OAuth 2 und OpenID Connect

JAX 2015

23.04.2015, Mainz

Simon Kölsch & Christoph Iserlohn

innoQ

Agenda

- OAuth 1, OAuth 2, OpenID Connect, O...?
- How does it work?
- Use Cases
- Should I use it

“Web 2.0” Platforms

and many more…

“Web 2.0” Platforms

= User content on big platforms with APIs

and many more...

“Web 2.0” Platforms

= User content on big platforms with APIs

2010 - ~ 550.000 “Facebook Apps”

2015 - ~ 30 Flickr Apps tagged “geo”

…

and many more…

The access problem

The access problem

Resource Owner

The access problem

Resource Owner

The access problem

Resource Owner

Client

The access problem

Resource Owner

needs Access

Client

The access problem

Resource Owner

gives Permission

needs Access

Client

Resource/Auth Server

The access problem

Resource Owner

gives Permission

needs Access

checks Permission
provides Access

Client

OAuth o

OAuth o

- Client gets user password

OAuth o

- Client gets user password
- Client impersonates user and gets resource

OAuth 1

- Client registers at the service

OAuth 1

- Client registers at the service
- Client asks resource owner for permission via the service

OAuth 1

- Client registers at the service
- Client asks resource owner for permission via the service
- Client gets a token

OAuth 1

- Client registers at the service
- Client asks resource owner for permission via the service
- Client gets a token
- Client can access the resource with the token

OAuth 1

- Authorization Protocol

OAuth 1

- Authorization Protocol
- RFC 5849 (38 pages spec)

OAuth 1

- Authorization Protocol
- RFC 5849 (38 pages spec)
- One security issue (fixed 2009)

OAuth 1

- Authorization Protocol
- RFC 5849 (38 pages spec)
- One security issue (fixed 2009)
- non-trivial request / response creation

OAuth 1

- Authorization Protocol
- RFC 5849 (38 pages spec)
- One security issue (fixed 2009)
- non-trivial request / response creation
- Scope: “web applications”

Why OAuth 2?

- Simplify client development process

Why OAuth 2?

- Simplify client development process
- Desktop Applications / Single-Page-Apps

Why OAuth 2?

- Simplify client development process
- Desktop Applications / Single-Page-Apps
- Mobile Phones

Why OAuth 2?

- Simplify client development process
- Desktop Applications / Single-Page-Apps
- Mobile Phones
- “Living room devices”

OAuth 2 Roles

- Resource Owner
- Client
- Resource Server
- Authentication Server

OAuth 2 Tokens

- Access Token

  "An access token is a string representing an authorization issued to the client."

OAuth 2 Tokens

- Access Token

  "An access token is a string representing an authorization issued to the client."

- Token Type (e.g. "Bearer")

OAuth 2 Tokens

- Access Token

  "An access token is a string representing an authorization issued to the client."

- Token Type (e.g. "Bearer")

- usually short-living

OAuth 2 Tokens

- Access Token

  "An access token is a string representing an authorization issued to the client."

- Token Type (e.g. "Bearer")
- usually short-living
- Refresh Token

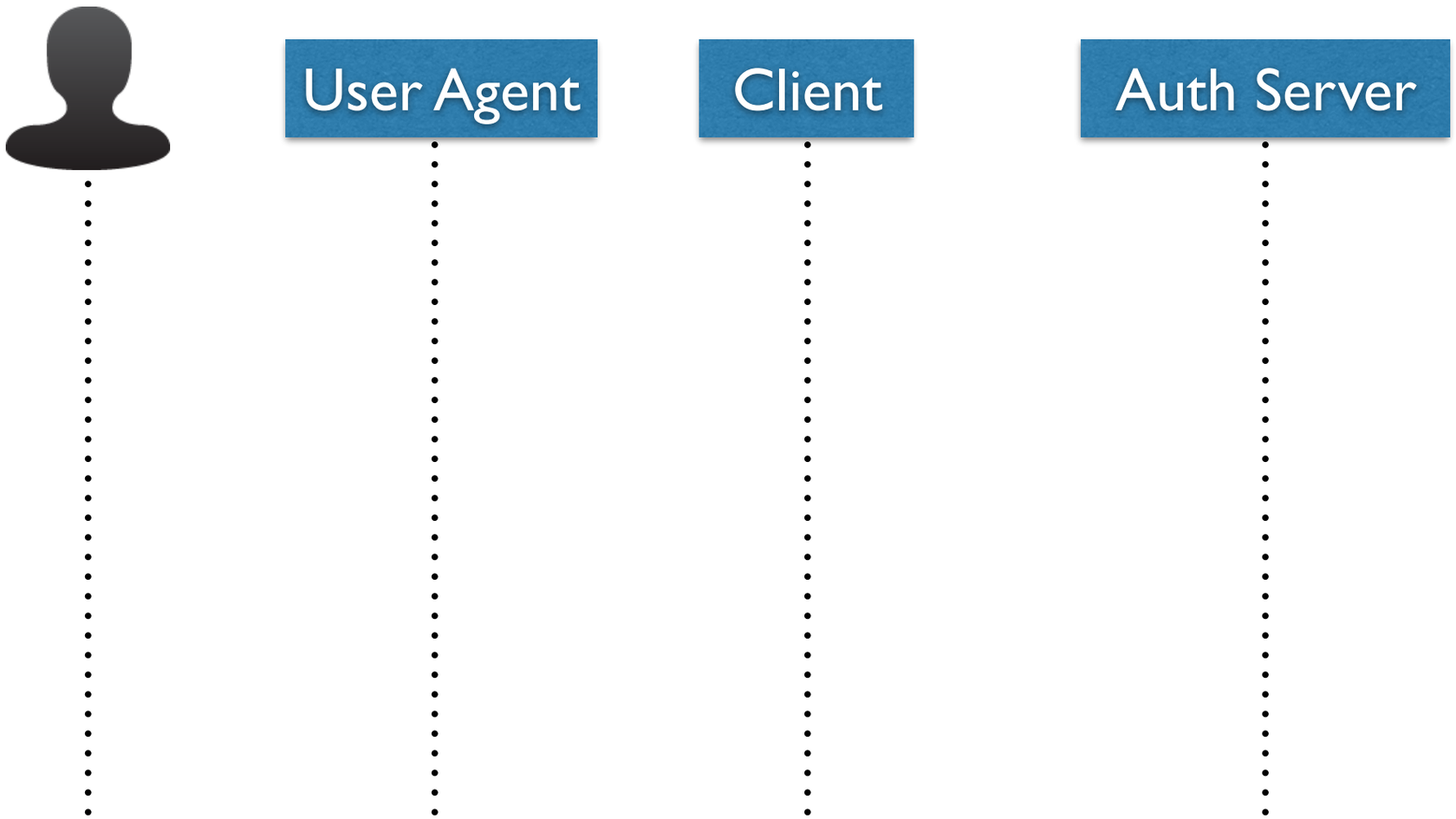
OAuth 2 Tokens

- Access Token

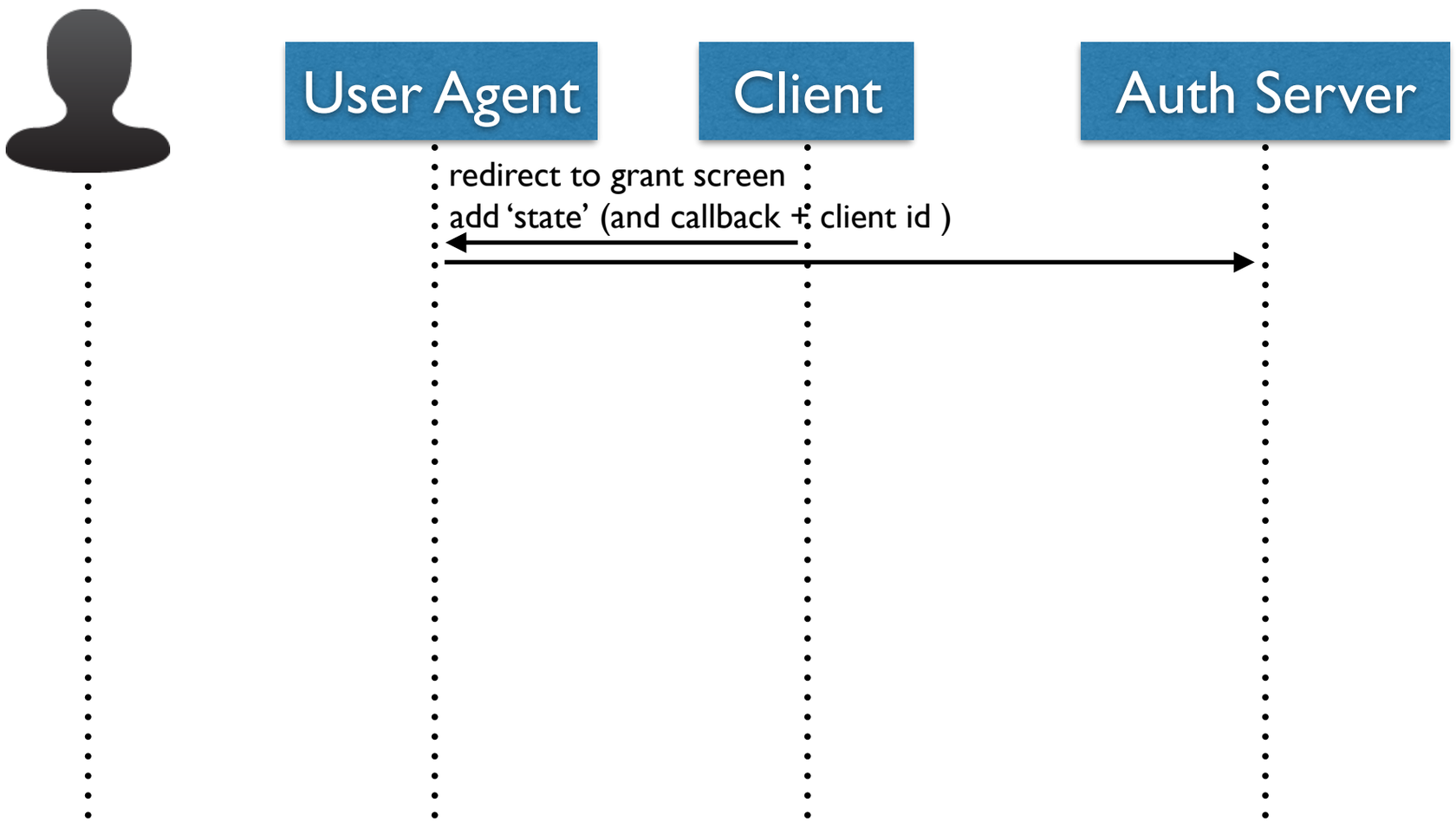
  "An access token is a string representing an authorization issued to the client."

- Token Type (e.g. "Bearer")
- usually short-living
- Refresh Token
- Scope

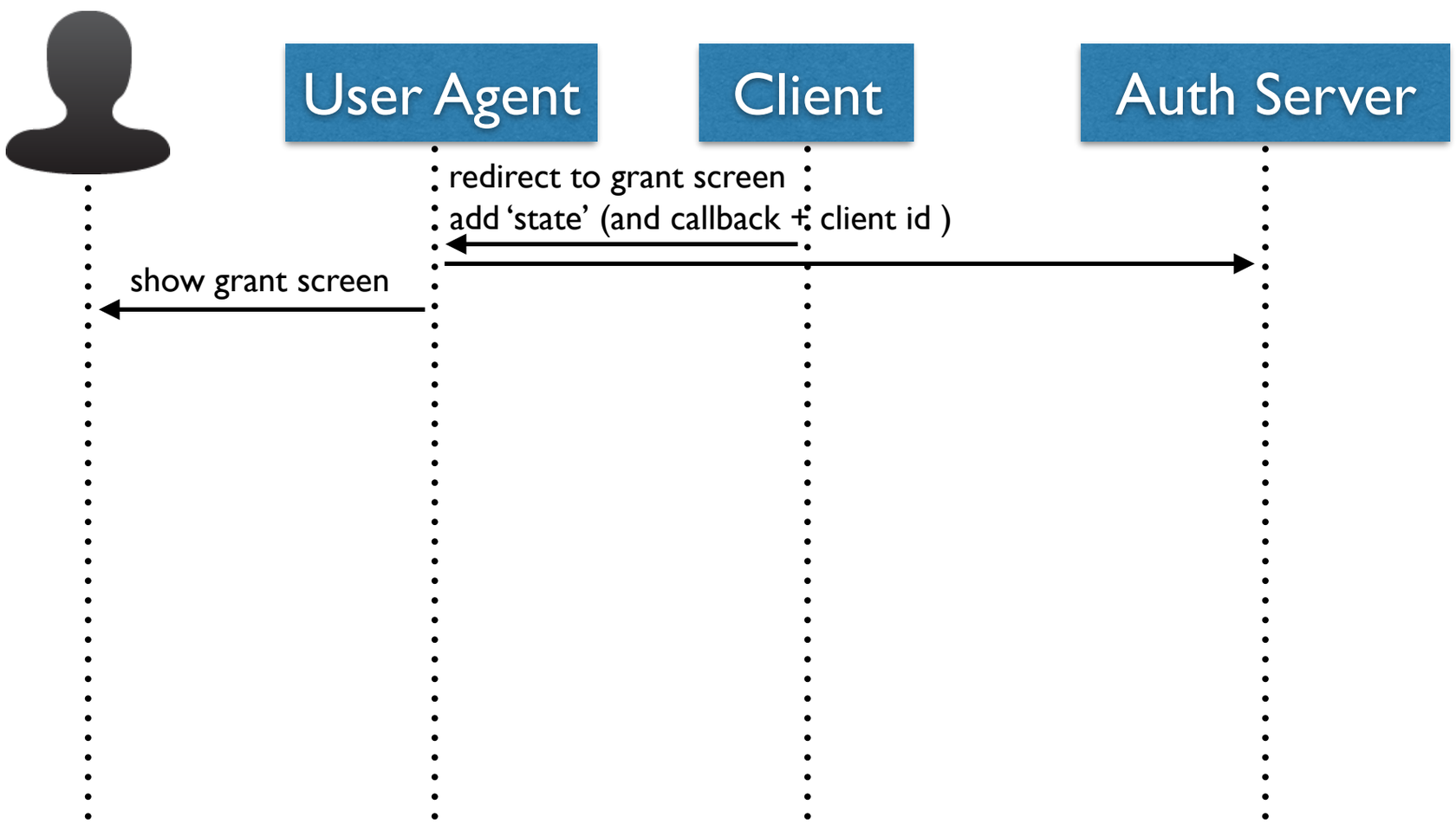
Authorization Code Grant



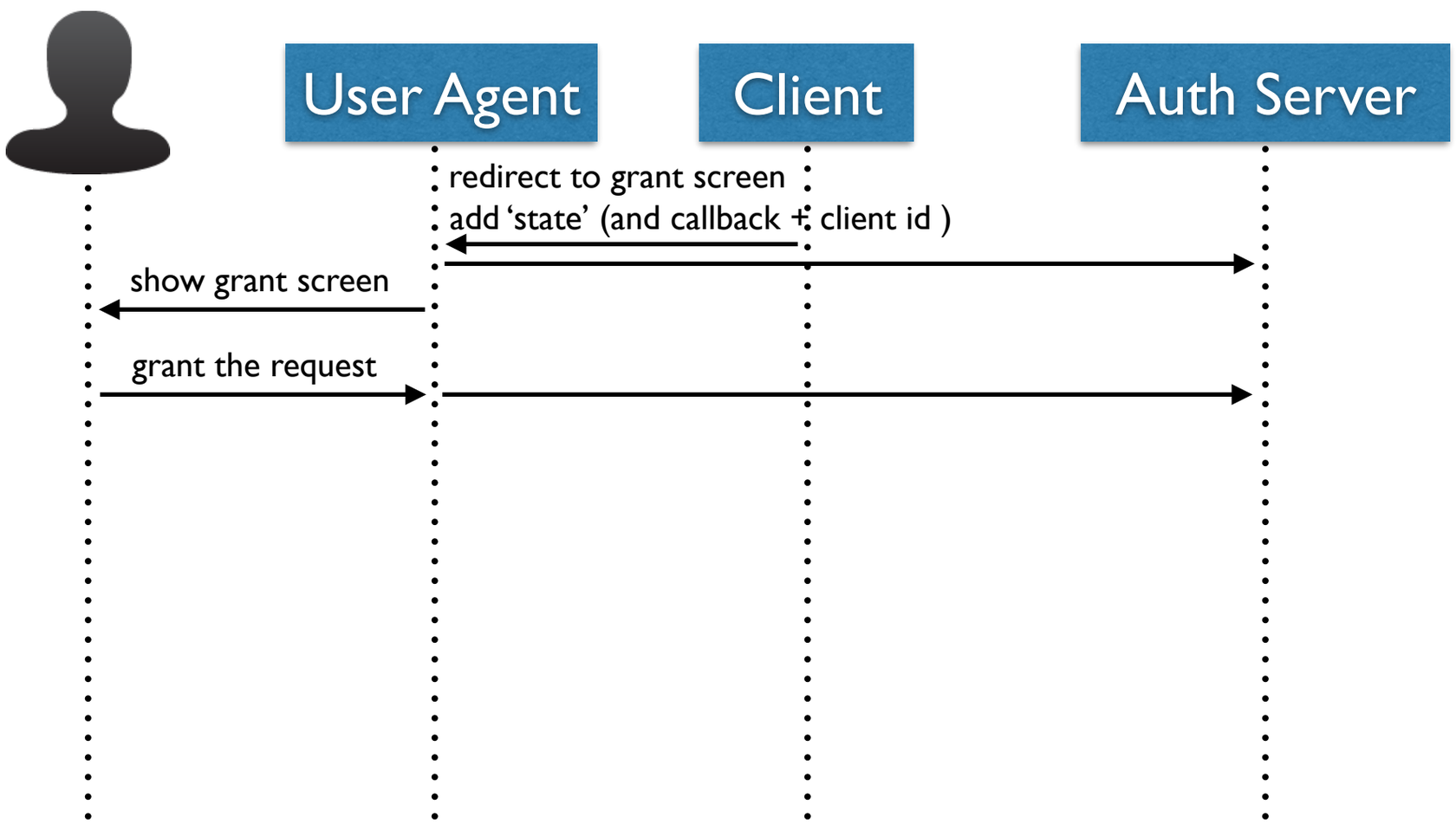
Authorization Code Grant



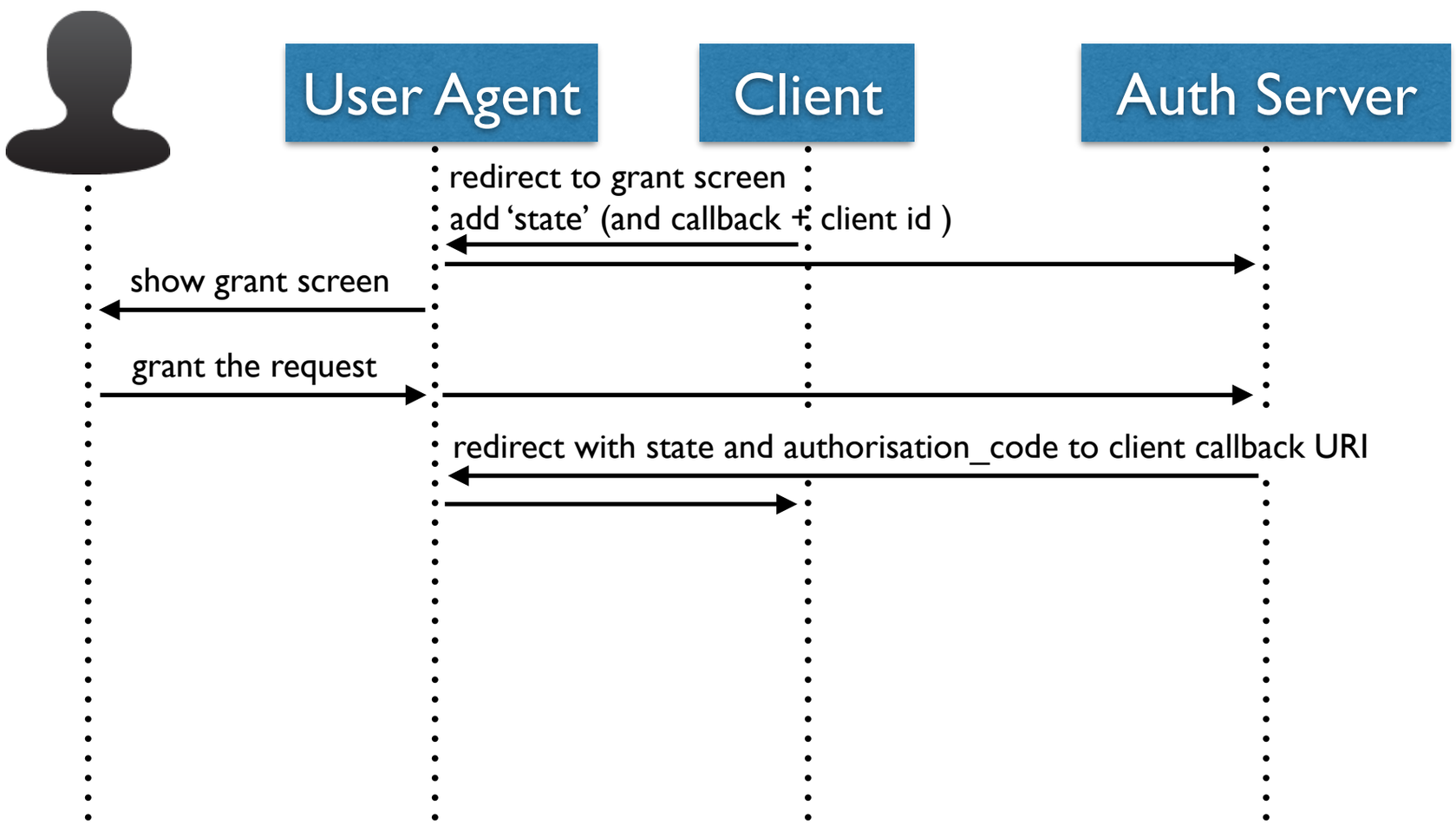
Authorization Code Grant



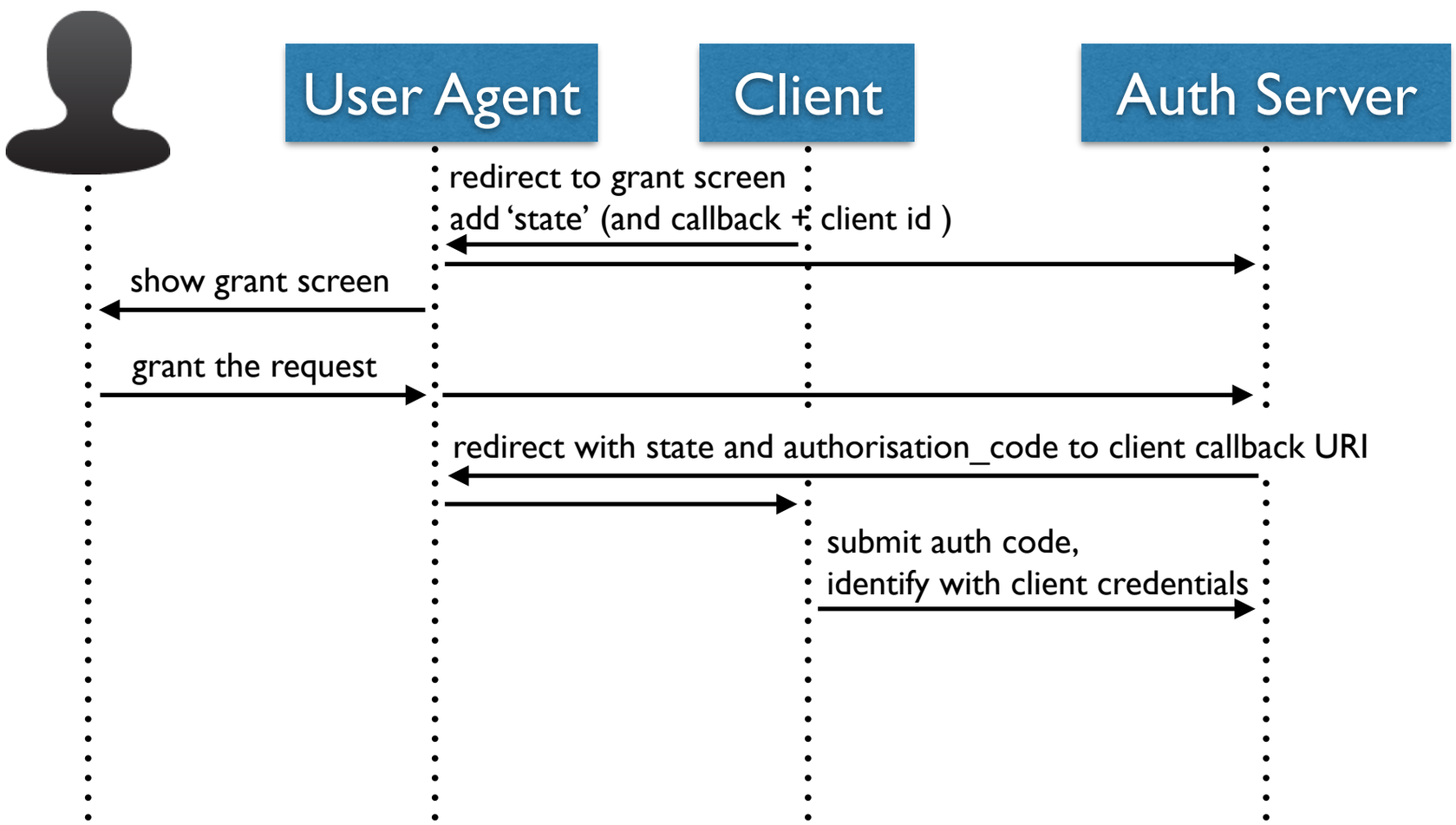
Authorization Code Grant



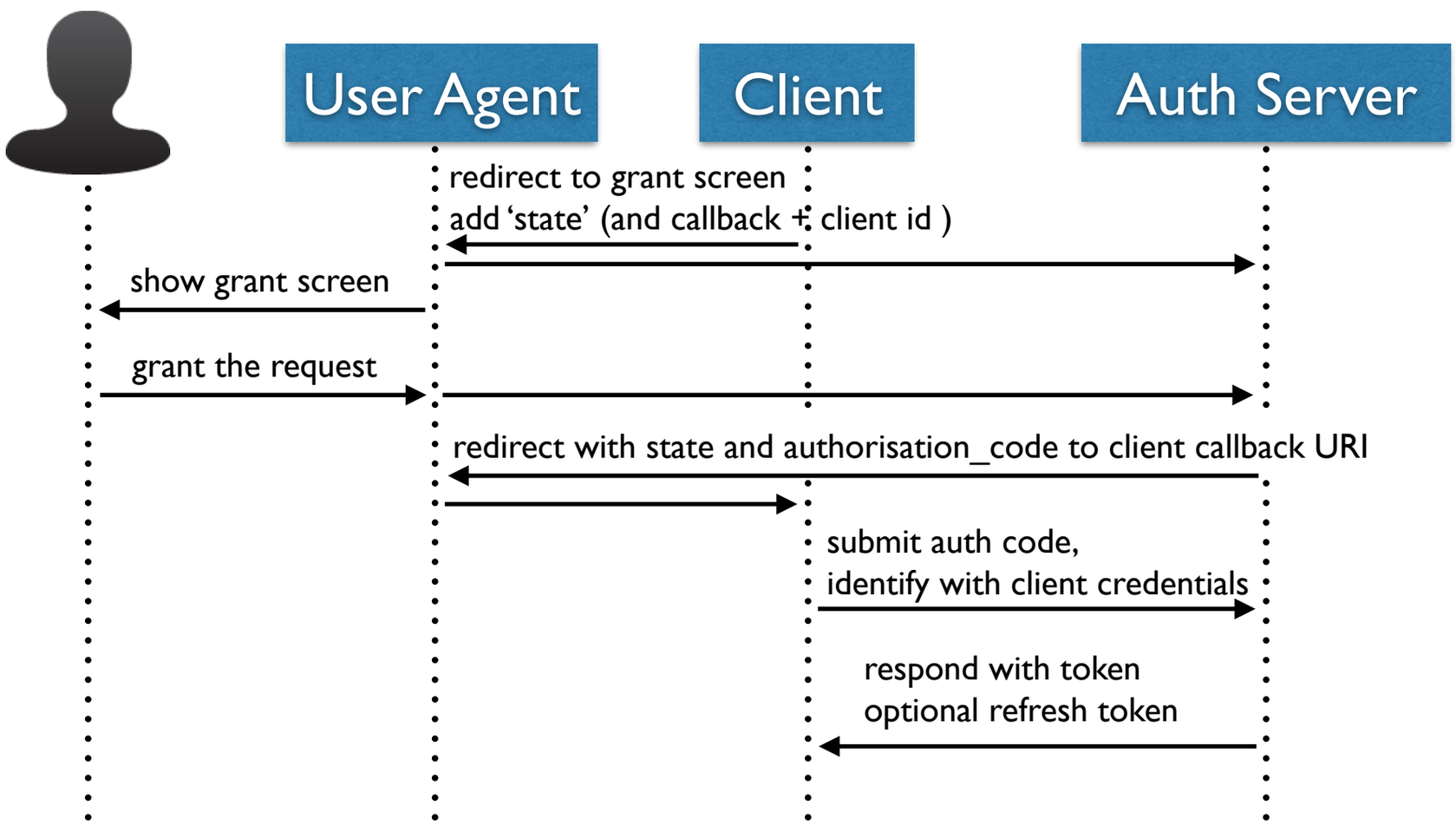
Authorization Code Grant



Authorization Code Grant



Authorization Code Grant



OAuth 2 Grant Types

- Authorization Code

OAuth 2 Grant Types

- Authorization Code
- Implicit

OAuth 2 Grant Types

- Authorization Code
- Implicit
- User Credentials

OAuth 2 Grant Types

- Authorization Code
- Implicit
- User Credentials
- (Client Credentials)

OAuth 2 Spec Overview

Framework

OAuth 2 Spec Overview

Framework

Security

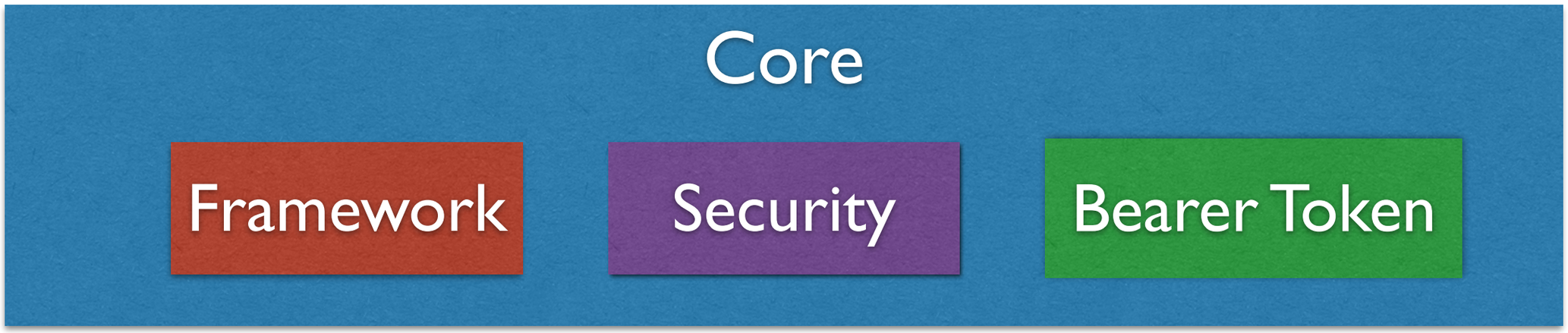
OAuth 2 Spec Overview

Framework

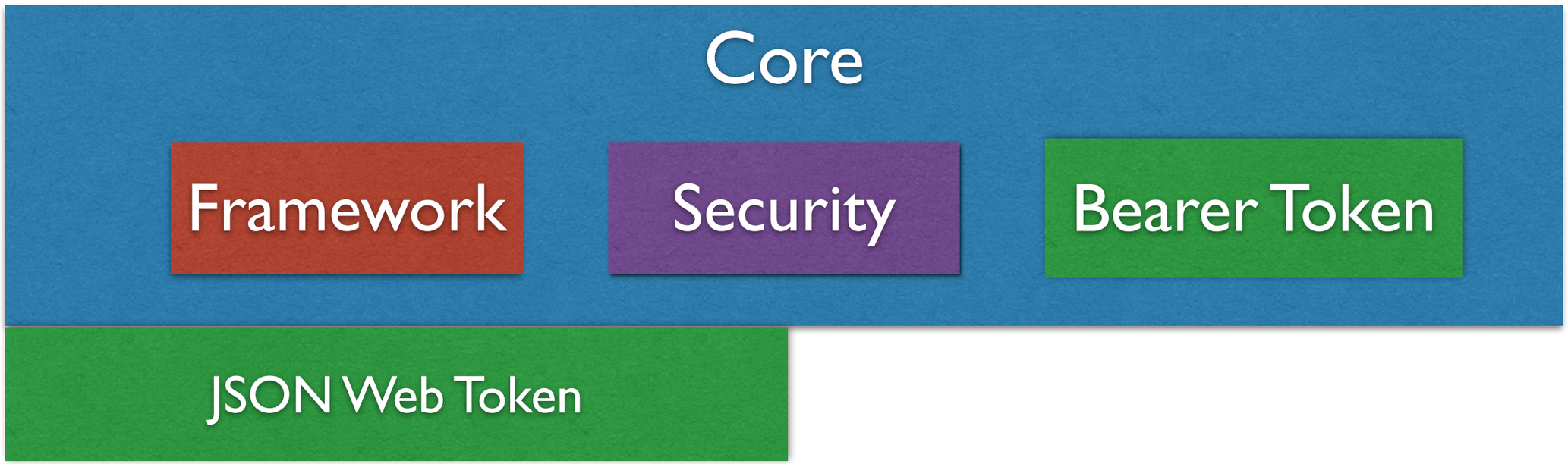
Security

Bearer Token

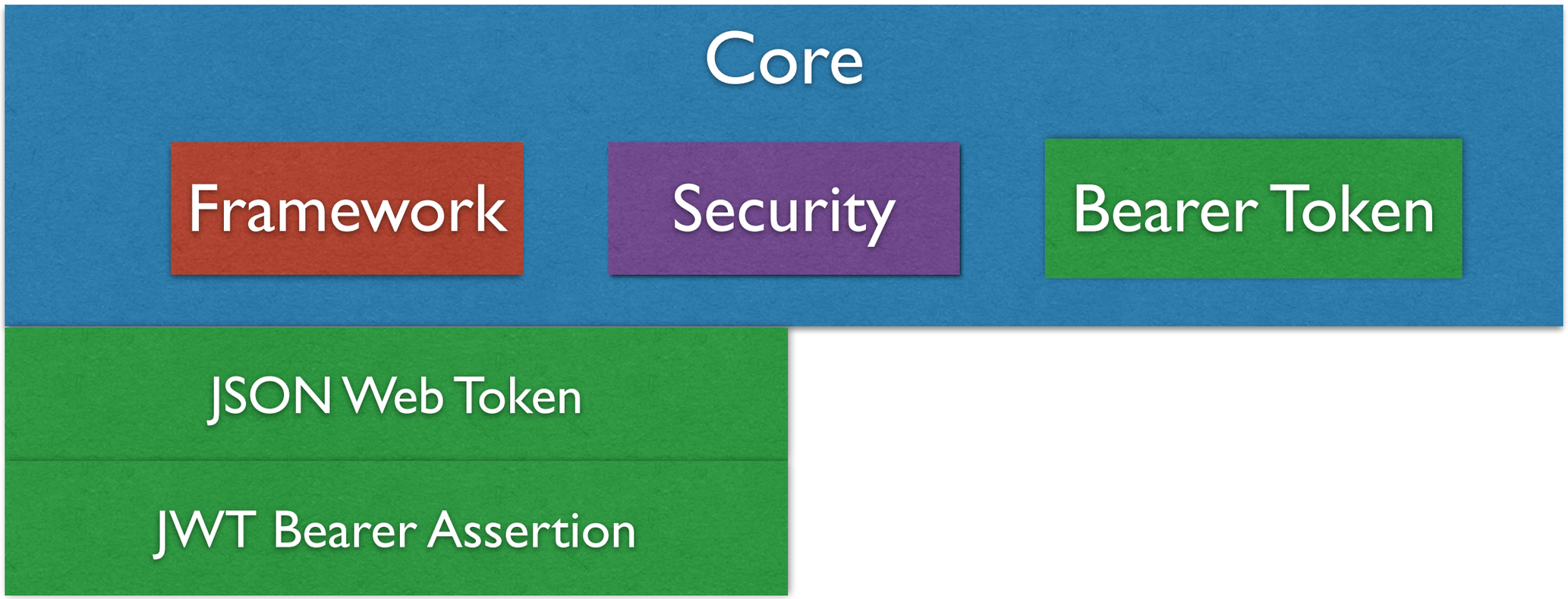
OAuth 2 Spec Overview



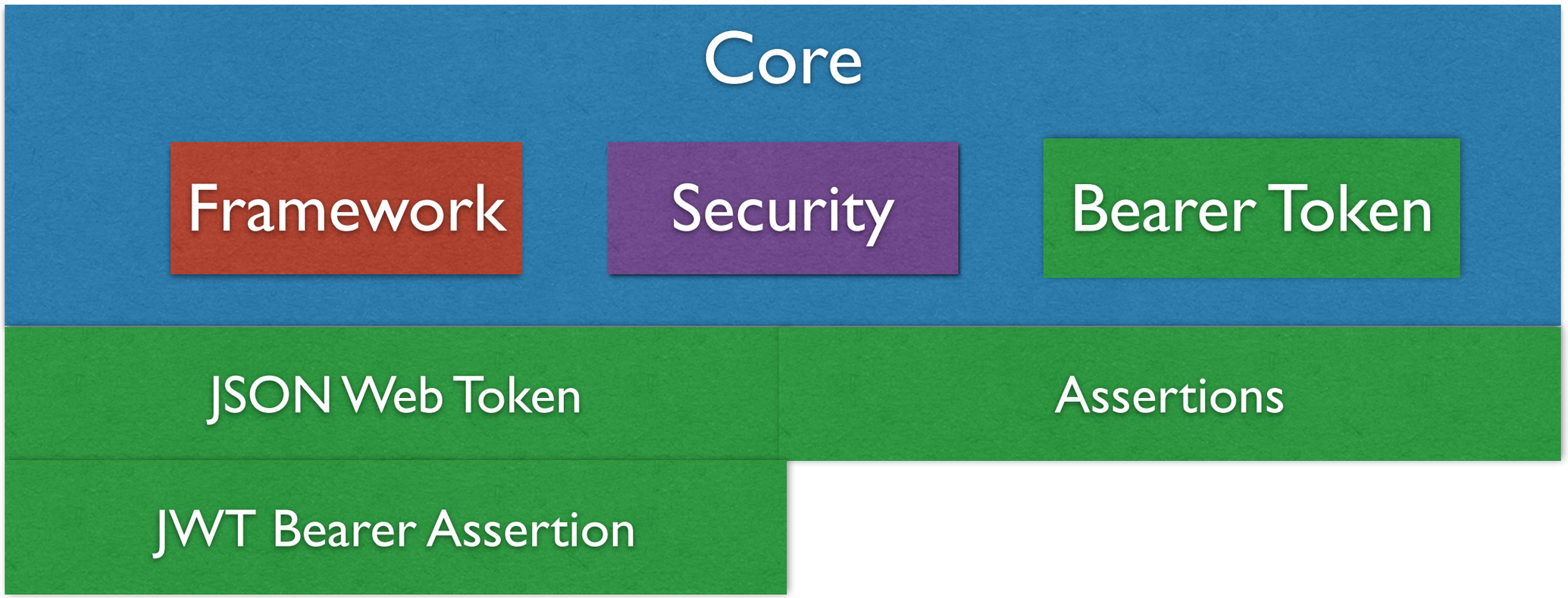
OAuth 2 Spec Overview



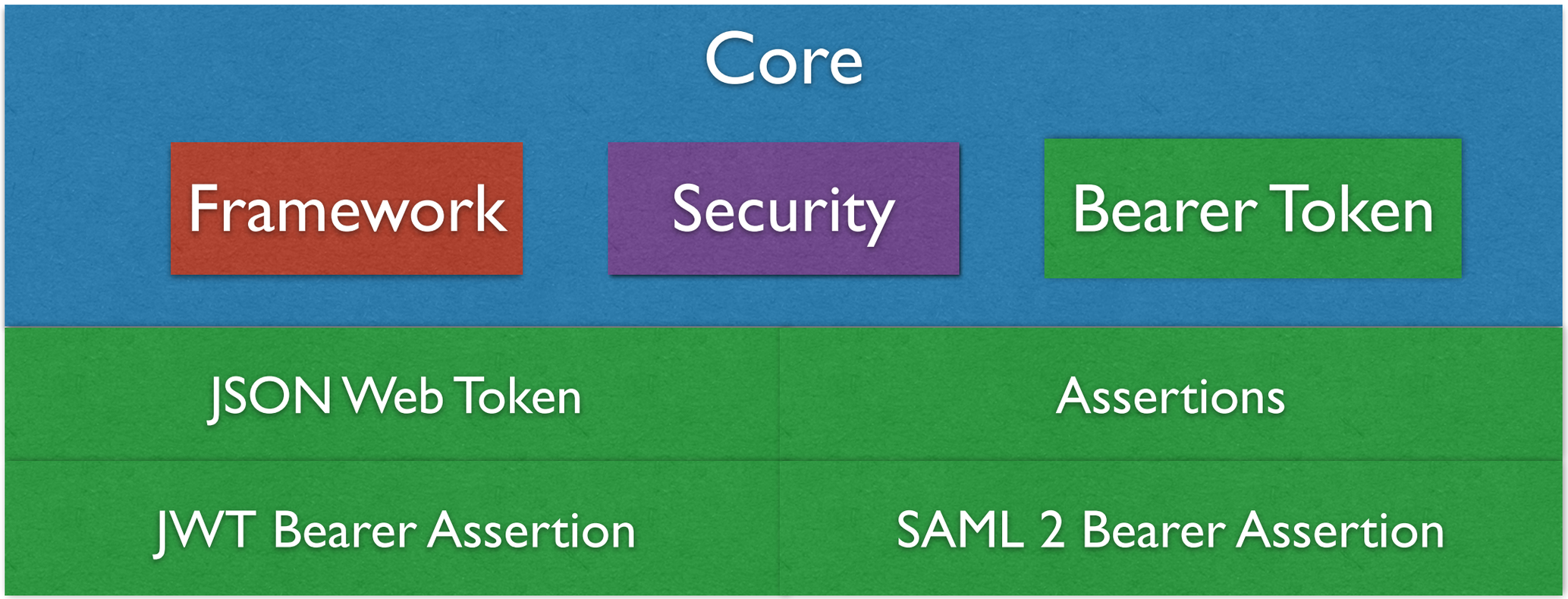
OAuth 2 Spec Overview



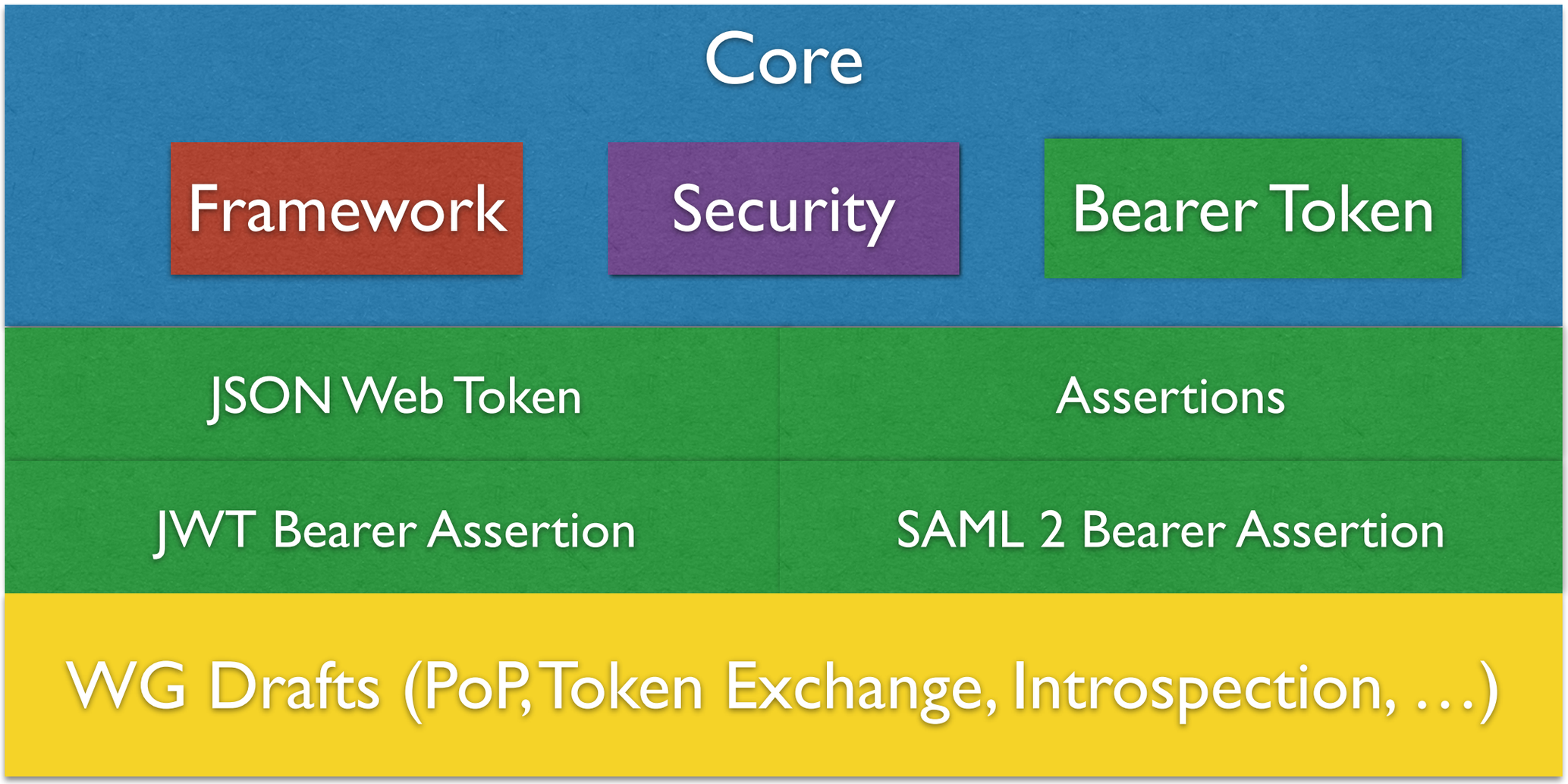
OAuth 2 Spec Overview



OAuth 2 Spec Overview



OAuth 2 Spec Overview



Short Comparison

OAuth 1	OAuth 2

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	› Many extensions

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	› Many extensions
› Additional crypto to TLS	

Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	› Many extensions
› Additional crypto to TLS	› TLS only (Core)

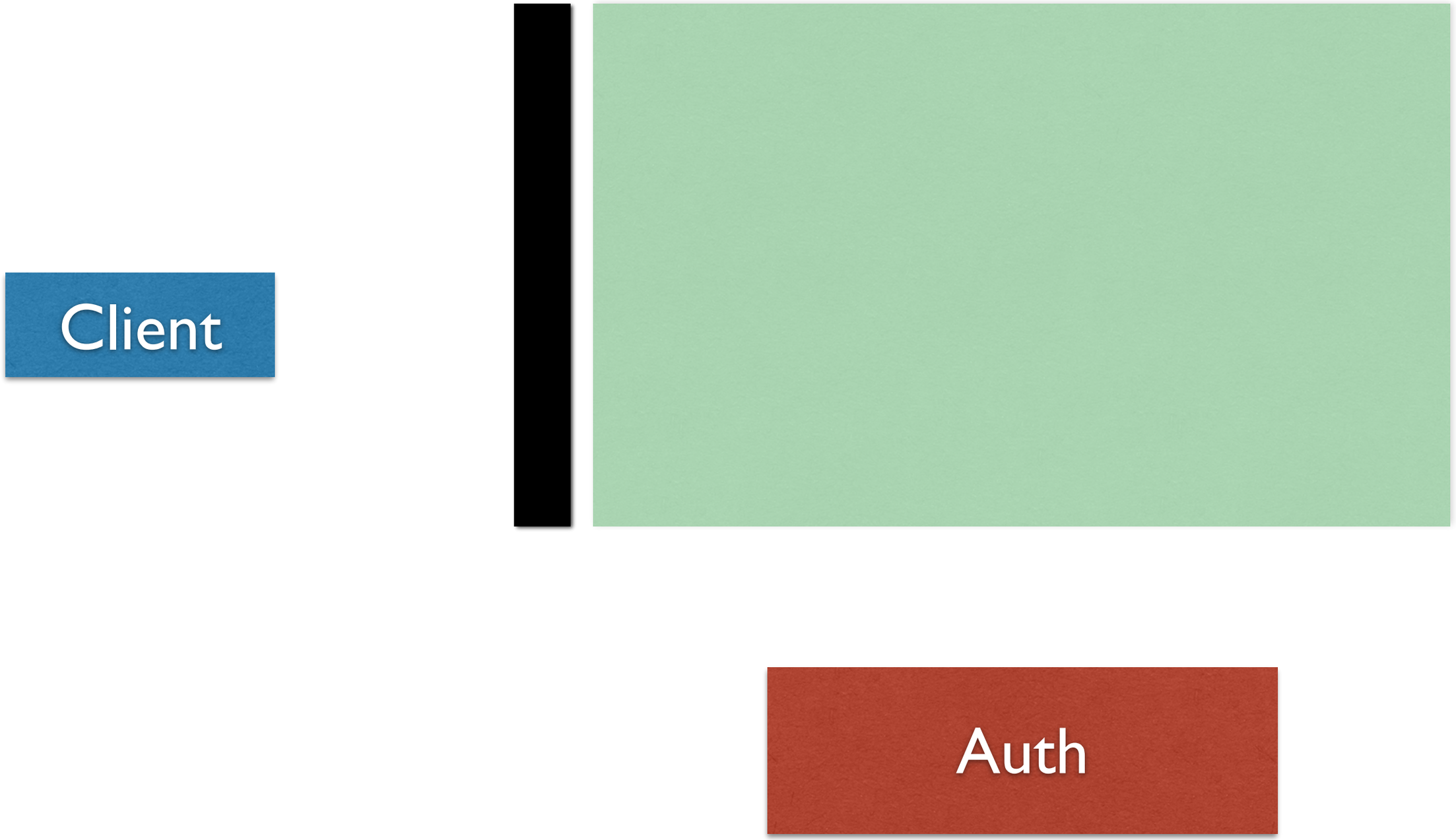
Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	› Many extensions
› Additional crypto to TLS	› TLS only (Core)
› Production ready libs	

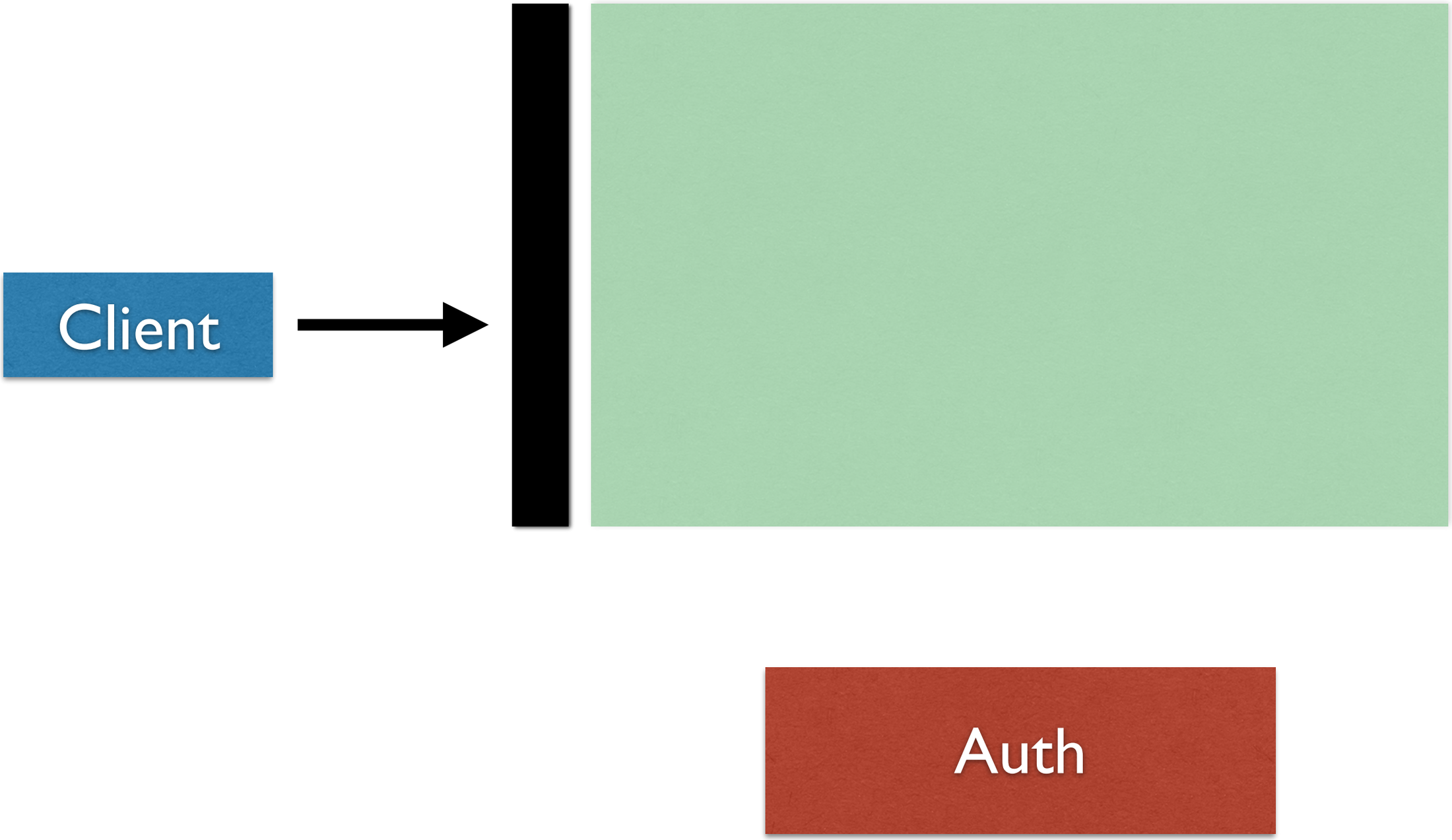
Short Comparison

OAuth 1	OAuth 2
› Authorization protocol	› Delegation framework
› Spec finalized	› Many extensions
› Additional crypto to TLS	› TLS only (Core)
› Production ready libs	› Incompatible implementations

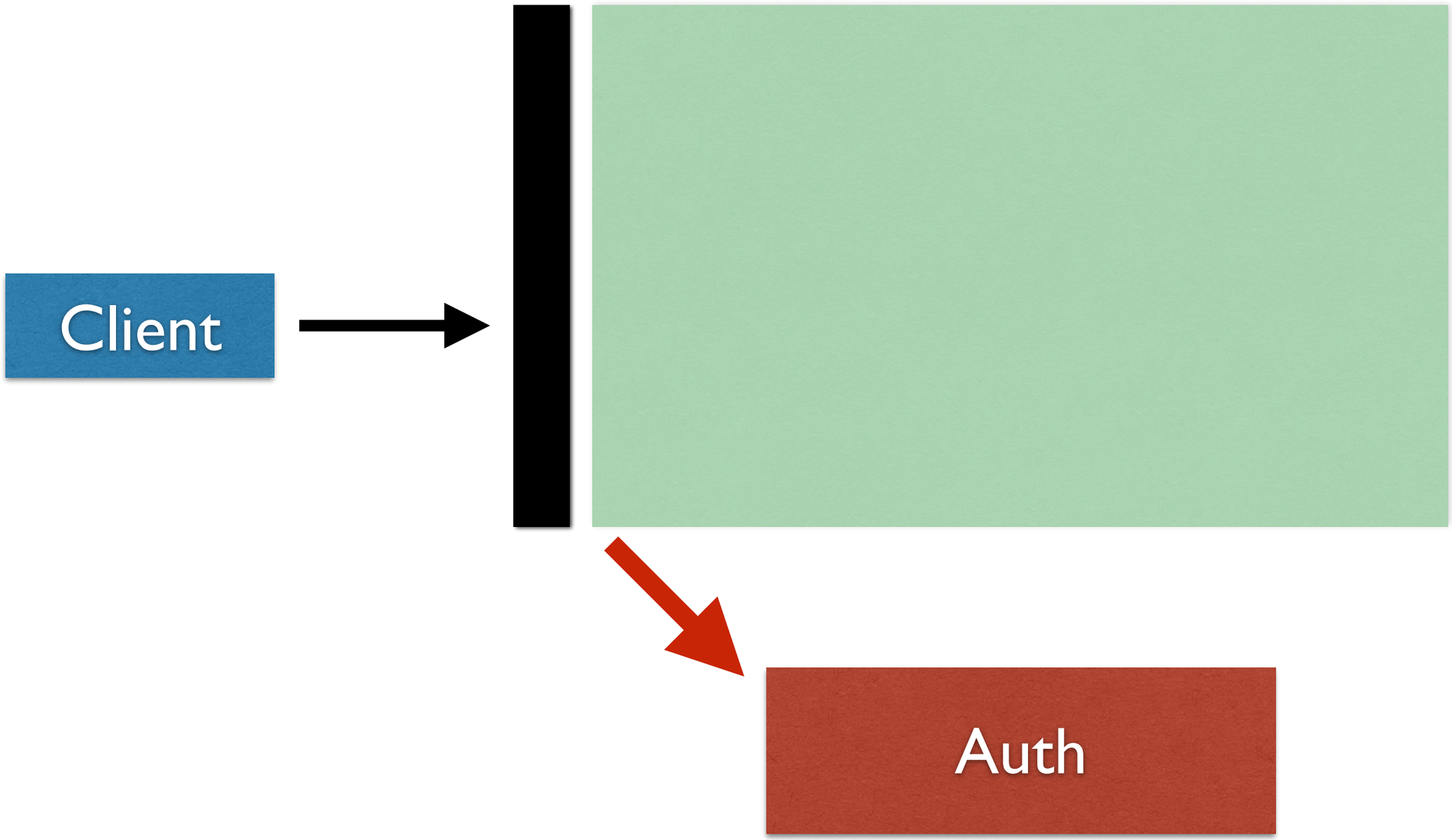
Example: Traditional System



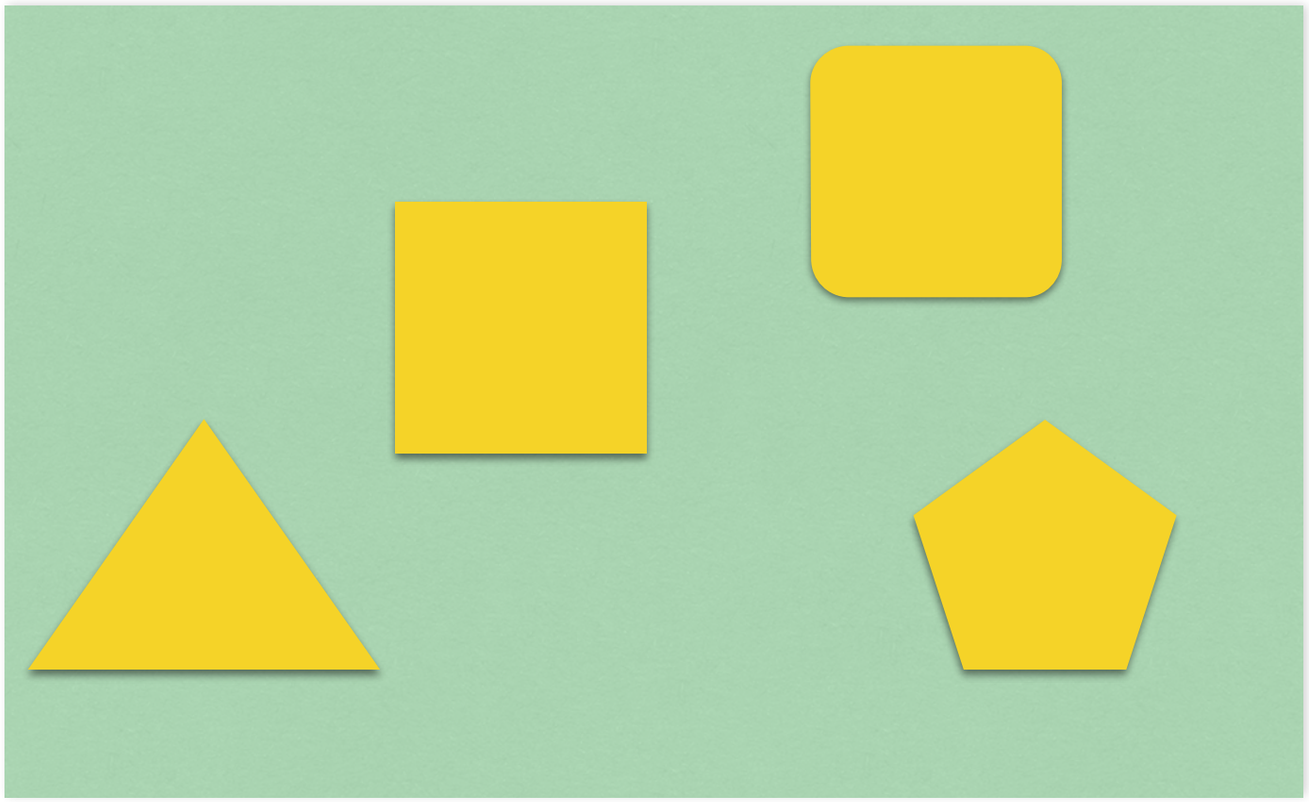
Example: Traditional System



Example: Traditional System



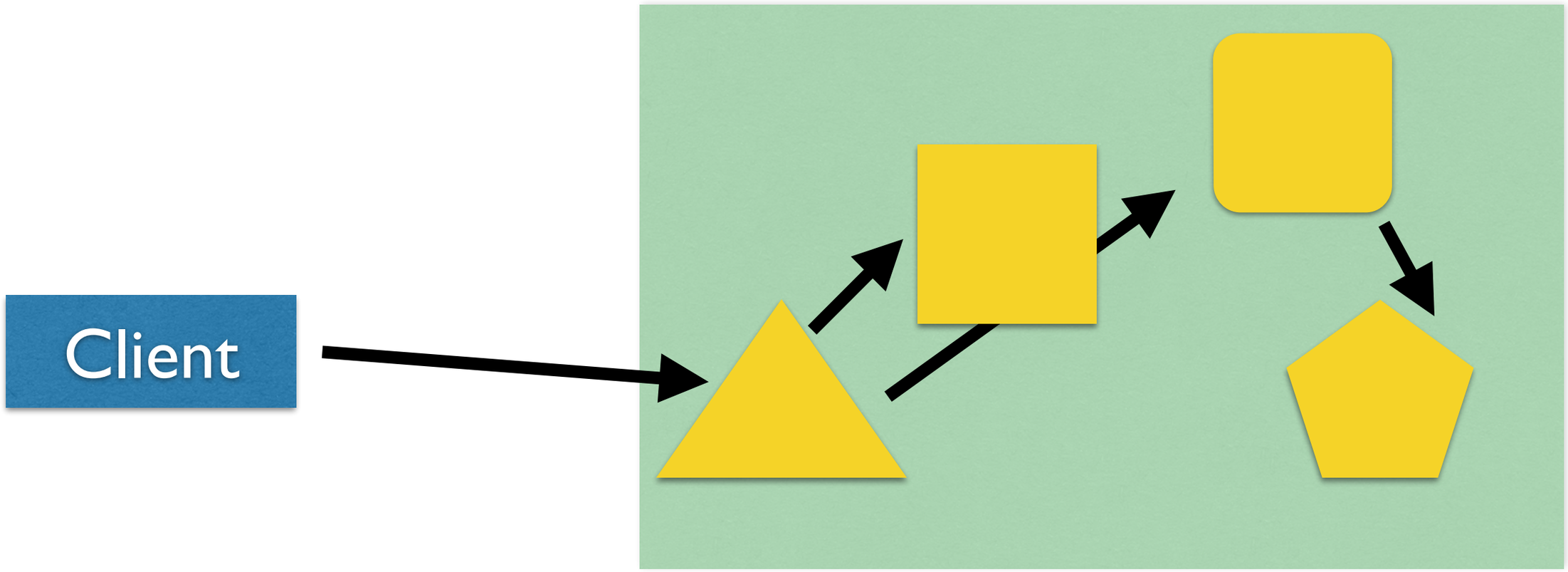
Example: Microservices



Client

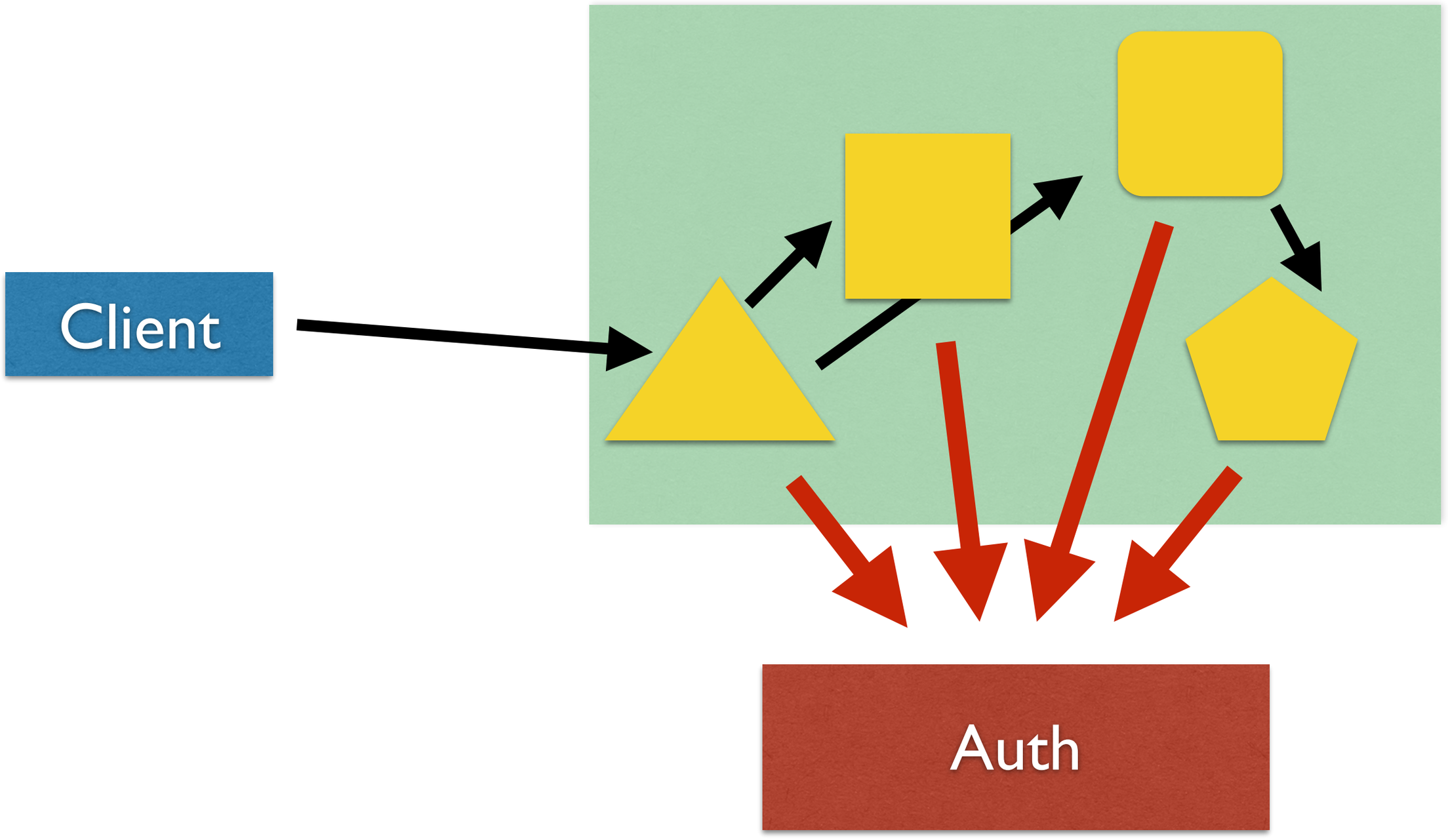
Auth

Example: Microservices

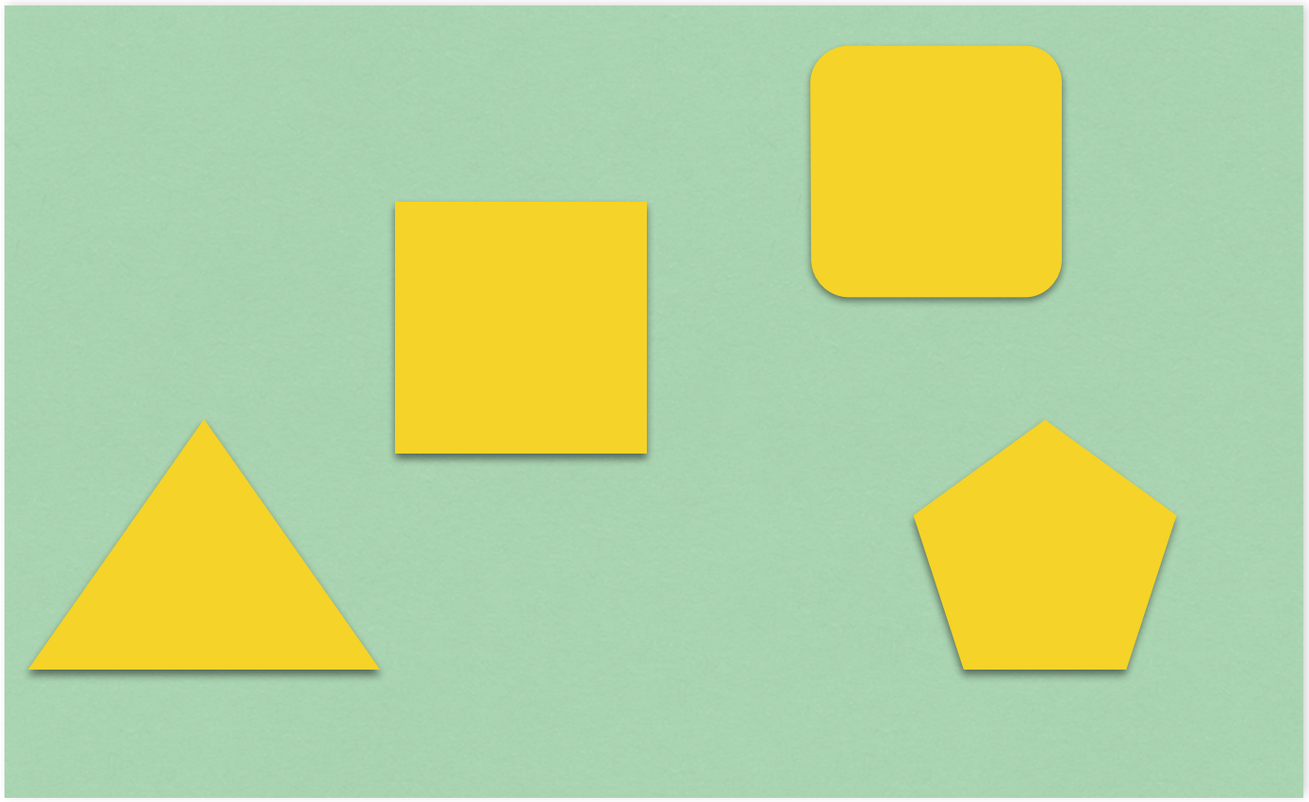


Auth

Example: Microservices



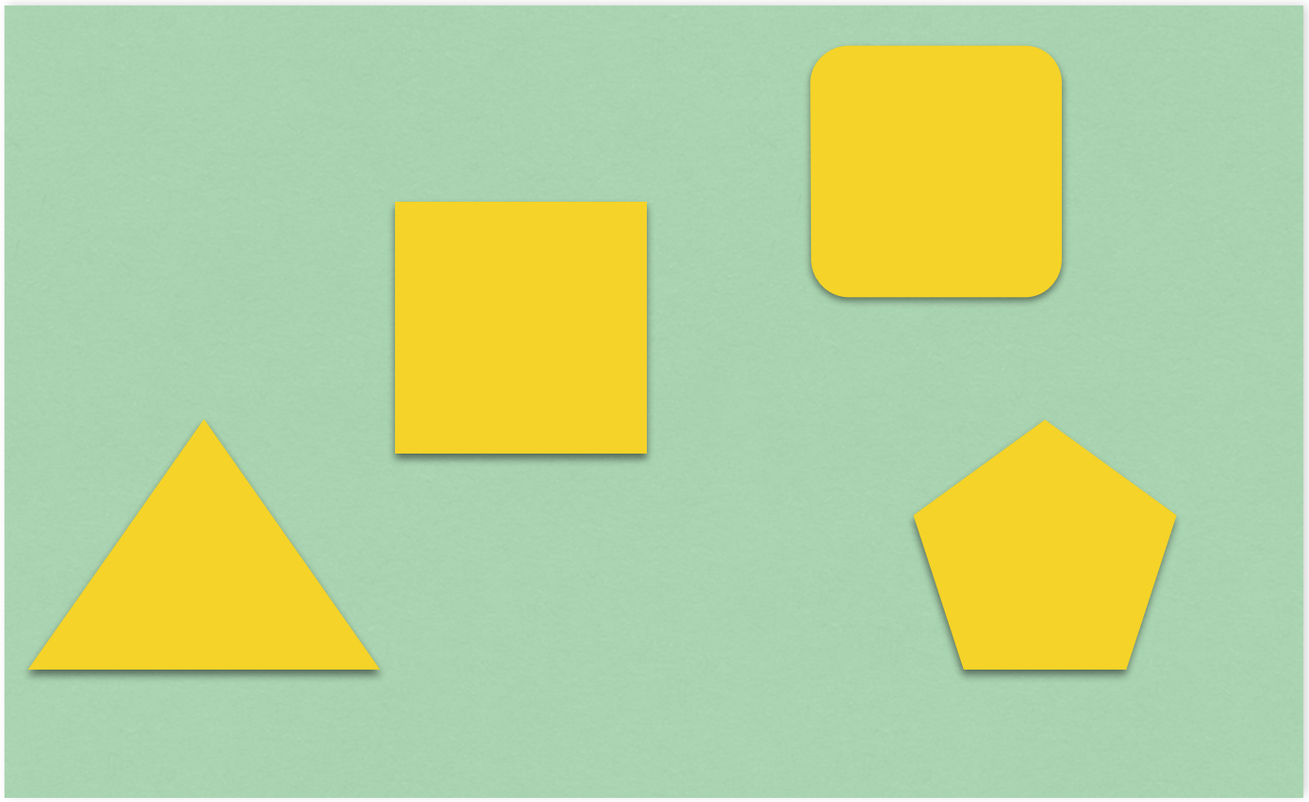
Example: Microservices



Client

Auth

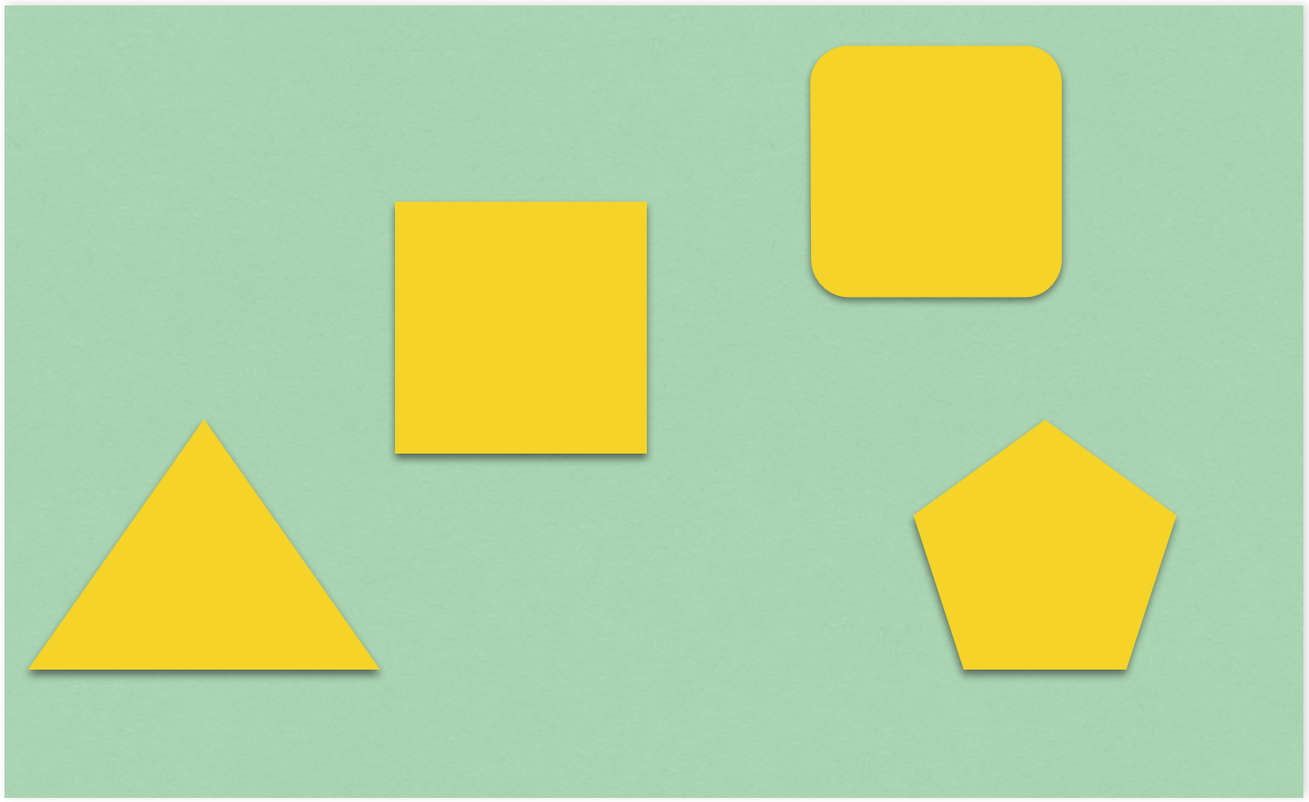
Example: Microservices



Client

Auth

Example: Microservices

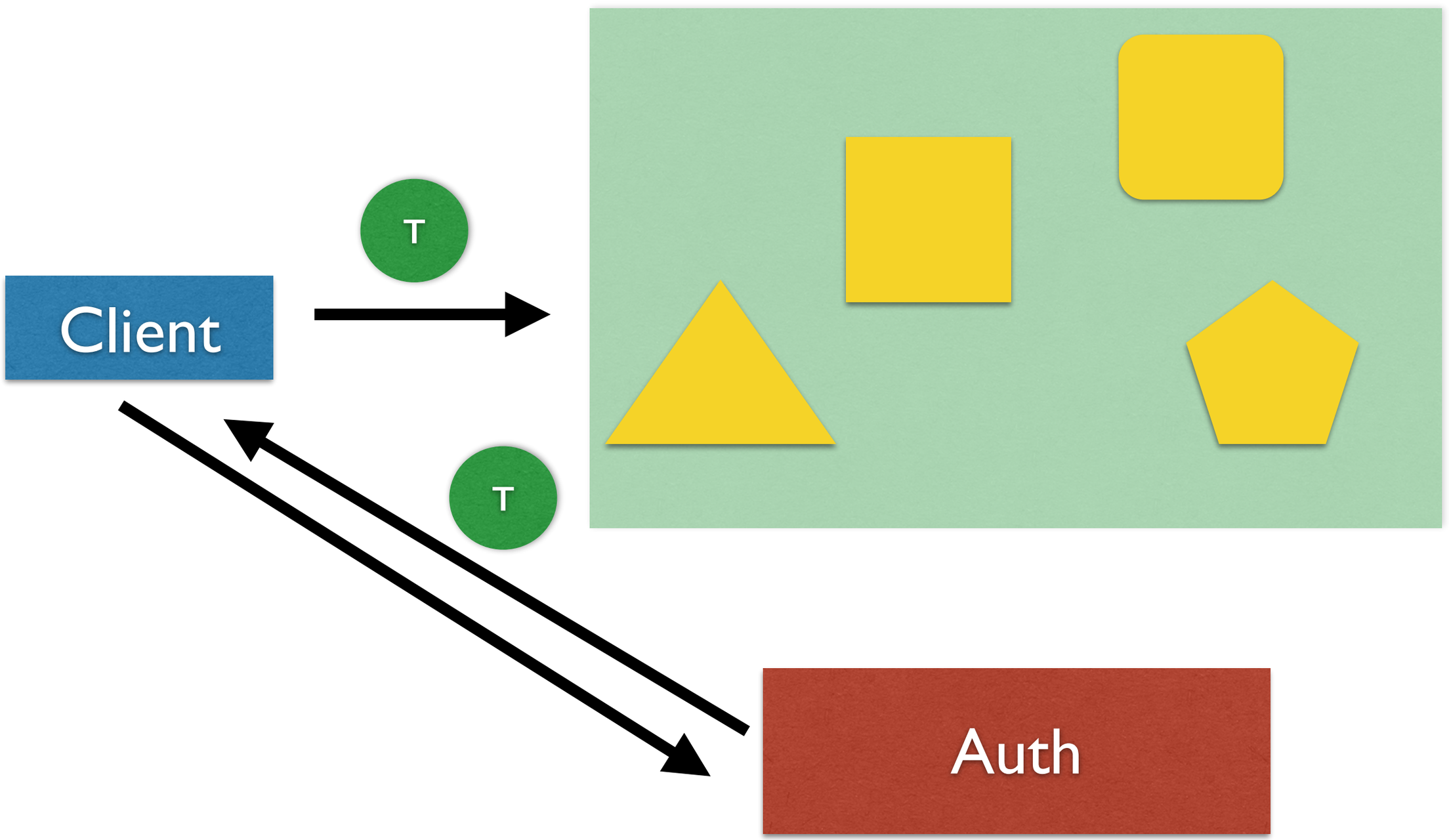


Client

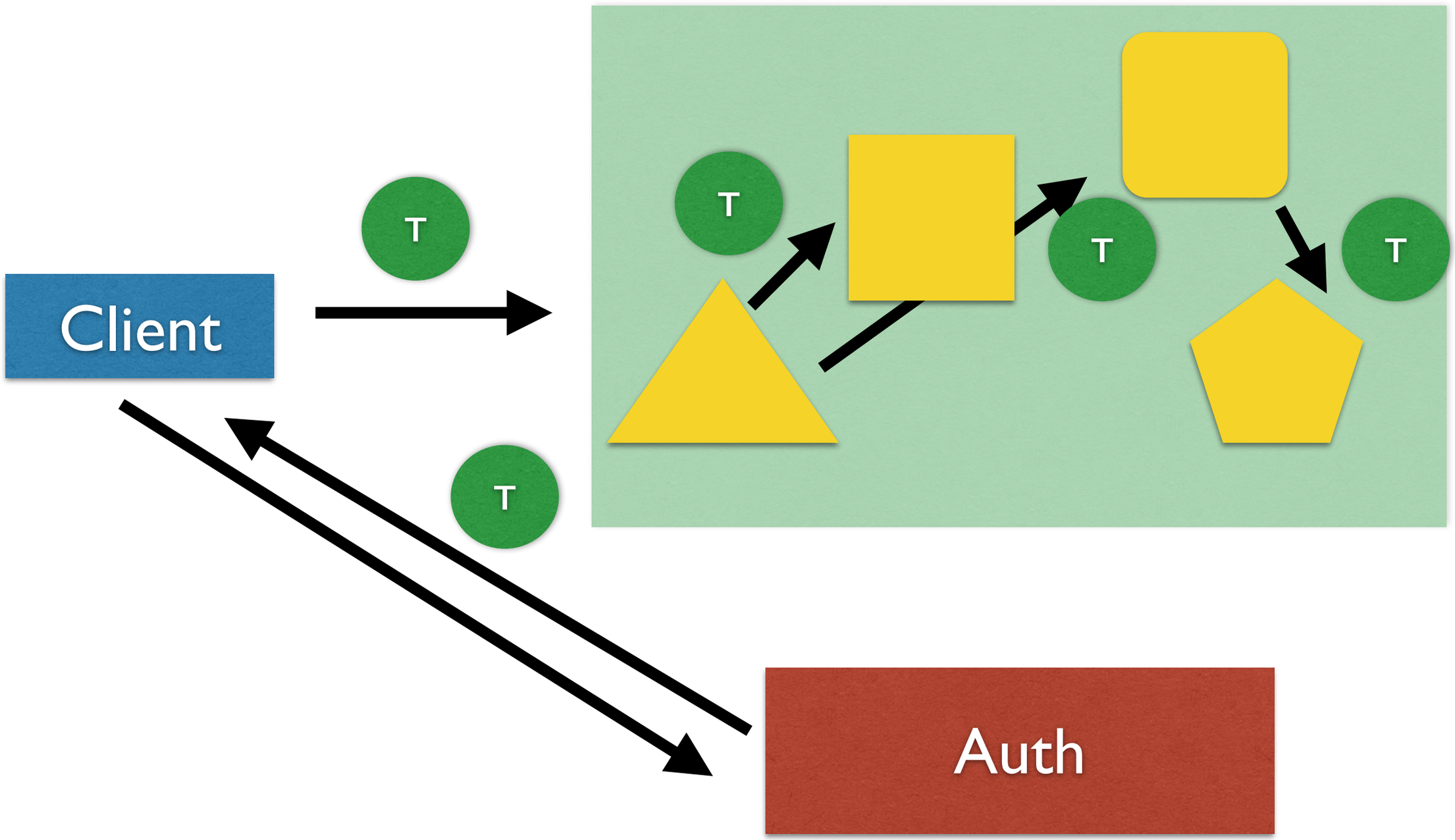
T

Auth

Example: Microservices



Example: Microservices



OAuth 2 for Single-Sign-On?

OAuth 2 for Single-Sign-On?

- Token = “Authorization to access Resource”
  Resource could be = Login to Service

OAuth 2 for Single-Sign-On?

- Token = “Authorization to access Resource”
  Resource could be = Login to Service
- Missing:

When?

OAuth 2 for Single-Sign-On?

- Token = “Authorization to access Resource”
  Resource could be = Login to Service
- Missing:

Who?

When?

OAuth 2 for Single-Sign-On?

- Token = “Authorization to access Resource”
  Resource could be = Login to Service
- Missing:

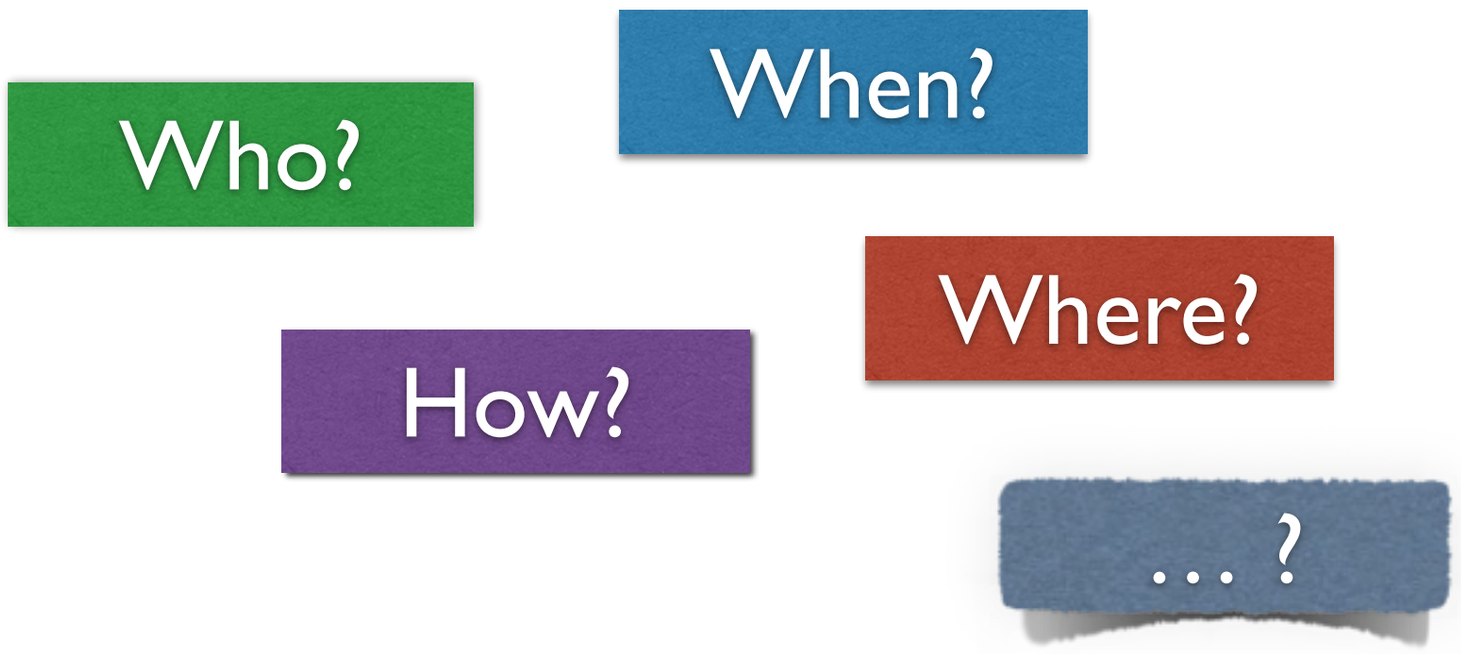
Who?

When?

Where?

OAuth 2 for Single-Sign-On?

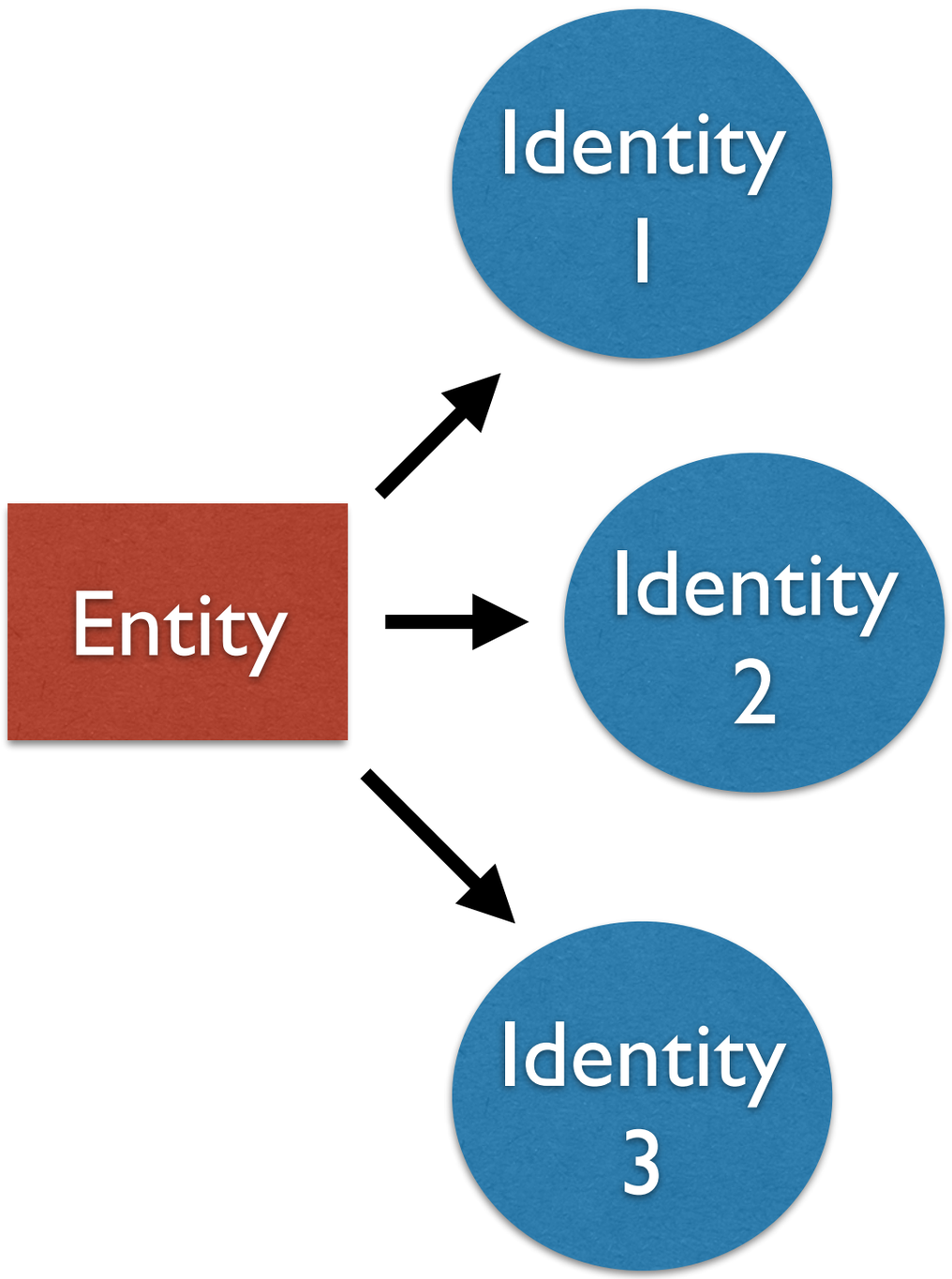
- Token = “Authorization to access Resource”
  Resource could be = Login to Service
- Missing:



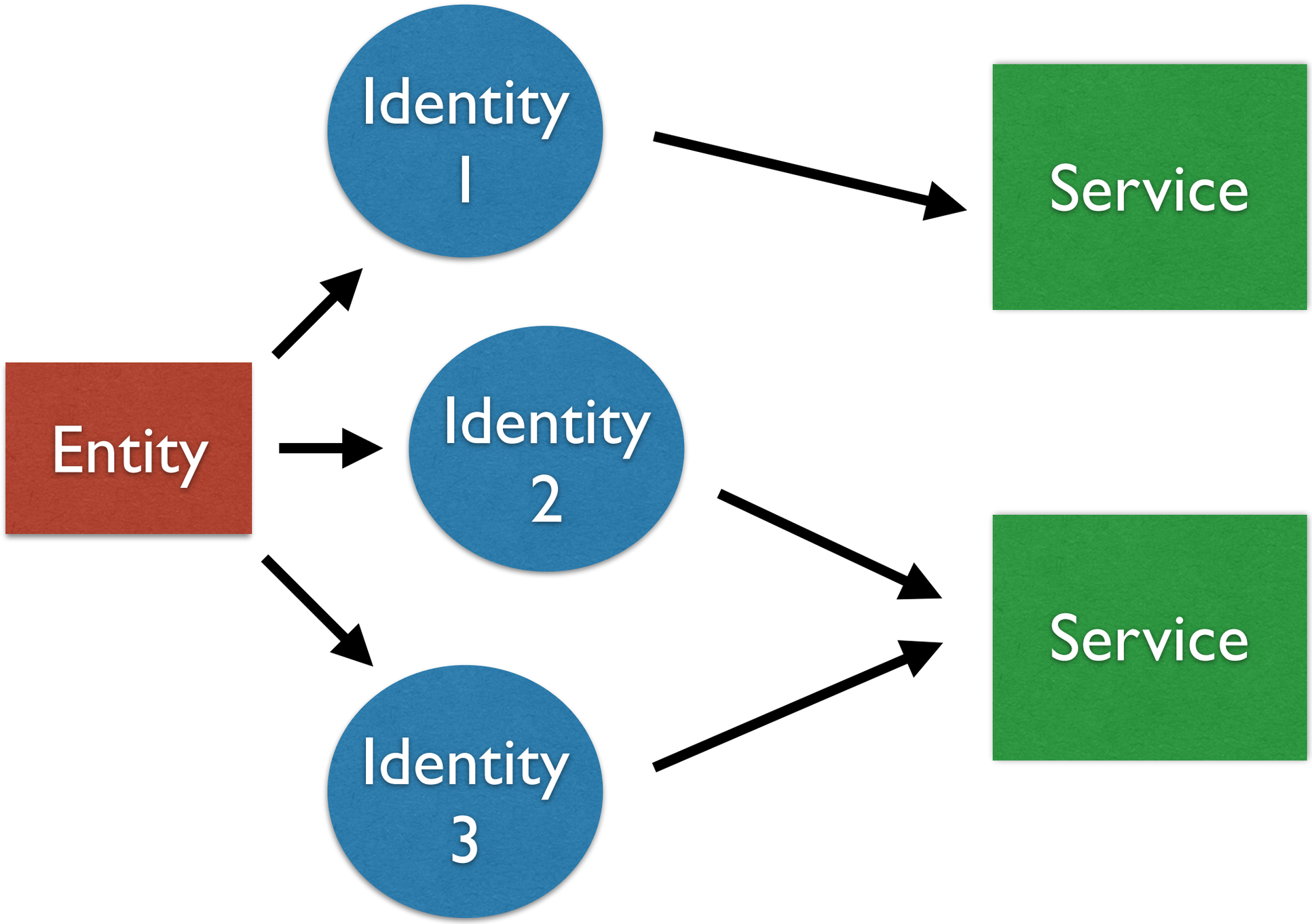
Identities

Entity

Identities



Identities



OpenID Connect

OpenID Connect

OpenID Connect

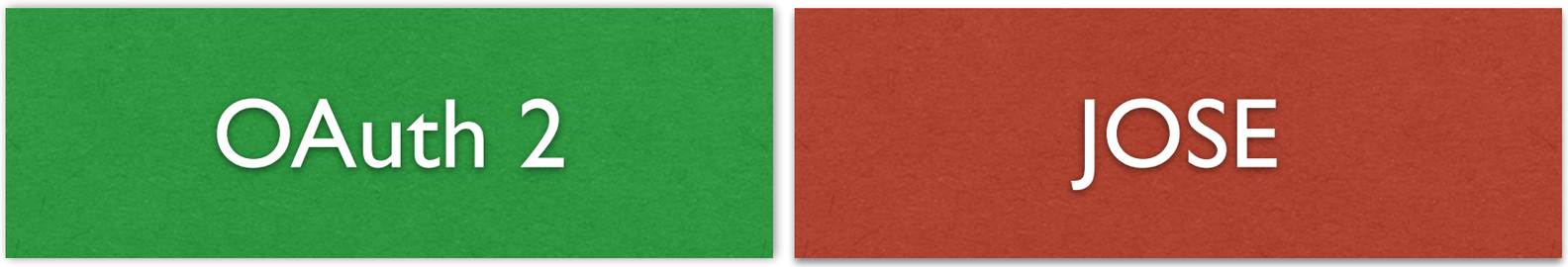
OpenID Connect

OAuth 2

OpenID Connect

OpenID Connect

Identity Layer



JWT Example

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

JWT Example

JWT Header

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
    "iss": "http://server.example.com",
    "sub": "248289761001",
    "aud": "s6BhdRkqt3",
    "exp": 1311281970,
    "iat": 1311280970,
    "nonce": "n-0S6_WzA2Mj",
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "gender": "female",
    "birthdate": "0000-10-31",
    "email": “janedoe@example.com”,
    "email_verified": {"essential": true}
}
```

JWT Example

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

JWT Payload

JWT Example

Signature

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

JWT Example Encoded

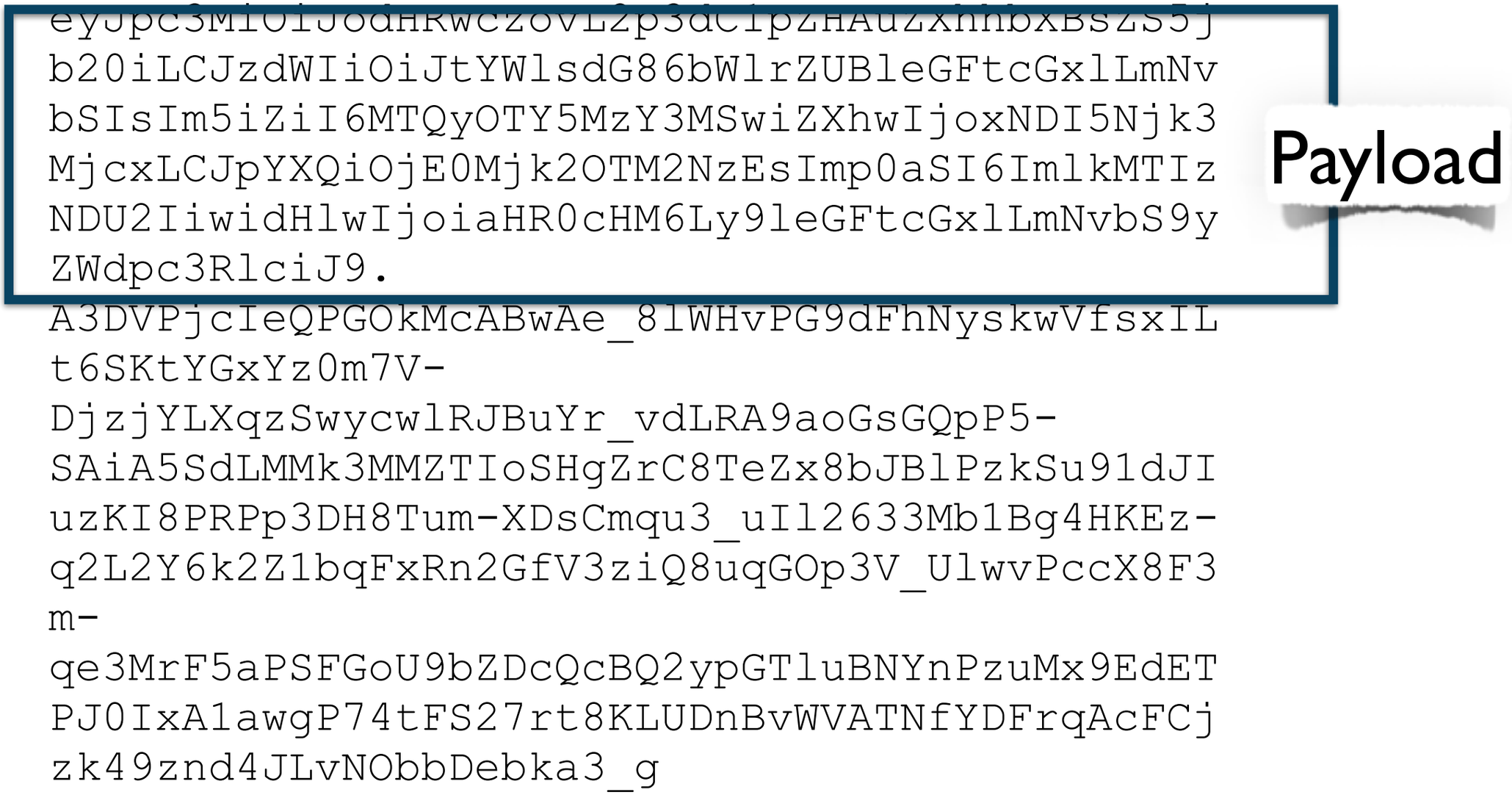

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv
bSIsIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxLCJpYXQiOjE0Mjk2OTM2NzEsImp0aSI6ImlkMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYLXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTIoSHgZrC8TeZx8bJBlPzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uIl2633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g
```

JWT Example Encoded

Header

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv
bSIsIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxLCJpYXQiOjE0Mjk2OTM2NzEsImp0aSI6ImlkMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYLXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTIoSHgZrC8TeZx8bJBlPzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uIl2633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g
```

JWT Example Encoded



Payload

JWT Example Encoded


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv
bSIsIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxLCJpYXQiOjE0Mjk2OTM2NzEsImp0aSI6ImlkMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYLXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTIoSHgZrC8TeZx8bJBlPzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uIl2633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g
```


Signature

JWT Example Encoded


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv
bSIsIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxLCJpYXQiOjE0Mjk2OTM2NzEsImp0aSI6ImlkMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYLXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTIoSHgZrC8TeZx8bJBlPzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uIl2633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g
```

ID Token Example

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

ID Token Example

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj"
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

Claim

ID Token Example

```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

mandatory Claims

ID Token Example

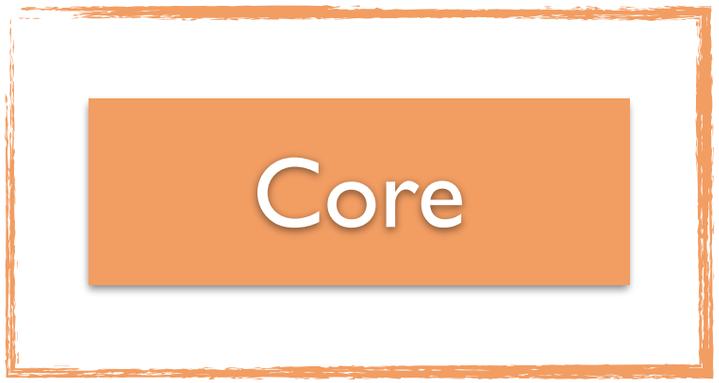
```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

scope E-Mail

ID Token Example

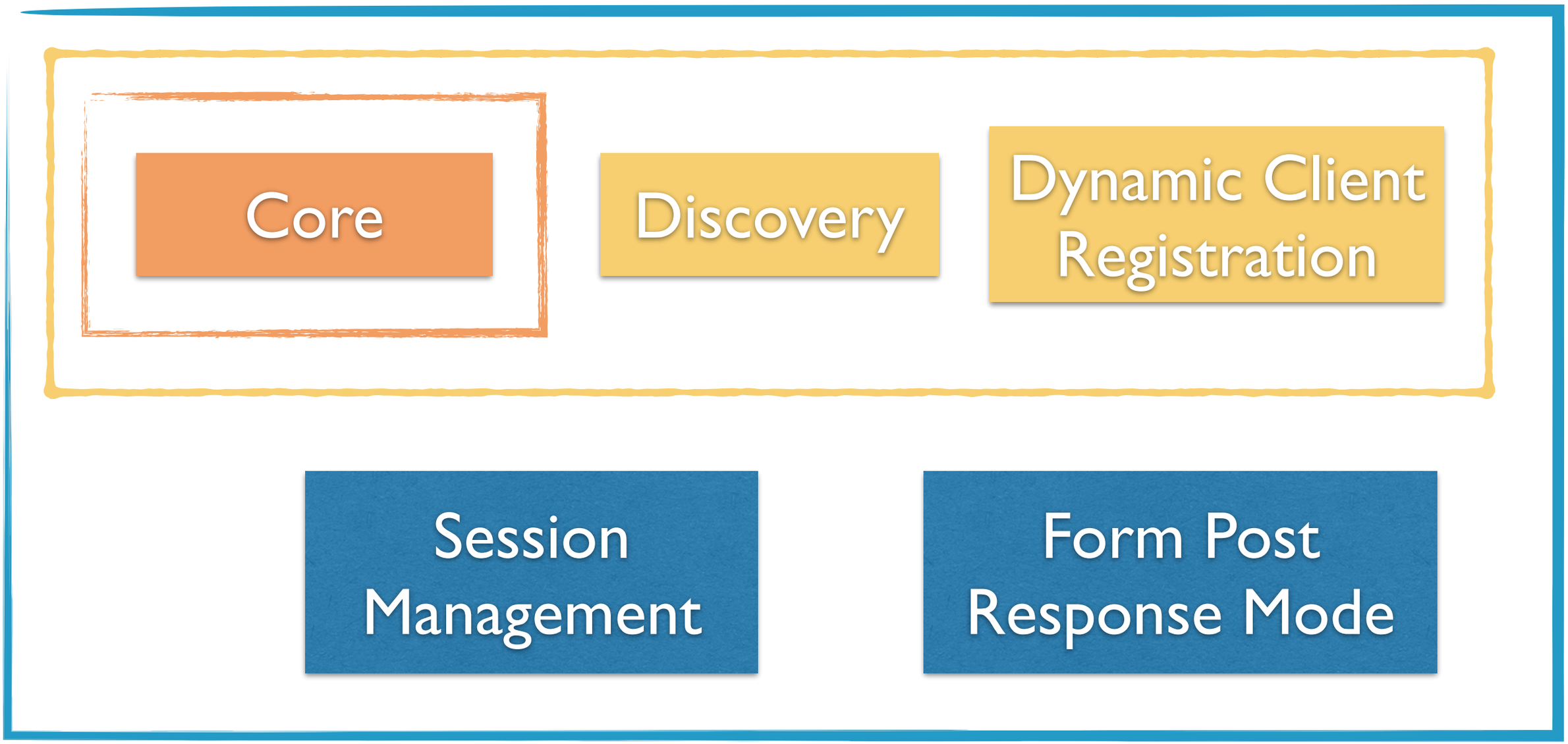
```
{
“typ”:”JWT”,
“alg”:”HS256”
}{
   "iss": "http://server.example.com",
   "sub": "248289761001",
   "aud": "s6BhdRkqt3",
   "exp": 1311281970,
   "iat": 1311280970,
   "nonce": "n-0S6_WzA2Mj",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "gender": "female",
   "birthdate": "0000-10-31",
   "email": “janedoe@example.com”,
   "email_verified": {"essential": true}
}
```

OpenID Connect Modules



OpenID Connect Modules

OpenID Connect Modules



Enterprise Use Cases

Enterprise Use Cases

- Securing internal APIs

Enterprise Use Cases

- Securing internal APIs
- Single-Sign-On

Enterprise Use Cases

- Securing internal APIs
- Single-Sign-On
- Active Directory

Enterprise Use Cases

- Securing internal APIs
- Single-Sign-On
- Active Directory
- Mobile Clients

Enterprise Use Cases

- Securing internal APIs
- Single-Sign-On
- Active Directory
- Mobile Clients
- White Label Software

Should I use it?

- OpenID Connect Extensions still partly Working Drafts

Should I use it?

- OpenID Connect Extensions still partly Working Drafts
- JOSE still Working Draft

Should I use it?

- OpenID Connect Extensions still partly Working Drafts
- JOSE still Working Draft
- OAuth Extensions: PoP, etc.

Should I use it?

- OpenID Connect Extensions still partly Working Drafts
- JOSE still Working Draft
- OAuth Extensions: PoP, etc.
- Implementation Complexity

Thank you!

Questions?

Comments?

Simon Kölsch @simkoelsch
simon.koelsch@innoq.com

Christoph Iserlohn
christoph.iserlohn@innoq.com

www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Robert-Bosch-Straße 7
64293 Darmstadt
Germany
Phone: +49 2173 3366-0

Radlkoferstraße 2
D-81373 München
Germany
Telefon +49 (0) 89 741185-270

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116