

See everything available through O'Reilly Online Learning and !

Search

Designing Evolvable Web APIs with ASP.NET by Darrel Miller, Glen...

◀ [PREV](#)
B. HTTP Headers

[NEXT](#) ▶
D. Caching in Action

Appendix C. Content Negotiation

There are two types of content negotiation (conneg): proactive and reactive.

Proactive Negotiation

This type of negotiation occurs when the server is responsible for selection and contains logic that executes per request in order to find the best representation. It makes the selection based on matching up against client preferences or additional headers and the server's available representations. The client expresses its preference through the previously mentioned `Accept*` headers (see [Table B-2](#)). Each of these headers allows for sending multiple values or ranges along with a *qualifier* (also known as a *q-value*) that contains prioritization. The server can use additional fields, though, like `User-Agent` or any other.

If the server determines that the client hasn't sent it enough information to make a selection it can make a default selection, return a status 406 Not

Acceptable, or perform *reactive negotiation* (see next section). Once it makes the selection the server should return to the client the chosen representation. The response should include a Vary header, which indicates exactly which header fields were used to make the selection. The server can also include a Content-Location header containing the URI of the negotiated content. It is important to remember that the server is not bound by the client preferences, but it should try to adhere to them as much as it can.

Figure C-1 indicates the steps of the process.

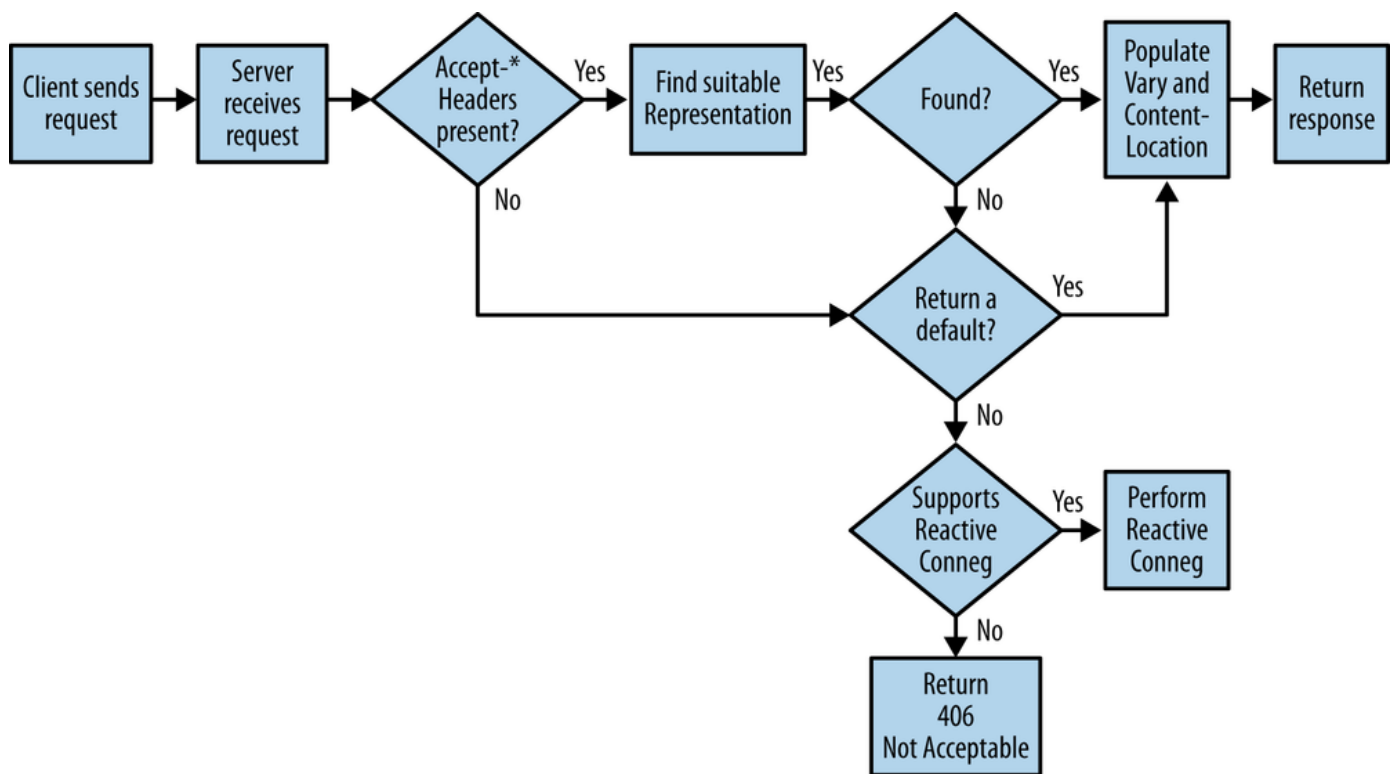



Figure C-1. Proactive conneg

Notice in the figure that in cases where the client has not sent client preferences, or a suitable match cannot be found, the server at its discretion can either return a default representation or return a 406 Not Acceptable response.

Web browsers conventionally use this type of negotiation. Whenever you make a request to a server, your browser sends a list of preferences of things that it supports. In some cases, it may send additional media types that are supported via browser plug-ins. The following is a request using Chrome; notice the various Accept headers that the browser is sending. Different browsers will also have different preferences.

```
GET http://www.yahoo.com/ HTTP/1.1
Host: www.yahoo.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
Accept-Encoding: gzip,deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```



Reactive Negotiation

With this type of negotiation (also referred to as *agent-driven negotiation*), the choice of selection is moved to the client. The way it works is that when the client sends a request to the server for a resource, the server returns a list of representations with a status code of 300 Multiple Choices. The client then chooses from the list based on its own logic and then sends a second request to get the selected representation.

This flow is depicted in [Figure C-2](#).

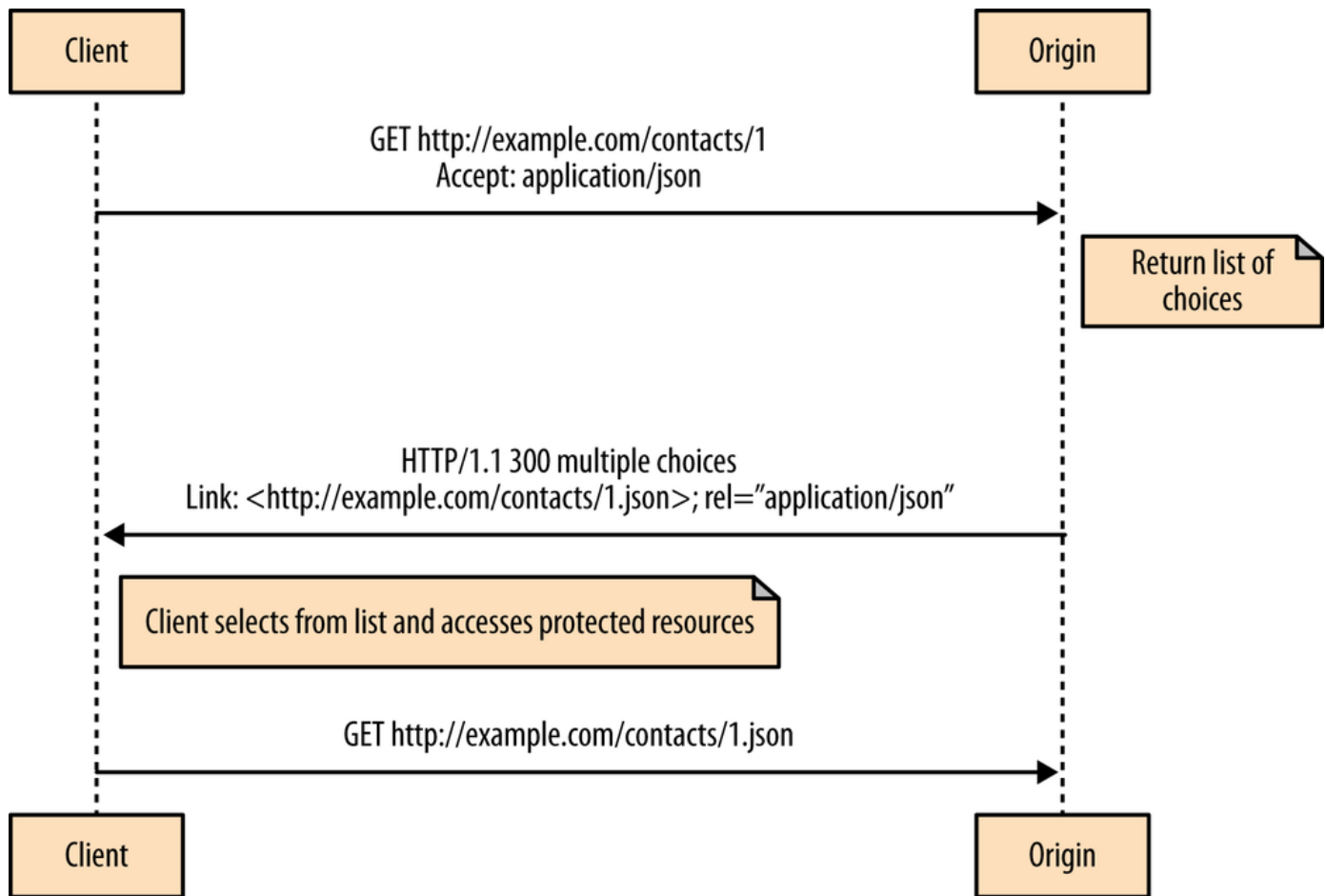


Figure C-2. Reactive conneg

As to the representation itself, which contains the choices, the spec is not at all prescriptive. Mike Amundsen has a nice article on this type of negotiation called "[Agent-Driven conneg in HTTP.](#)"

In his post he recommends several different fully supported approaches. One approach is to return an XHTML representation using `<a hrefs>` for each option, as in the following example:

```

HTTP/1.1 300 Multiple Choices
Host: www.example.org
Content-Type: application/xhtml
Content-Length:XXX
  
```

`<p>`

```
Select one:
</p>
<a href="/results/fr" hreflang="fr">French</a>
<a href="/results/en-US" hreflang="en-US">US English</a>
<a href="/results/de" hreflang="de">German</a>
```

An alternative approach is to use Link headers. This has the advantage of being a standard header that any client can understand. Here is an example:

```
HTTP/1.1 300 Multiple Choices
Host: www.example.org
Content-Length: 0
Link: <http://www.example.org/results/png>; type="image/png",
      <http://www.example.org/results/jpeg>;type="image/jpeg",
      <http://www.example.org/results/gif>;type="image/gif"
```

The benefit of using an established mechanism is that any HTTP client can be expected to understand it. You could return `application/json` and just embed the links in JSON, but unless your client has that knowledge out-of-band, it won't know how to parse it. Using the profile header helps because the client can be pointed to a spec that defines the link format (without your having to introduce a new media type). In this example, the profile document would specify to use an alternate JSON array for the list:

```
HTTP/1.1 300 Multiple Choices
Host: www.example.org
Content-Type: application/json
Content-Length: XXX
Link: <http://www.example.org/profile>; rel="profile"

{
  "alternates" : [
    {"href": "http://www.example.org/results/png", "type": "image/png"},
    {"href": "http://www.example.org/results/jpeg", "type": "image/jpe"},
    {"href": "http://www.example.org/results/gif", "type": "image/gif"}
  ]
}
```

```
]
}
```

Get *Designing Evolvable Web APIs with ASP.NET* now with
O'Reilly online learning.

O'Reilly members experience live online training, plus books,
videos, and digital content from 200+ publishers.

START YOUR FREE TRIAL

UPCOMING CONFERENCES

Artificial Intelligence Conference
Open Source Software Conference
Software Architecture Conference
Strata Data Conference
TensorFlow World
Velocity Conference

THE O'REILLY APPROACH

Our Company
Teach/Speak/Write
Careers
Community Partners

SOLUTIONS

For Teams

For Enterprise

For Individuals

For Government

For Education

SUPPORT

Customer Service

Contact Us

Privacy Policy



DOWNLOAD THE O'REILLY APP



Take O'Reilly online learning with you and learn anywhere, anytime on your phone or tablet. Download the app today and:

- Get unlimited access to books, videos, and live training
- Never lose your place—all your devices are synced
- Learn during your commute with online and offline access

O'REILLY®

© 2020, O'Reilly Media, Inc. All trademarks and registered trademarks appearing on oreilly.com are the property of their respective owners.

[Terms of Service](#) • [Privacy Policy](#) • [Editorial Independence](#)