In a response, the **payload's purpose** is **defined** by **both** the **request method** and the **response status code**"*



```
HTTP/1.1 200 OK
Date: Tue, 24 Mar 2020 19:51:31 GMT
Connection: close
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Type: application/json
Content-Length: …

{ "hello": "world" }
```

Request methods

- “***Unlike distributed objects***, *the standardized request methods in HTTP* ***are not resource-specific***, *since uniform interfaces provide for* ***better visibility*** *and reuse in network-based systems*”
  in https://tools.ietf.org/html/rfc7231#section-4

- “What **makes HTTP significantly different from RPC** is that the requests are directed to resources using a **generic interface** with **standard semantics** that **can be interpreted by intermediarie**s almost as well as by the machines that originate services.
  The result is an **application that allows for layers of transformation** and indirection that are independent of the information origin”
  in https://www.ics.uci.edu/~fielding/pubs/dissertation/evaluation.htm

Request methods

- GET
  - obtain a representation for the target resource
- PUT
  - define a resource state (**create** or update)
- PATCH
  - partially update a resource (RFC 5789)
- DELETE
  - delete a resource
- POST
  - **processing** of the **enclosed request representation** by the **target resource**

Request methods

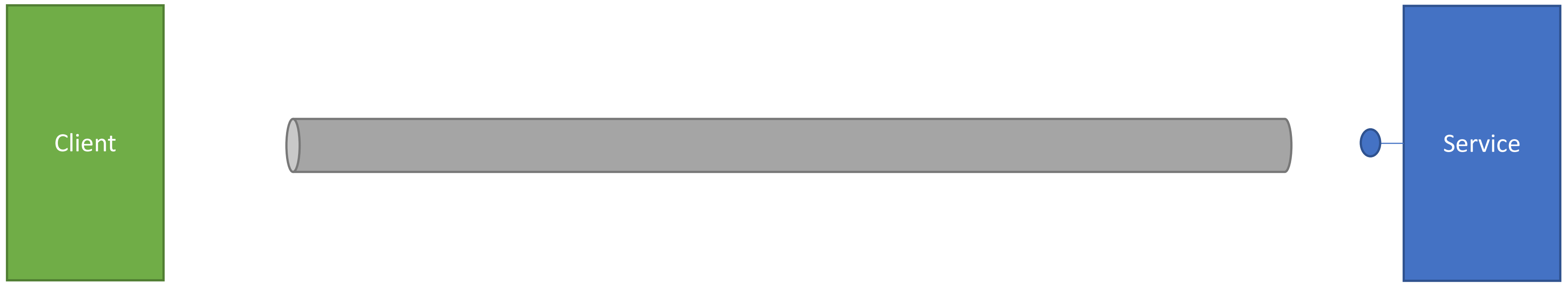
- GET
  - obtain a representation for the target resource
- PUT
  - define a resource state (create or update)
- PATCH
  - partially update a resource (RFC 5789)
- DELETE
  - delete a resource
- POST
  - processing of the enclosed request representation by the target resource

POST does not mean create

PUT can also be used to create

- HEAD
  - Similar to GET but without the representation body
- OPTIONS
  - Obtain the communication options available for the target resource
- TRACE
  - Obtain a Loop-back

Interaction failure



- Communication failures
  - DNS lookup failure (not fond or timeout)
  - TCP connection failure (rejected or timeout)
  - Message bytes send error
  - Response bytes receive error
  - TCP connection closed
  - Malformed response message
- Receive message with a non-success status code

Status code

- A **request message** solicits the realization of an operation on a resource.
- Origin-servers may not be **able** or **willing** to perform the requested operation.
- A **response message** contains the request outcome.
  - Sent even if the requested operation was not performed.
  - The origin-server can opt to not send a response and instead close the connection.
- The response's **status code** is the primary way to convey the request's outcome.
- The remaining response message elements should be interpreted according to the response status code.

The status-code element is a three-digit integer code giving the ***result of the attempt to understand and satisfy the request****.*

In https://tools.ietf.org/html/rfc7231#section-6

Five status code categories

- o 1xx (Informational): The request was received, **continuing** process
- o 2xx (Successful): The request was **successfully received**, **understood**, and **accepted**
- o 3xx (Redirection): **Further action** needs to be taken in order to **complete the request**
- o 4xx (Client Error): The request contains **bad syntax** or **cannot be fulfilled**
- o 5xx (Server Error): The server **failed to fulfil** an **apparently valid request**

In https://tools.ietf.org/html/rfc7231#section-6

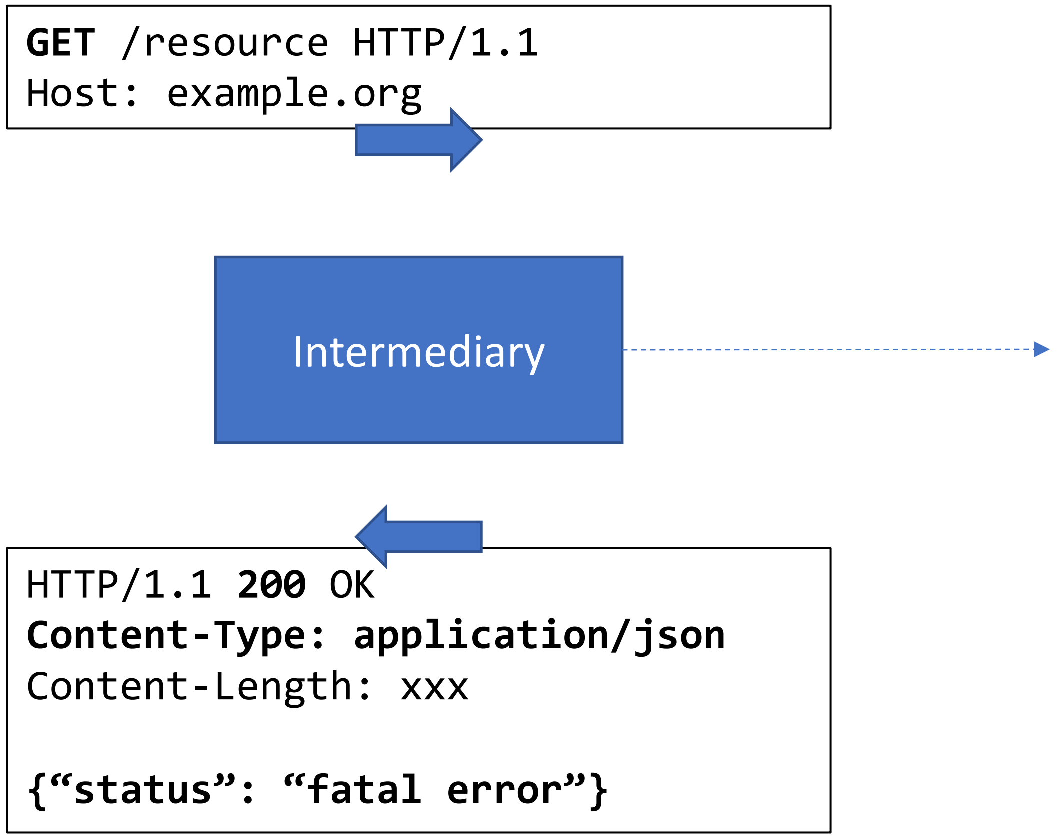
Expect: 100-continue

```
PUT https://httpbin.org/put HTTP/1.1

Host: httpbin.org

Content-Type: application/json

Content-Length: 5

Expect: 100-continue

(empty line)

                                                    HTTP/1.1 100 Continue

                                                    (empty line)

"123"

                                                    HTTP/1.1 200 OK

                                                    Content-Type: application/json

                                                    Content-Length: 337

                                                    Connection: keep-alive

                                                    Server: gunicorn/19.9.0

                                                    {…}
```

Success comes in various status

- **`200 OK`**
  - Request has succeeded
  - Body semantics depends on the request method
    - **GET** - representation of the target resource
    - **POST** - representation of the status or results obtained from the action
    - **PUT**, **DELETE** - representation of the status of the action
    - **OPTIONS** - representation of the communications options
    - **TRACE** - representation of the request message

Uniform interface



```
GET /resource HTTP/1.1
Host: example.org
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: xxx

{“status”: “fatal error”}
```

Intermediary can conclude that a representation of the resource identified by

https://example.org/resource

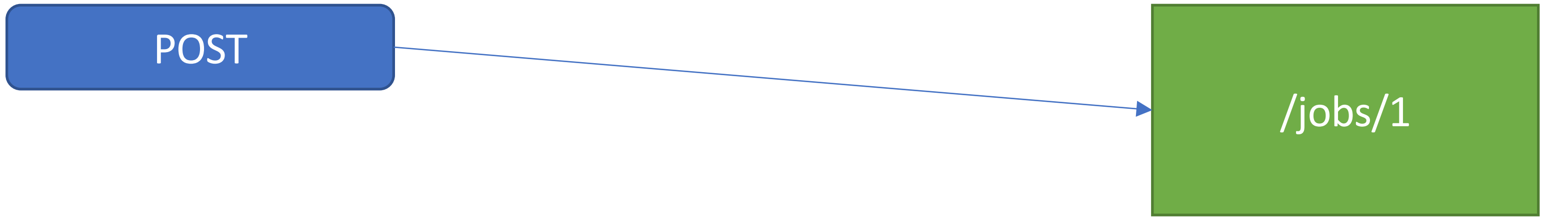
is the JSON object

```
{“status”: “fatal error”}
```

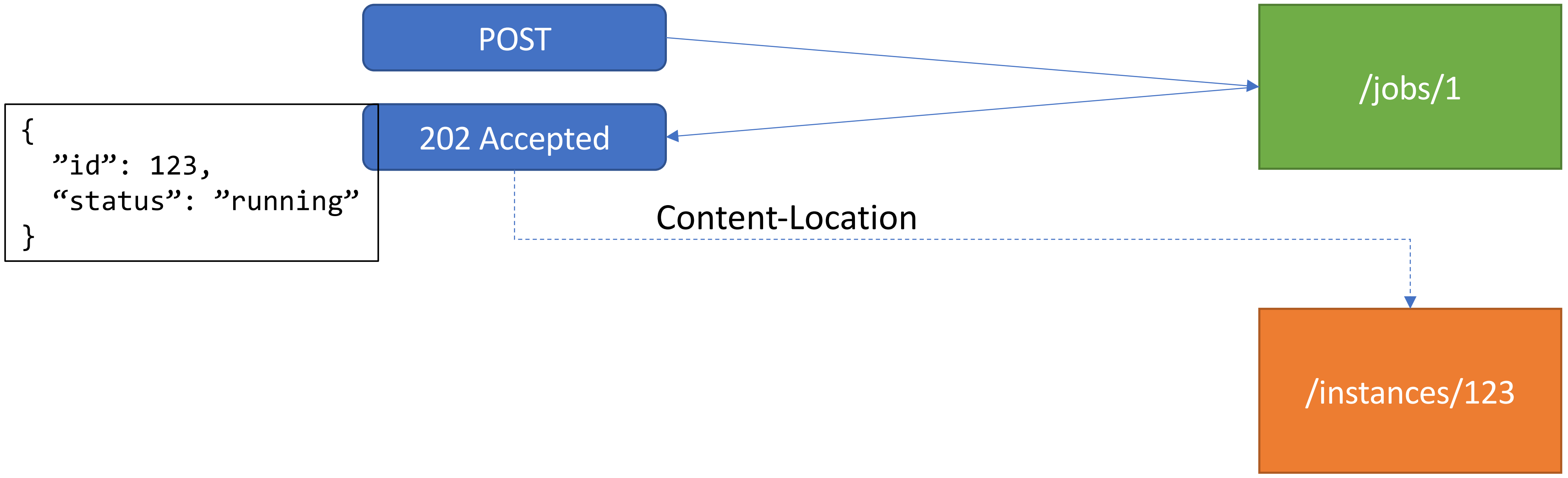
Success comes in various status

- **`201 Created`**
  - Request has been fulfilled and a resource created
  - **`Location`** header contains URI for the created resource
- **`202 Accepted`**
  - Request has been accepted for processing, but has not completed yet
  - The representation sent with this response should describe the request's current status and point to a status monitor

Asynchronous jobs

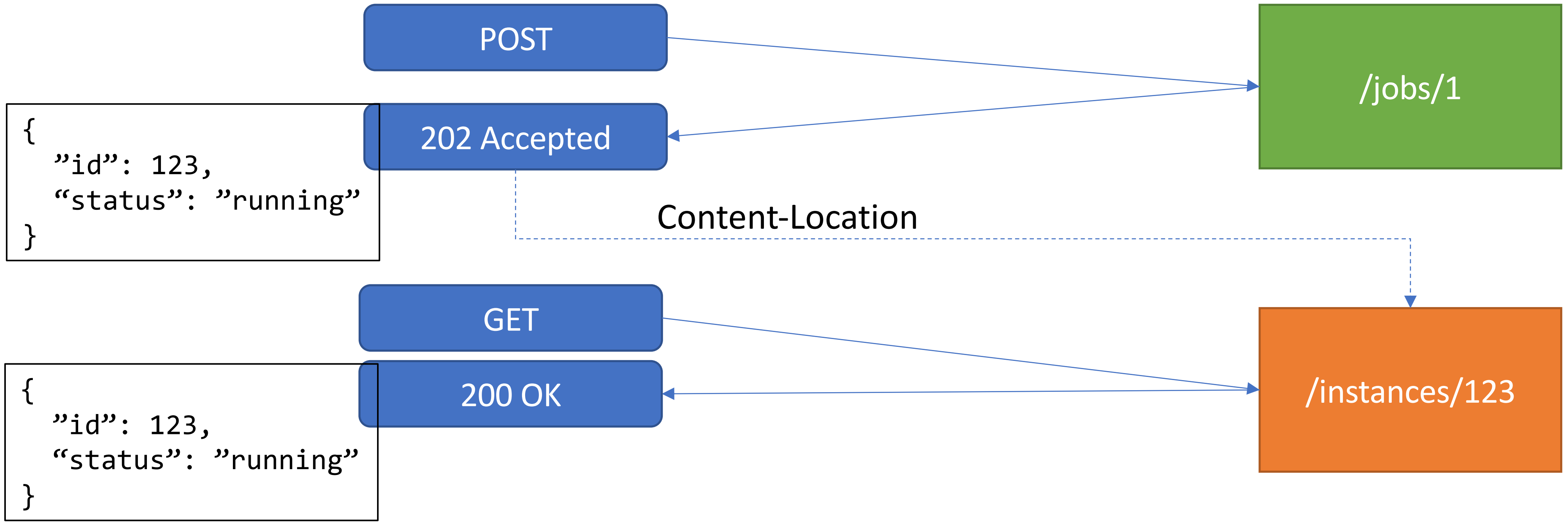


Asynchronous jobs



```
{
  ”id”: 123,
  “status”: ”running”
}
```

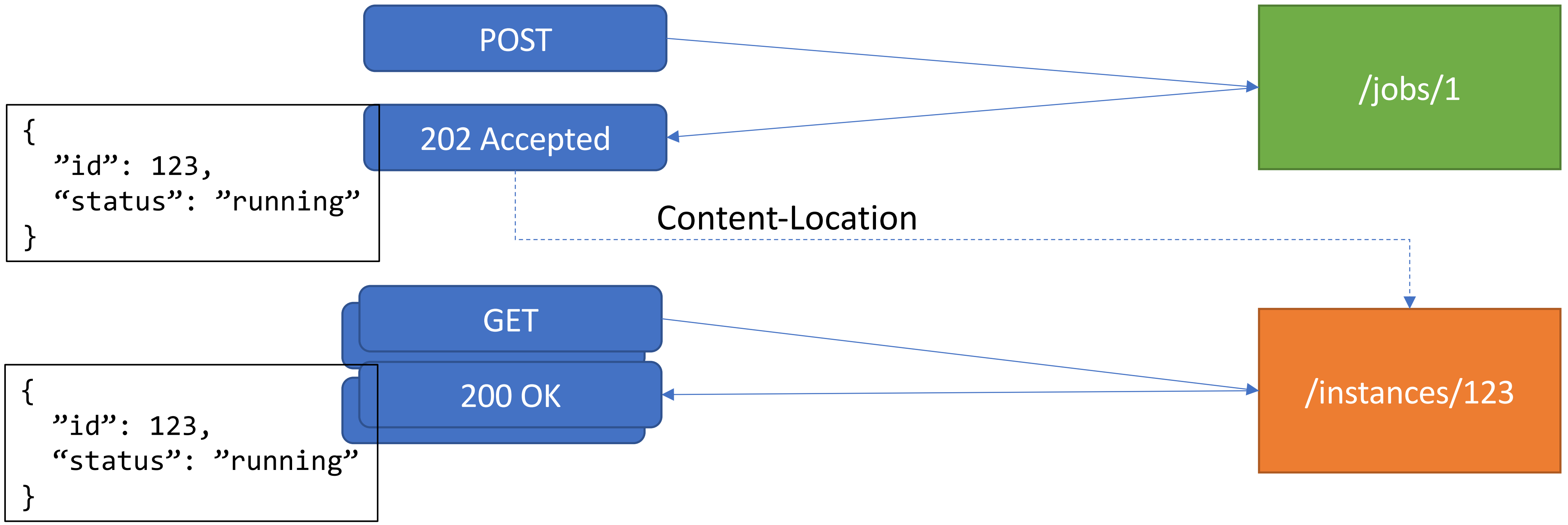
Asynchronous jobs



```
{
  ”id”: 123,
  “status”: ”running”
}
```

```
{
  ”id”: 123,
  “status”: ”running”
}
```

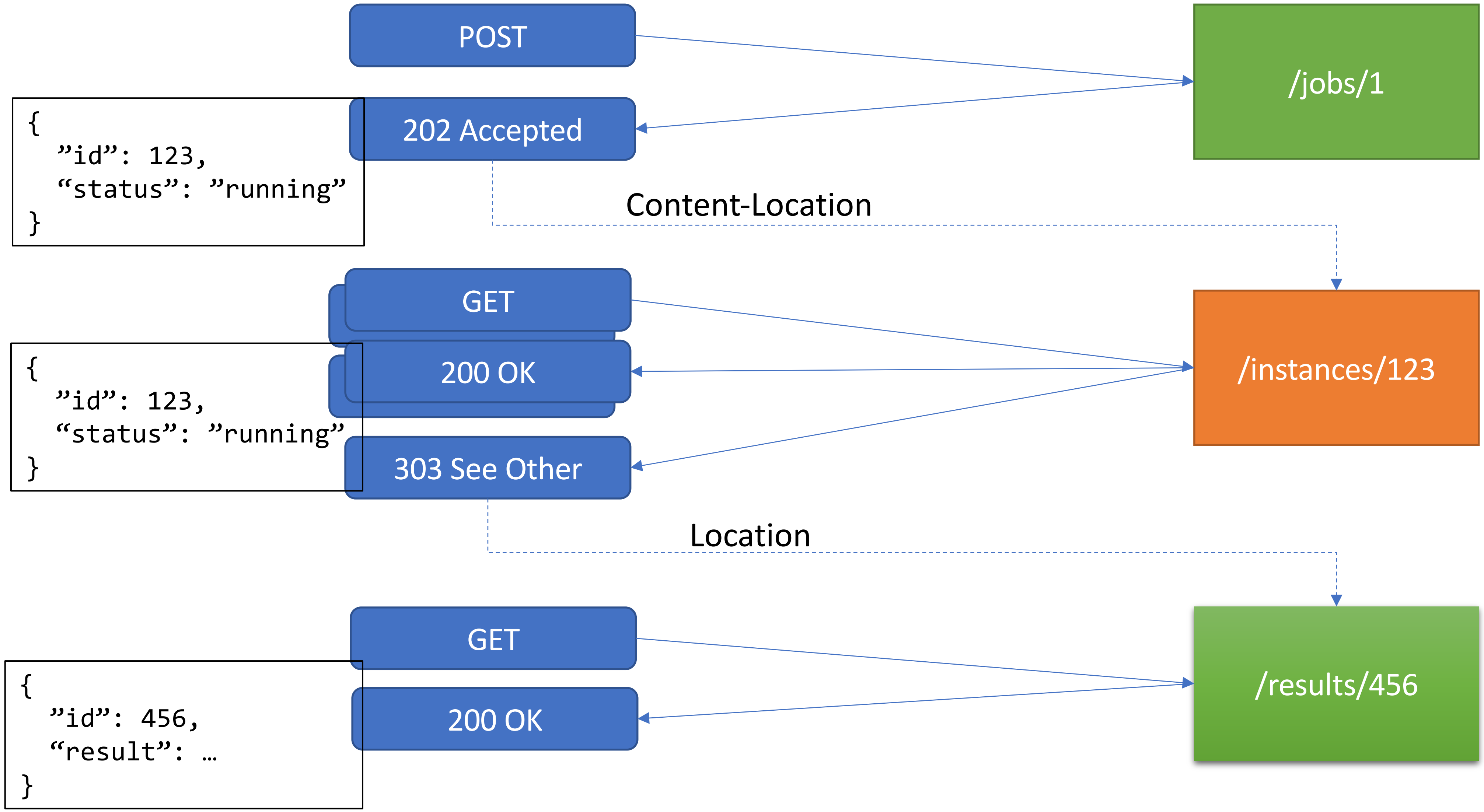
Asynchronous jobs



```
{
  ”id”: 123,
  “status”: ”running”
}
```

```
{
  ”id”: 123,
  “status”: ”running”
}
```

Asynchronous jobs



```
{
  ”id”: 123,
  “status”: ”running”
}
```

```
{
  ”id”: 123,
  “status”: ”running”
}
```

```
{
  ”id”: 456,
  “result”: …
}
```

3xx

There are several types of redirects:

1. Redirects that indicate the **resource might be available at a different URI**, as provided by the Location field, as in the status codes **301 (Moved Permanently)**, **302 (Found)**, and **307 (Temporary Redirect)**.
2. Redirection that offers a choice of matching resources, each capable of representing the original request target, as in the **300 (Multiple Choices)** status code.
3. Redirection to a different resource, identified by the Location field, that can represent an indirect response to the request, as in the **303 (See Other)** status code.
4. Redirection to a previously cached result, as in the **304 (Not Modified)** status code.

5. In https://tools.ietf.org/html/rfc7231#section-6.4

4xx

- **`400 Bad Request`**
  - “server cannot or will not process the request due to something that is **perceived to be a client error** (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).”
- **`401 Unauthorized`**
  - “request has not been applied because it **lacks valid authentication credentials** for the target resource.”
  - Missing or invalid credentials
- **`403 Forbidden`**
  - “The 403 (Forbidden) status code indicates that the **server understood the request** but **refuses to authorize it**” (…) “If **authentication credentials were provided** in the request, the server considers them **insufficient to grant access**”
- **`404 Not Found`**
  - “The 404 (Not Found) status code indicates that the origin server **did not find a current representation** for the target resource or **is not willing to disclose** that one exists”

- **`405 Method Not Allowed`**
  - “indicates that the method received in the request-line is known by the origin server but not supported by the target resource” (…) “The origin server MUST generate an Allow header field in a 405 response containing a list of the target resource's currently supported methods”
- **`406 Not Acceptable`**
  - “indicates that the target resource **does not have a current representation** that would be **acceptable to the user agent**, according to the **proactive negotiation** header fields”

- There aren’t HTTP status code for all possible failure scenarios.
- Uniform interface - status code don’t have domain-specific semantics.
- What to do when needing to provide more information.
- Two common **anti-patterns** are:
    - **Redefining the meaning** of standard codes for a specific set of resources.
    - Using an **unassigned status code** in the 4xx or 5xx classes.

- A solution is to add an **error representation** on the **response body**

Application/problem+json (RFC 7807)

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
  "type": "https://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "/account/12345/msgs/abc",
  "balance": 30,
  "accounts": [
    "/account/12345",
    "/account/67890”
  ]
}
```

Application/problem+json (RFC 7807)

A new format just to represent errors

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
  "type": "https://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "/account/12345/msgs/abc",
  "balance": 30,
  "accounts": [
    "/account/12345",
    "/account/67890”
  ]
}
```

Application/problem+json (RFC 7807)

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
  "type": "https://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "/account/12345/msgs/abc",
  "balance": 30,
  "accounts": [
    "/account/12345",
    "/account/67890”
  ]
}
```

Application/problem+json (RFC 7807)

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
  "type": "https://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "/account/12345/msgs/abc",
  "balance": 30,
  "accounts": [
    "/account/12345",
    "/account/67890”
  ]
}
```