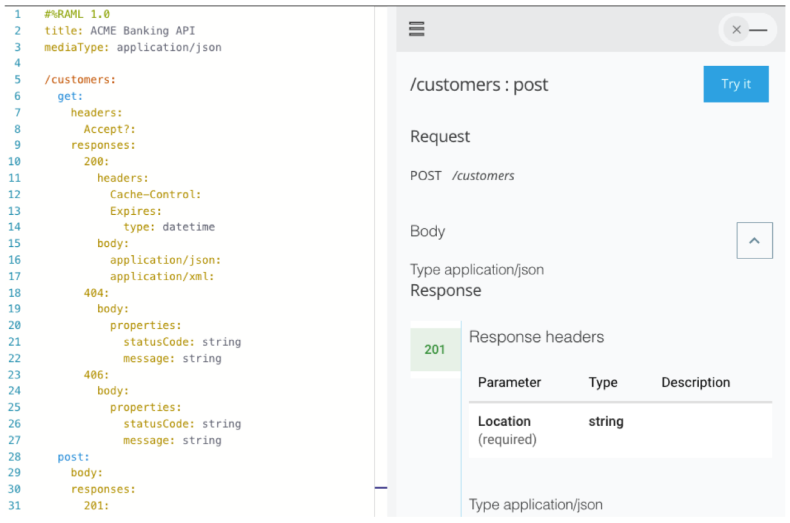
MuleSoft

Module 5: Specifying Responses

Goal

MuleSoft



```
#%RAML 1.0
title: ACME Banking API
mediaType: application/json

/customers:
  get:
    headers:
      Accept?:
    responses:
      200:
        headers:
          Cache-Control:
          Expires:
            type: datetime
        body:
          application/json:
          application/xml:
      404:
        body:
          properties:
            statusCode: string
            message: string
      406:
        body:
          properties:
            statusCode: string
            message: string
  post:
    body:
    responses:
      201:
```

/customers : post

Request

POST */customers*

Body

Type application/json

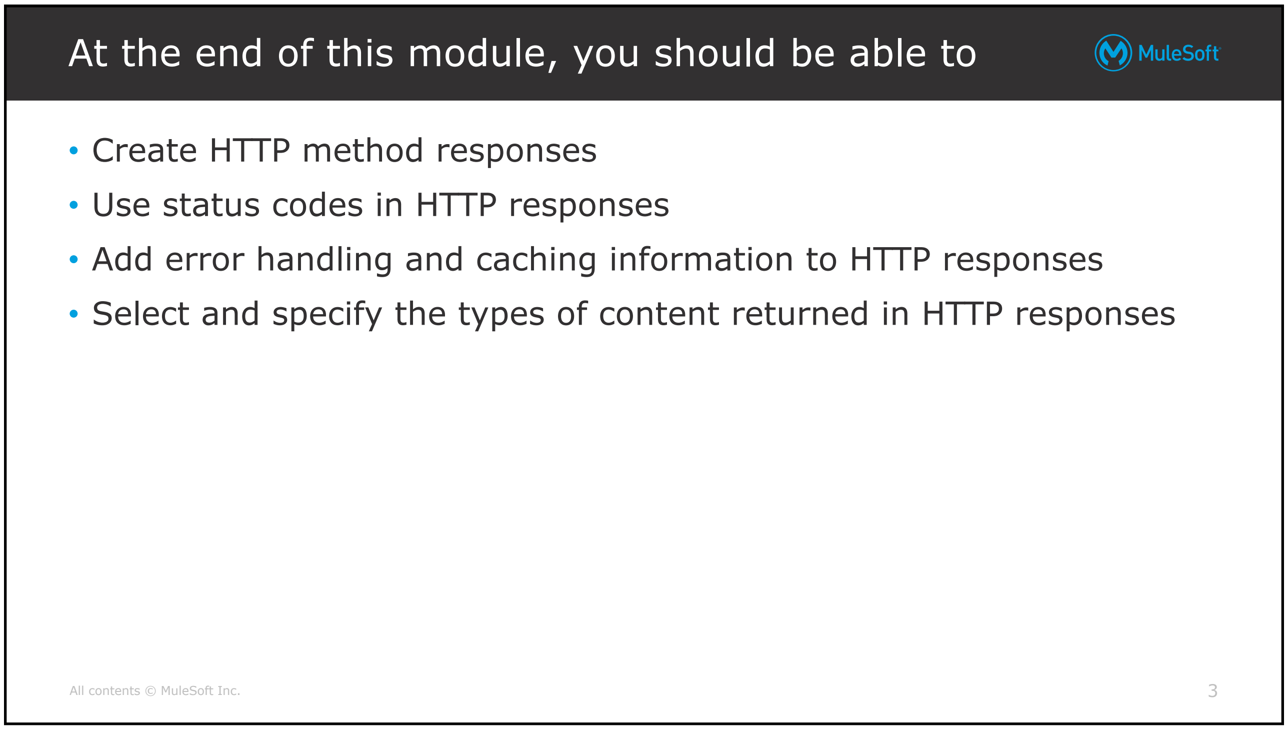
Response

201

Response headers

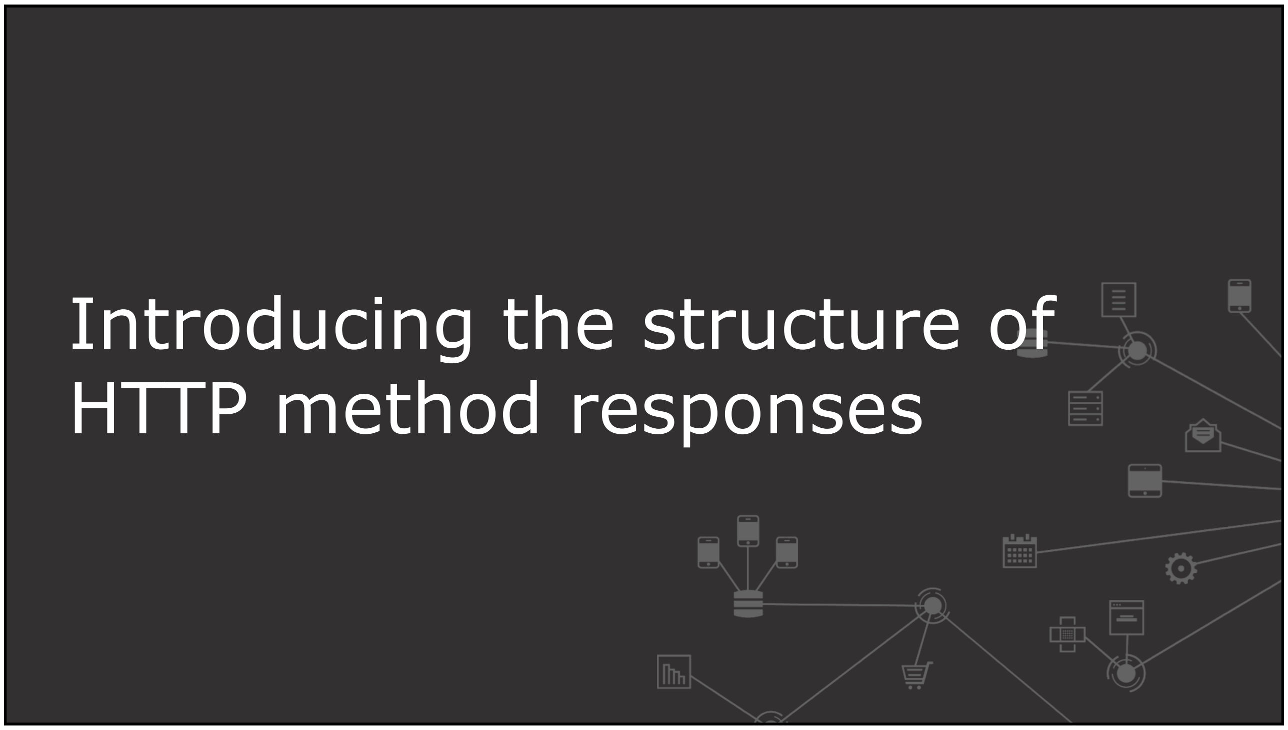
Parameter	Type	Description
Location (required)	string	

Type application/json

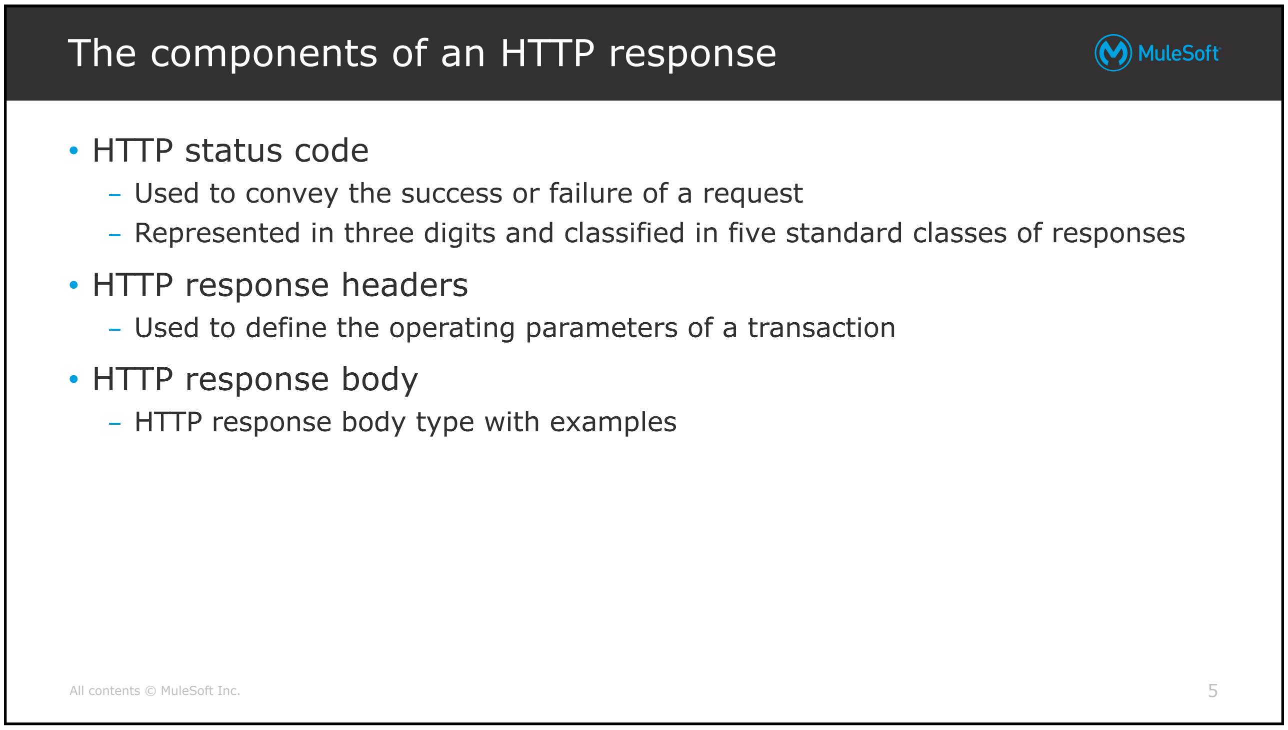


At the end of this module, you should be able to

- Create HTTP method responses
- Use status codes in HTTP responses
- Add error handling and caching information to HTTP responses
- Select and specify the types of content returned in HTTP responses

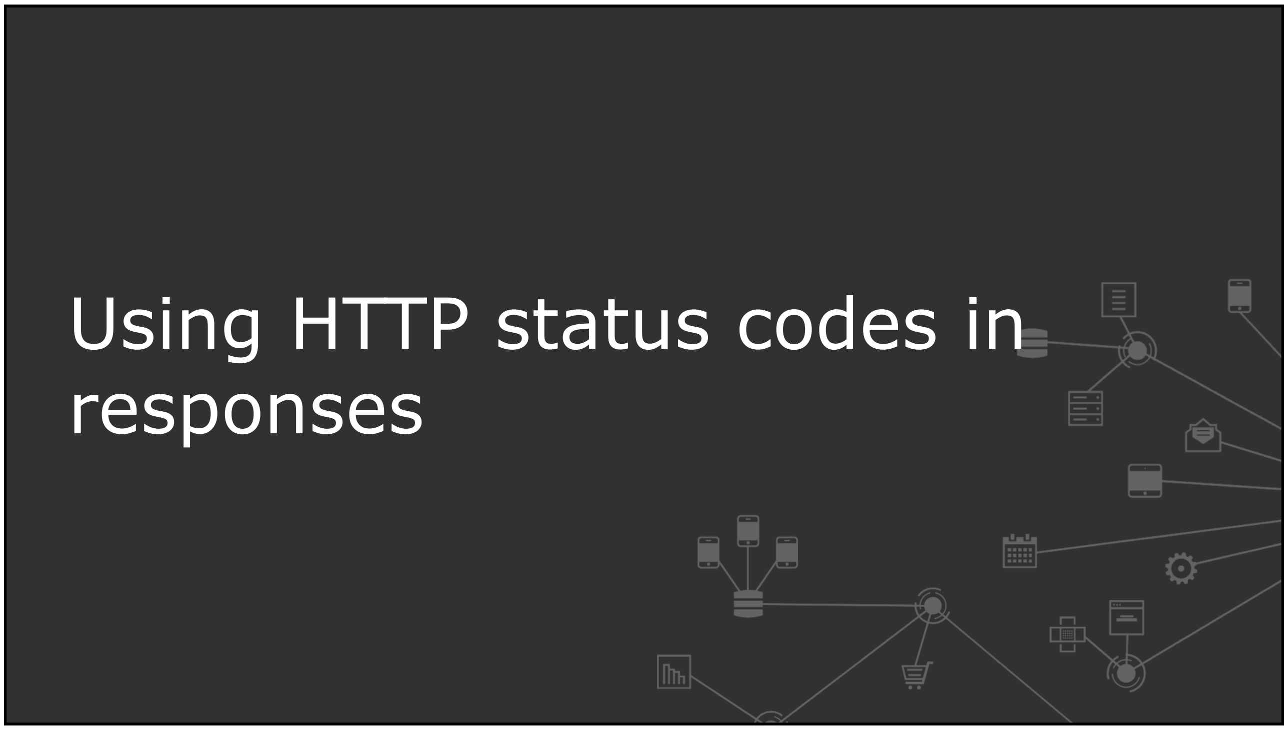


Introducing the structure of HTTP method responses

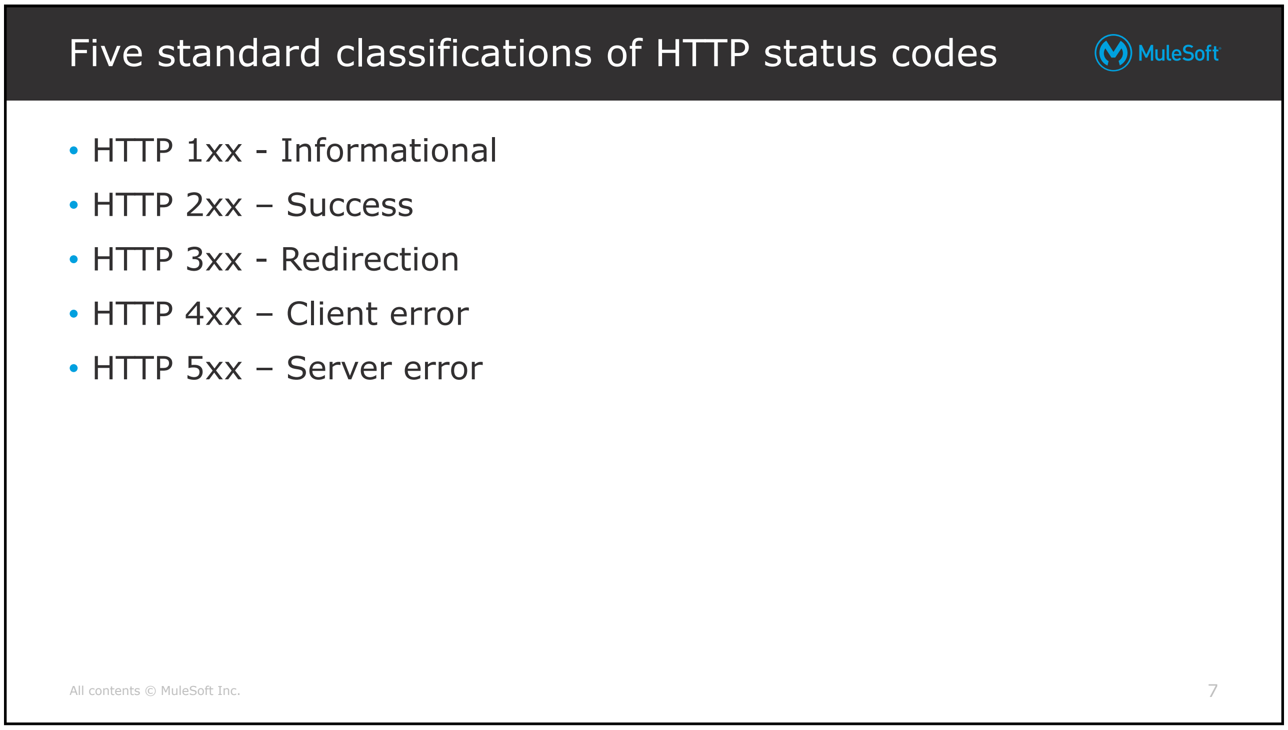


The components of an HTTP response

- HTTP status code
  - Used to convey the success or failure of a request
  - Represented in three digits and classified in five standard classes of responses
- HTTP response headers
  - Used to define the operating parameters of a transaction
- HTTP response body
  - HTTP response body type with examples

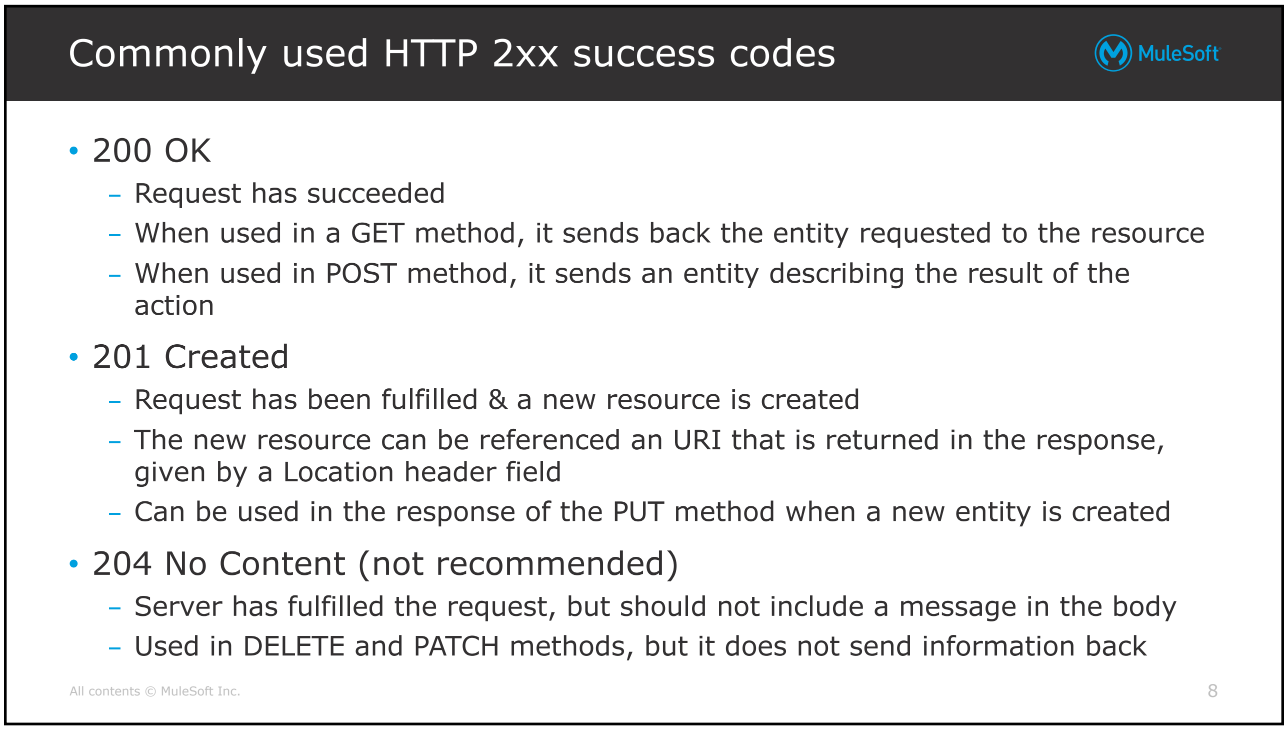


Using HTTP status codes in responses



Five standard classifications of HTTP status codes

- HTTP 1xx - Informational
- HTTP 2xx – Success
- HTTP 3xx - Redirection
- HTTP 4xx – Client error
- HTTP 5xx – Server error

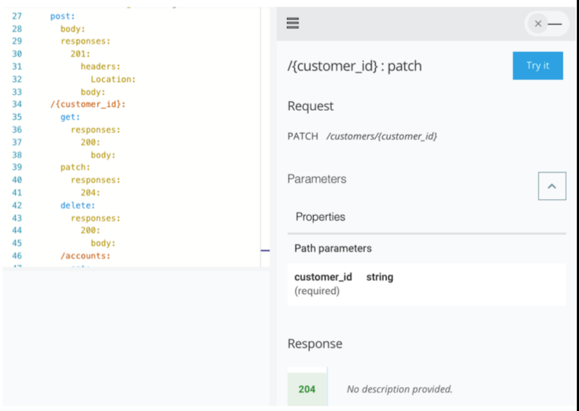


Commonly used HTTP 2xx success codes

- 200 OK
  - Request has succeeded
  - When used in a GET method, it sends back the entity requested to the resource
  - When used in POST method, it sends an entity describing the result of the action
- 201 Created
  - Request has been fulfilled & a new resource is created
  - The new resource can be referenced an URI that is returned in the response, given by a Location header field
  - Can be used in the response of the PUT method when a new entity is created
- 204 No Content (not recommended)
  - Server has fulfilled the request, but should not include a message in the body
  - Used in DELETE and PATCH methods, but it does not send information back

Walkthrough 5-1: Add HTTP 2xx responses to GET methods

- Define media type for the API resource methods
- Add a HTTP 200 response body to all GET and DELETE methods indicating success of the HTTP request
- Add a HTTP 201 response body to all POST methods
- Add a response body with both HTTP 200 and 201 status codes to the PUT method
- Add a HTTP 204 response body to PATCH methods



```
    post:
      body:
      responses:
        201:
          headers:
            Location:
          body:
    /{customer_id}:
      get:
        responses:
          200:
            body:
      patch:
        responses:
          204:
      delete:
        responses:
          200:
            body:
      /accounts:
```

/{customer_id} : patch

Request

PATCH /customers/{customer_id}

Parameters

Properties

Path parameters

customer_id string (required)

Response

204 *No description provided.*

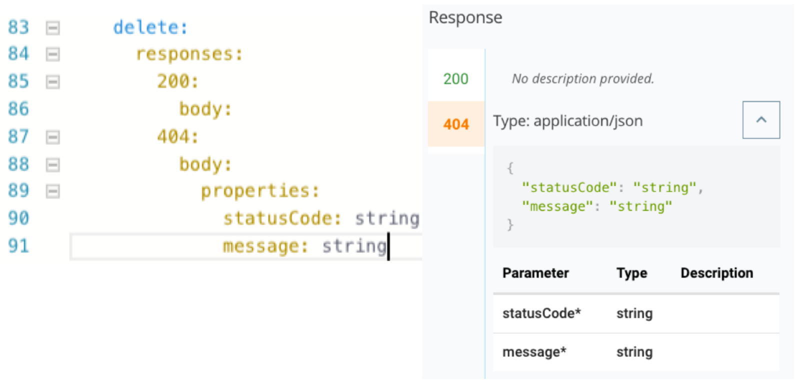
Specifying responses for errors

Commonly used client-side and server-side error codes

- HTTP 4xx
  - 404 Not Found
    - The requested resource could not be found but may be available again in the future
    - Subsequent requests by the client are permissible
- HTTP 5xx
  - 501 Not Implemented
    - The server does not recognize the request method and is not capable of supporting it for any resource
    - APIs are moving towards using PATCH methods but that might not be supported by the backend system; then this error code is returned
  - 503 Service Unavailable
    - The server is currently unable to handle the request due to a temporary overloading or maintenance
    - Length of the delay can be indicated in a Retry-After header, if it is known

Walkthrough 5-2: Add responses bodies to return custom error information for client-side errors

- Add HTTP 4xx status code responses to all GET and DELETE methods
- Create a custom error message object to be returned in the response body



```
delete:
  responses:
    200:
      body:
    404:
      body:
        properties:
          statusCode: string
          message: string
```

Response

200 — No description provided.

404 — Type: application/json

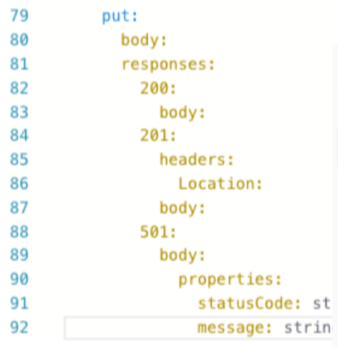
```
{
  "statusCode": "string",
  "message": "string"
}
```

Parameter	Type	Description
statusCode*	string	
message*	string	

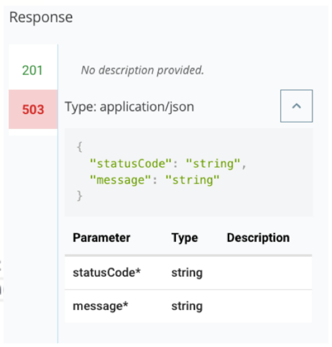
Walkthrough 5-3: Add responses bodies to return error information for server-side errors

MuleSoft

- Add HTTP 5xx status code responses to all PATCH, PUT and POST methods
- Create a custom error message object to be returned in the response body



```
put:
  body:
  responses:
    200:
      body:
    201:
      headers:
        Location:
      body:
    501:
      body:
        properties:
          statusCode: st
          message: strin
```



Response

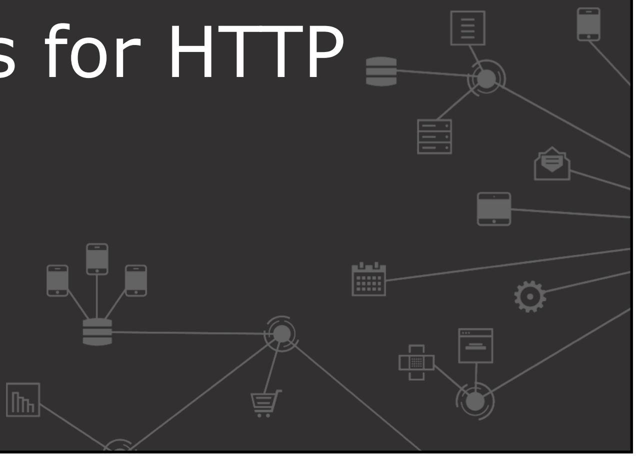
201 *No description provided.*

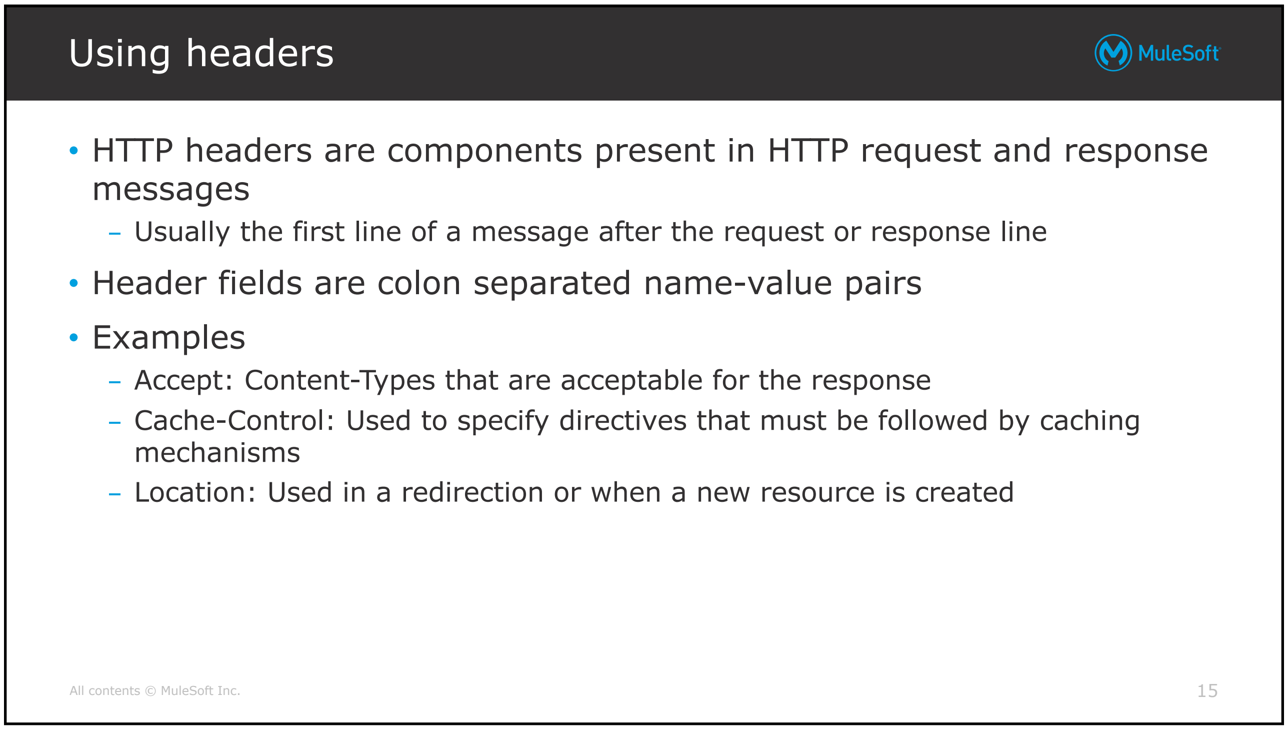
503 Type: application/json

```
{
  "statusCode": "string",
  "message": "string"
}
```

Parameter	Type	Description
statusCode*	string	
message*	string	

Defining headers for HTTP responses





Using headers

MuleSoft

- HTTP headers are components present in HTTP request and response messages
  - Usually the first line of a message after the request or response line
- Header fields are colon separated name-value pairs
- Examples
  - Accept: Content-Types that are acceptable for the response
  - Cache-Control: Used to specify directives that must be followed by caching mechanisms
  - Location: Used in a redirection or when a new resource is created

Supporting multiple media types in responses

Specifying desired content types for responses

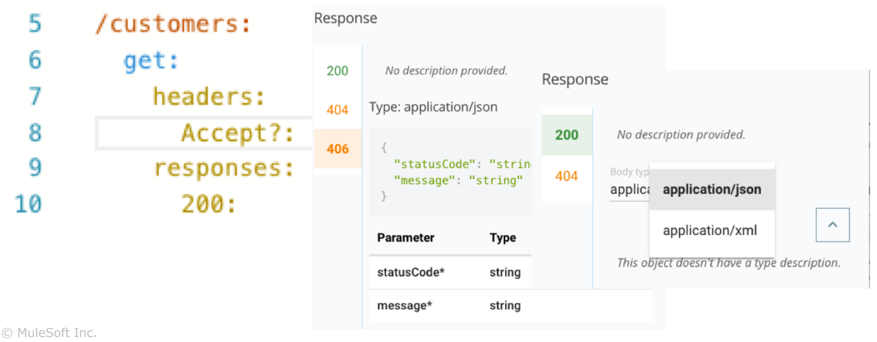
MuleSoft

- Client sends requests with an Accept header
- Accept header is used to specify the desired media type of the response to be returned
  - If the Accept header value is not set, the response body is returned as application/json by default
    - It is also assumed that the client accepts all media types
  - If the value in the header is not supported by the server, it returns an HTTP 406 error

Walkthrough 5-4: Add flexible content-type support to a resource method

MuleSoft

- Add XML body type to the HTTP 200 response of a resource method
- Add an optional accept header to the request to specify the type of response accepted by the client
- Add a relevant HTTP status code for client-side error when an unsupported type is requested



```
/customers:
  get:
    headers:
      Accept?:
    responses:
      200:
```

Response

200 No description provided.
404 Type: application/json
406

```
{
  "statusCode": "strin
  "message": "string"
}
```

Parameter	Type
statusCode*	string
message*	string

Response

200 No description provided.
404

Body typ
applic

application/json
application/xml

This object doesn't have a type description.

Defining client-side caching for responses

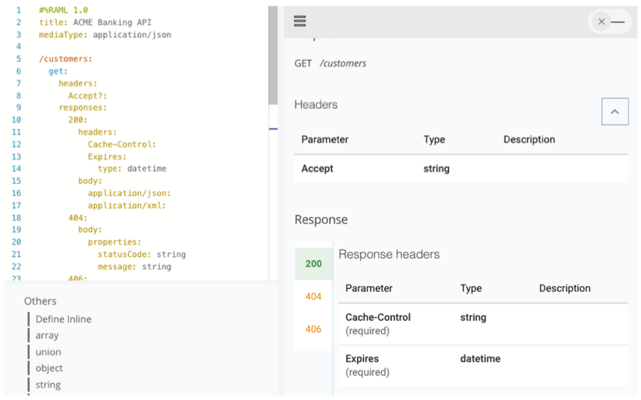
Specifying HTTP headers to help client applications cache information from responses

MuleSoft

- Cache-control
  - Accepts two parameters
    - Private or public depending whether a proxy is accessing the data or not
    - Max-age that sets the expiration time for the cache in milliseconds
- Expires
  - Accepts a datetime attribute that specifies the expiration of the cached data
  - If both Cache-control and Expires headers are used, the max-age property in the cache control header takes precedence

Walkthrough 5-5: Add caching information to HTTP responses

- Add a Cache-Control header to a GET response to enable caching
- Add an Expires header to a GET response to set the date when the cached resource becomes invalid



```
#%RAML 1.0
title: ACME Banking API
mediaType: application/json

/customers:
  get:
    headers:
      Accept?:
    responses:
      200:
        headers:
          Cache-Control:
          Expires:
            type: datetime
        body:
          application/json:
          application/xml:
      404:
        body:
          properties:
            statusCode: string
            message: string
      406:
```

GET /customers

Headers

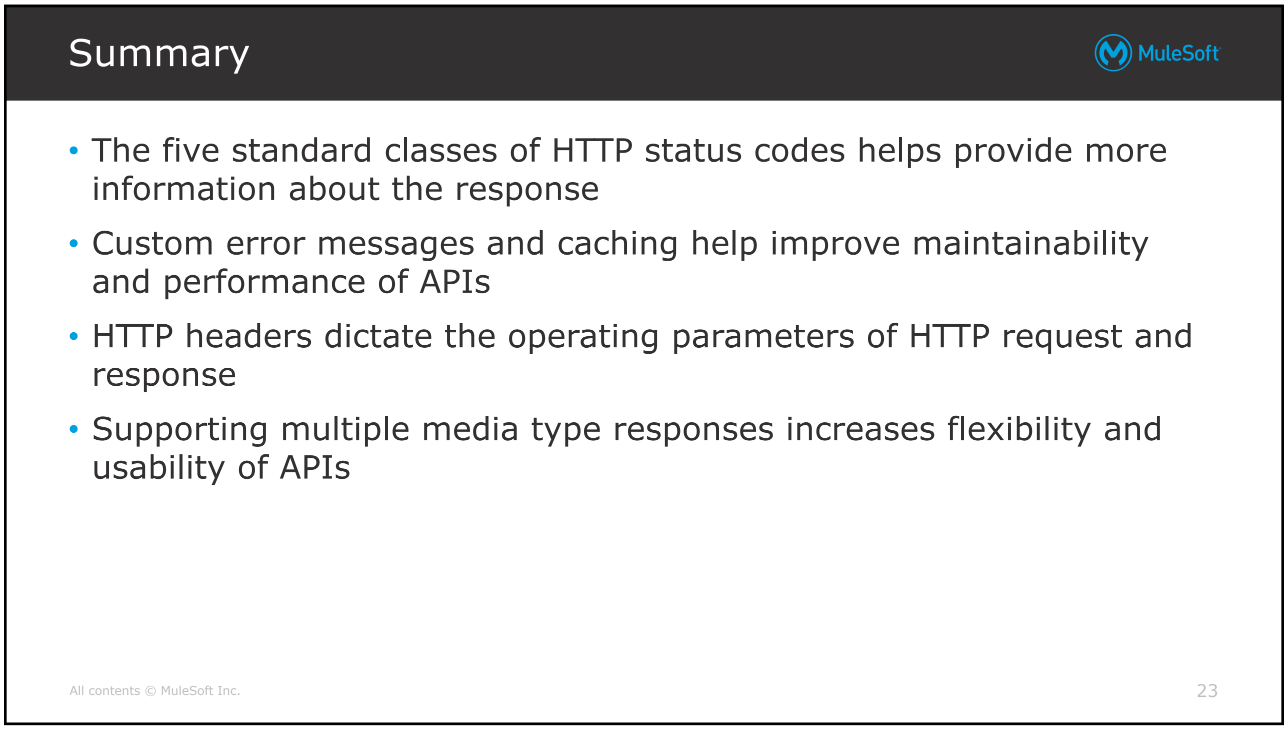
Parameter	Type	Description
Accept	string	

Response

Response headers

Parameter	Type	Description
Cache-Control (required)	string	
Expires (required)	datetime	

Summary



Summary

- The five standard classes of HTTP status codes helps provide more information about the response
- Custom error messages and caching help improve maintainability and performance of APIs
- HTTP headers dictate the operating parameters of HTTP request and response
- Supporting multiple media type responses increases flexibility and usability of APIs