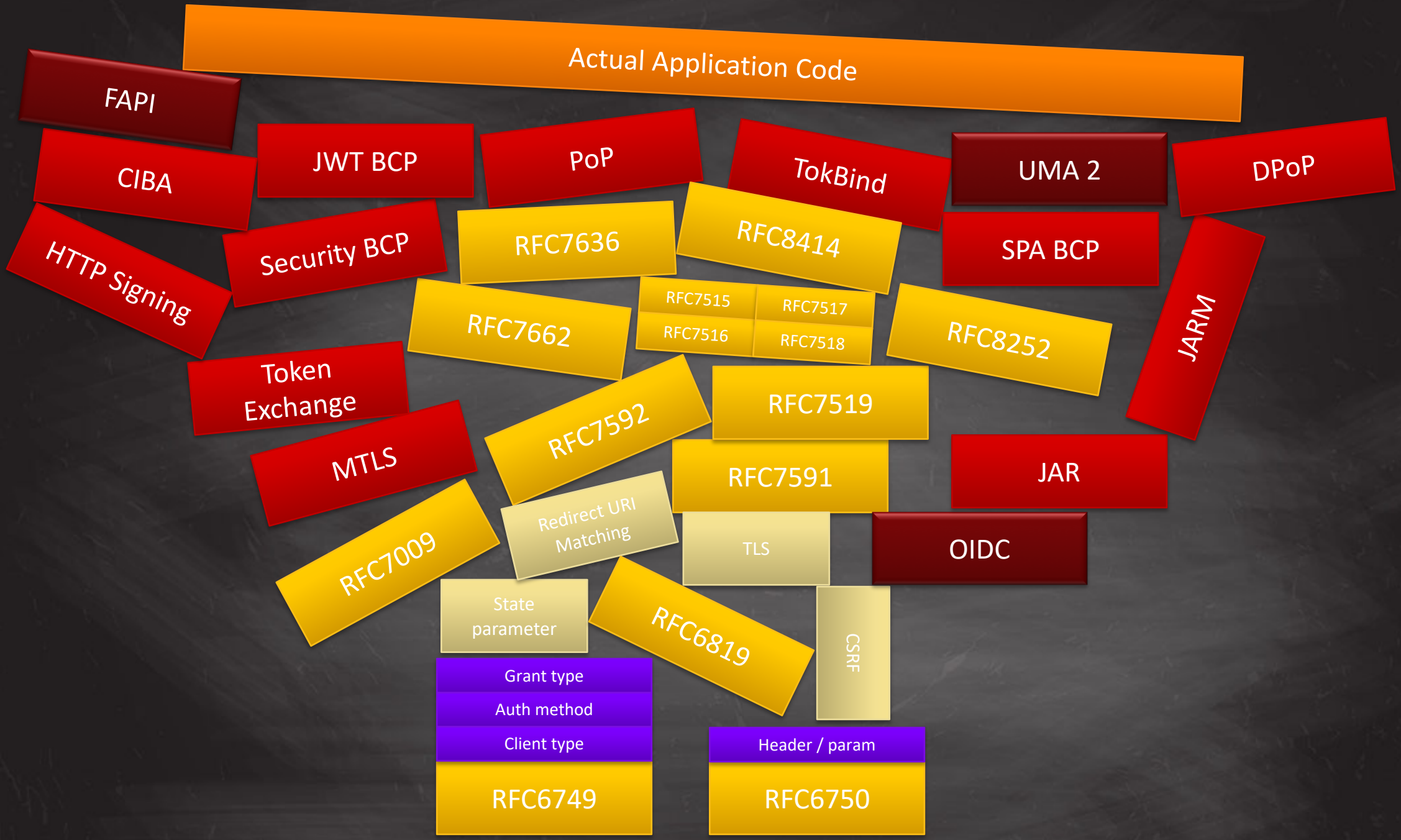
@justin _ _ richer

https://bspk.io/

RFC6749

RFC6750


Actual Application Code
FAPI
CIBA
JWT BCP
PoP
TokBind
UMA 2
DPoP
HTTP Signing
Security BCP
RFC7636
RFC8414
SPA BCP
JARM
RFC7662
RFC7515
RFC7517
RFC7516
RFC7518
RFC8252
Token Exchange
RFC7592
RFC7519
MTLS
RFC7591
JAR
RFC7009
Redirect URI Matching
TLS
OIDC
State parameter
RFC6819
CSRF
Grant type
Auth method
Client type
RFC6749
Header / param
RFC6750

OAuth is reaching its edges

@justin _ _ richer

https://bspk.io/

What if we had a new base?

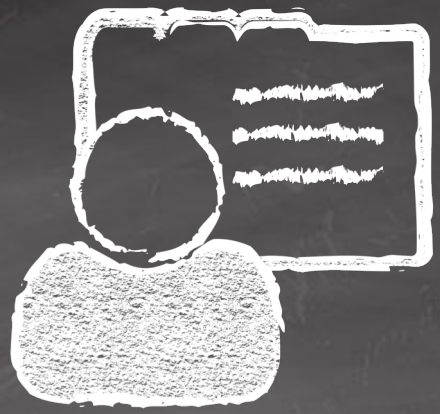

123
XYZ


123
OAUTH.XYZ

This is not an extension to OAuth 2

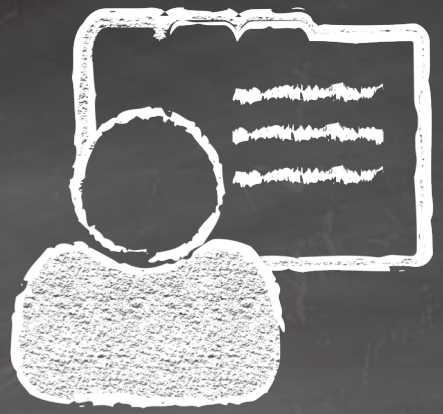
The Front Channel

- User is present
- Brower is flexible



The Front Channel

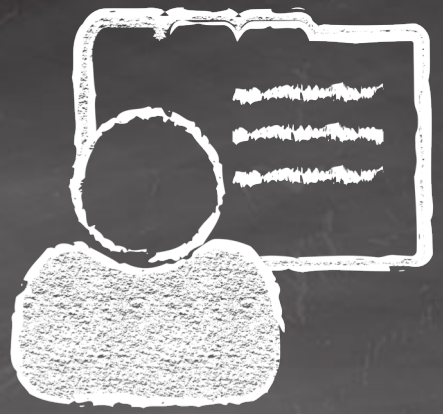
- User is present
- Brower is flexible



- Information leakage
- Tampering
- Injection
- URL size limitations
- HTTP Referrer headers
- HTTP server logs

The Front Channel

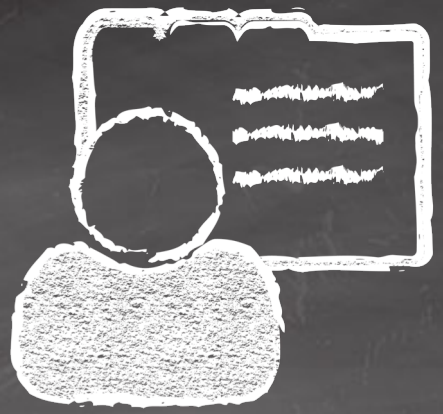
- User authentication
- User interaction
- Client identifier
- Requested scope
- Application state
- etc…



- Authorization code
- Access tokens
- Identity assertions
- Application state
- etc.

Trying to protect the front channel

- OIDC
- JAR
- JARM
- PKCE
- Token Binding
- Security BCPs



Proposal:

Avoid the Front Channel until we need it

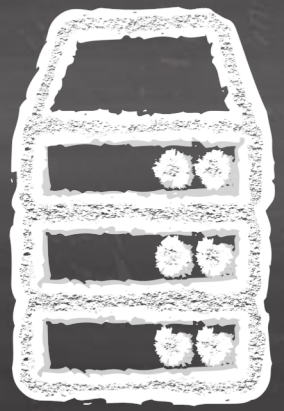
Transactions!

OAuth has always been transactional

Transactions:

Registering Intent

The client starts at the AS



Start a Transaction

```
{
    "client": ...
    "interact": ...
    "user": ...
    "resources": [ ... ],
    "key": ...
}
```

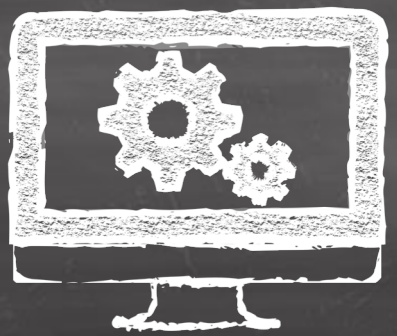
"What I want"

```
"resources": [{
    "actions": ["read", "write", "dolphin"],
    "locations": ["https://server.example.net/",
                  "https://resource.local/other"],
    "data": ["metadata"]
}]
```

"How to recognize me"

```
"key": {
    "type": "jwsd",
    "jwks": {
      "keys": [ {
        "kty": "RSA",
        "e": "AQAB",
        "kid": "xyz-1",
        "alg": "RS256",
        "n": "kOB5rR4Jv0GMeLaY6_It_..."
      } ]
    }
}
```

The client has to prove possession of all referenced keys



Sign the request body and present a header

```
Detached-JWS: eyJiNjQiOmZhbHNlLCJhbGciOiJSU...
```

"What I Am"

```
"client": {
    "name": "My Client Display Name",
    "uri": "https://example.net/client"
}
```

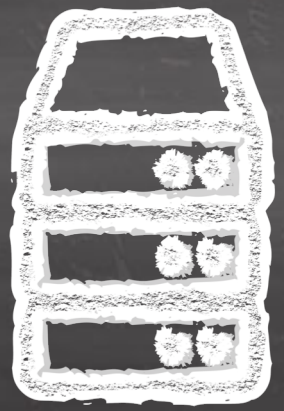
"What I know about the user"

```
"user": {
    "assertion": "eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoi..."
    "type": "oidc_id_token"
}
```

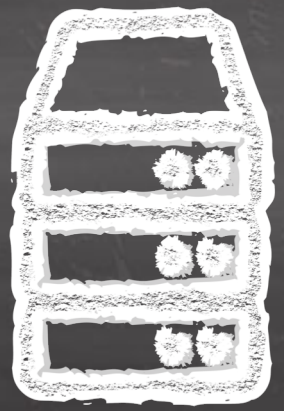
"How I can interact with the user"

```
"interact": {
    "type": "redirect",
    "callback": "https://client.example.net/return/123455",
    "state": "LKLTI25DK82FX4T4QFZC"
}
```

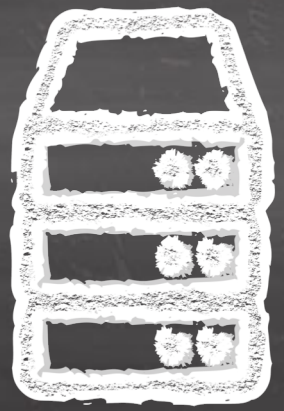
Process all aspects of the transaction request



Maybe we can already issue an access token



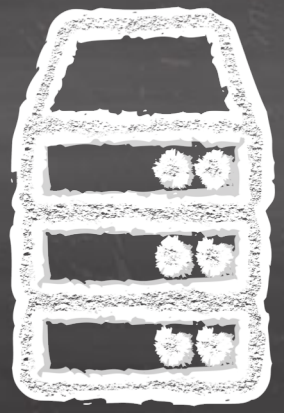
Or:
"I need to talk to the user"



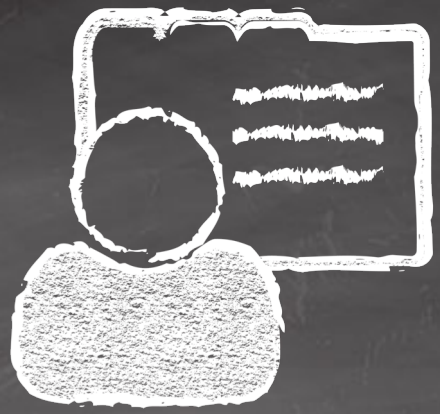
"Go fetch me the user"

```
{
    "interaction_url":

"https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ",

    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

Each step points to the next

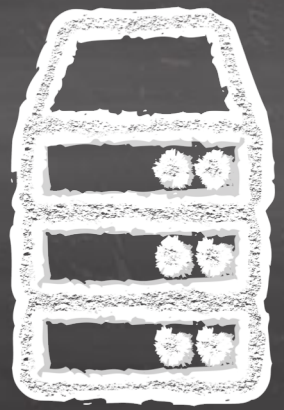


The Front Channel

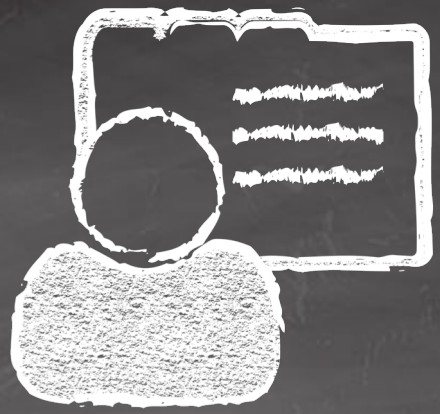


https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ

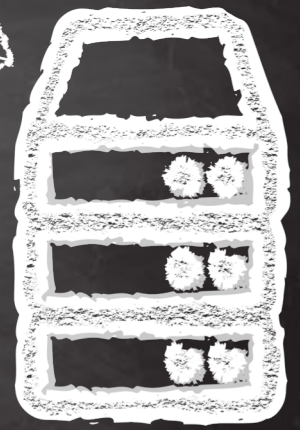
Look up the transaction based on the incoming interaction URL

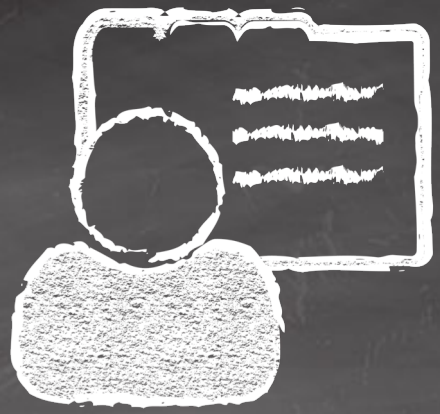


User interacts like you'd expect



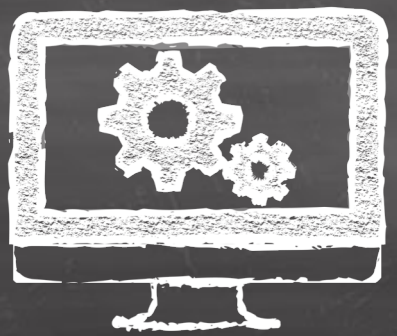
- Authenticate
- Authorize
- Consent
- Modify





https://client.example.net/return/123455
?state=LKLTI25DK82FX4T4QFZC&interact=4IFWWIKYBC2PQ6U56NL1

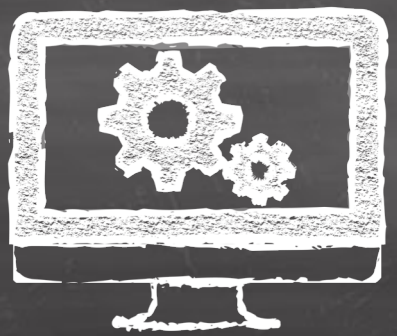
Validate the state value



Continue the Transaction

```
{
    "handle": "80UPRY5NM33OMUKMKSKU",
    "interact_handle": "4IFWWIKYBC2PQ6U56NL1"
}
```

The client STILL has to prove possession of all referenced keys



"Here's an access token"

```
{
    "access_token": {
        "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "type": "bearer"
    }
}
```

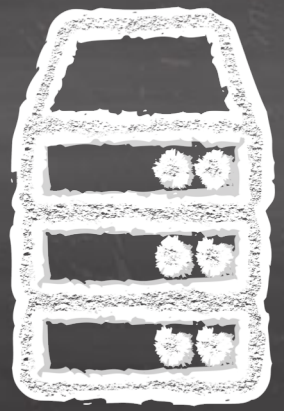
Handles:

Referencing previous state

"Use this, I'll remember you"

```
{
    "client_handle": {
        "value": "VBUEOIQA82PBY2ZDJW7Q", "type": "bearer"
    },
    "key_handle": {
        "value": "7C7C4AZ9KHRS6X63AJAO", "type": "bearer"
    }
    ...
}
```

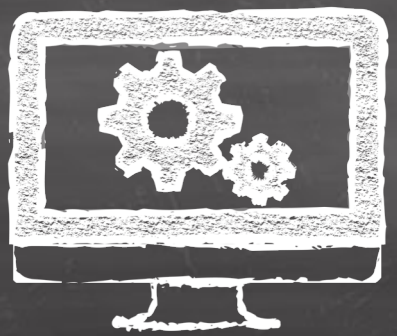
This can happen out of band



Starting a new transaction with handles

```
{
    "client": "VBUEOIQA82PBY2ZDJW7Q",
    "key": "7C7C4AZ9KHRS6X63AJAO"
}
```

The client STILL has to prove possession of all referenced keys



An access token and a transaction handle

```
{
    "access_token": {
        "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "type": "bearer”
    },
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

Refreshing a Token

```
{
    "handle": "80UPRY5NM33OMUKMKSKU"
}
```

Remembering or identifying the user

```
{
    "user_handle": {
        "value": "XUT2MFM1XBIKJKSDU8QM",
        "type": "bearer"
    }
}
```

Scopes, redux

```
"resources": [
    "read", "write", "dolphin"
]
```

Structured scopes

```
"resources": [
    "read", "write", "dolphin",
    {
        "actions": ["read", "write", "dolphin"],
        "locations": ["https://server.example.net/",
                      "https://resource.local/other"],
        "data": ["metadata"]
    }
]
```

What about other devices?

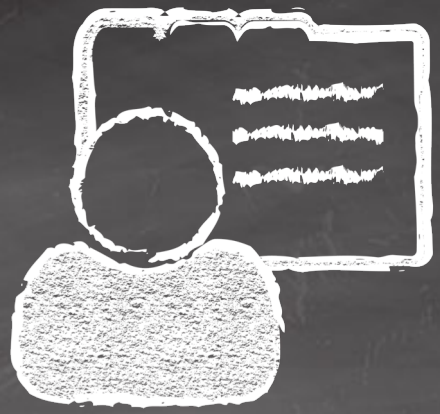
"How I Can Interact With The User"

```
"interact": {
    "type": "device”
}
```

"Go fetch me the user"

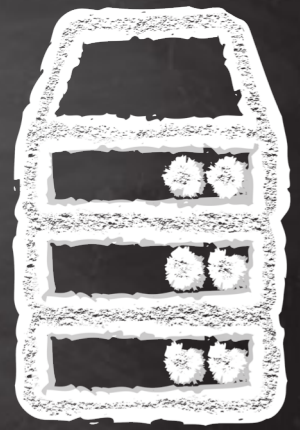
```
{
    "user_code_url":
        "https://server.example.com/interact/device",
    "user_code": "A1BC-3DFF",
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

Tell the user

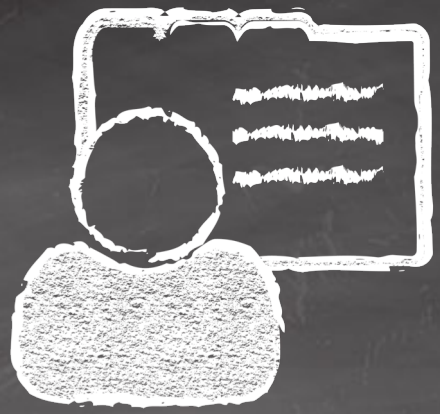


https://server.example.com/interact/device

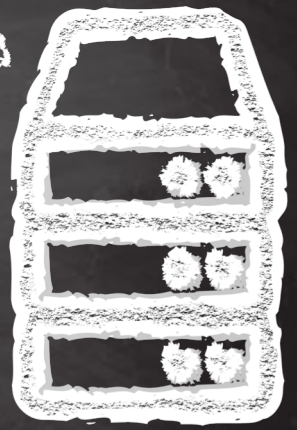
A1BC-3DFF



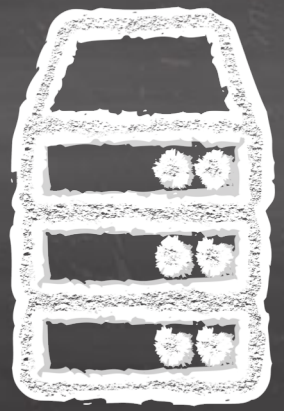
User interacts like you'd expect



- Authenticate
- Authorize
- Consent
- Modify
- A1BC-3DFF



Look up the transaction based on the user code



Are we ready yet?

```
{
    "handle": "80UPRY5NM33OMUKMKSKU"
}
```

Not ready yet

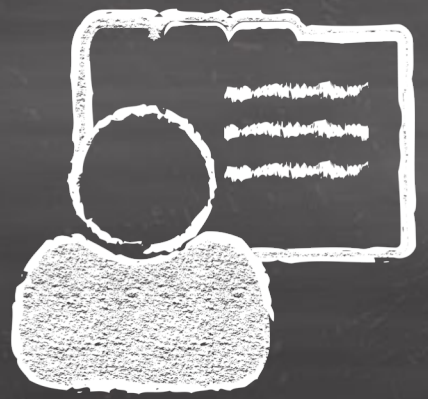
```
{
    "wait": 30,
    "handle": {
        "value": "BI9QNW6V9W3XFJK4R02D",
        "type": "bearer"
    }
}
```

What about a combined URL?

We can use the regular interaction URL

```
{
    "interaction_url":

"https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ",

    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

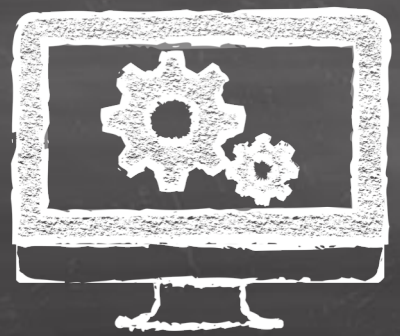
What about identity?



Pass identity assertions like OIDC, VC

```
{
    "access_token": ...
    "id_token": "eyj0...",
    "verifiable_claims": "..."
}
```

What about binding tokens?



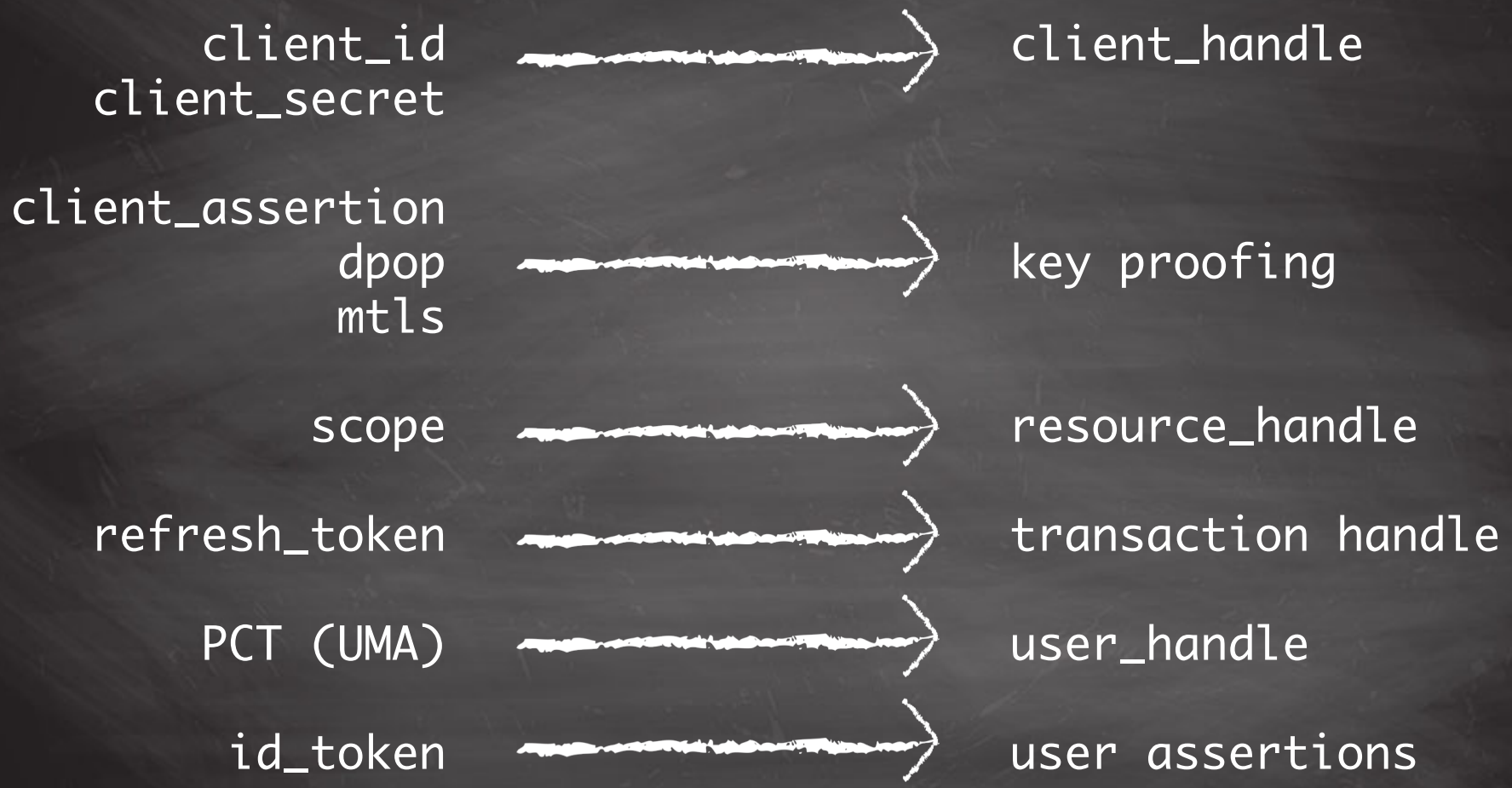
Access token is bound to a key

```
{
    "access_token": {
        "value": "0S9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "type": "jwsd",
        "key": {
            "kid": "token-1234", ...
        }
    }
}
```

Key proof is presented alongside token

```
Authorization: JWSD OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0
Detached-JWS: eyJiNjQiOmZhbHNlLCJhbGciOiJSU...
```

Mapping to OAuth2



Mapping to OAuth2

```
{
    "client": "client_id"
    "resources": ["scope1", "scope2"],
    "key": "client_id"
}
```

Pros and Cons

- Wider set of use cases
- More secure by default
- Built on existing experience
- Simpler data model
- Fewer moving parts
- Static and dynamic scenarios
- Multimodal JSON

- Not backwards compatible
- Different assumptions
- Different data model
- Multimodal JSON
- Unknown in large deployment scale
- We don’t know what’s broken yet

Working group item?

@justin_richer

https://bspk.io/

Questions?