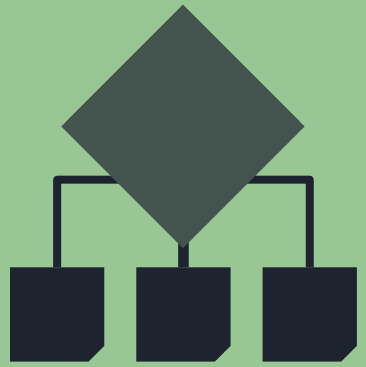


Delegation, not impersonation

The standards behind acting on behalf of users: from **ROPC** to **Token Exchange**



ROPC

Resource Owner Password Credentials

RFC 6749 §4.3

The password grant.

```
POST /oauth/token

grant_type=password

client_id=my-app

client_secret=********

username=alice@example.com

password=Tr0ub4dor&3

scope=read:orders
```

1. The normal token endpoint
2. Asks for the password grant
3. Which app is asking
4. The app authenticates itself
5. The user's login name
6. The user's real password
7. Access the app wants

Why ROPC hurts.

Password exposure

The client handles raw credentials.

Breaks MFA

Not designed for MFA or multi-step login.

No audit trail

Logs show the user, not who acted.

Unbounded access

No scope limit, no expiry, no approval.

How ROPC got phased out.

2012	**OAuth 2.0 · RFC 6749**	Allowed for highly privileged apps, only when other flows are not viable.
2020	**Token Exchange · RFC 8693**	Defines impersonation and delegation semantics.
2025	**Security BCP · RFC 9700**	ROPC must not be used.
2026	**OAuth 2.1 · Draft 17**	ROPC not specified.



OAuth gives us the tools to do better



RFC 8693

Token Exchange

RFC 8693

The token exchange request.

```
POST /oauth/token
Authorization: Bearer ...
grant_type=...:token-exchange
subject_token=eyJhbGci...
subject_token_type=...:jwt
actor_token=eyJhbGci...
actor_token_type=...:access_token
audience=https://api.example
scope=read:orders
```

1. The normal token endpoint
2. Client authenticates itself
3. Asks for a token exchange
4. Who it is about: the user
5. Format of that token
6. Who acts: the agent
7. Format of the actor token
8. Where the new token will work
9. How much access is wanted

Impersonation vs delegation

IMPERSONATION

```
{
  "iss": "https://as.example",
  "sub": "user_123",
  "aud": "https://api.example",
  "scope": "orders"
}
```

↳ Looks like the user

DELEGATION

```
{
  "iss": "https://as.example",
  "sub": "user_123",
  "act": { "sub": "agent_42" },
  "aud": "https://api.example"
}
```

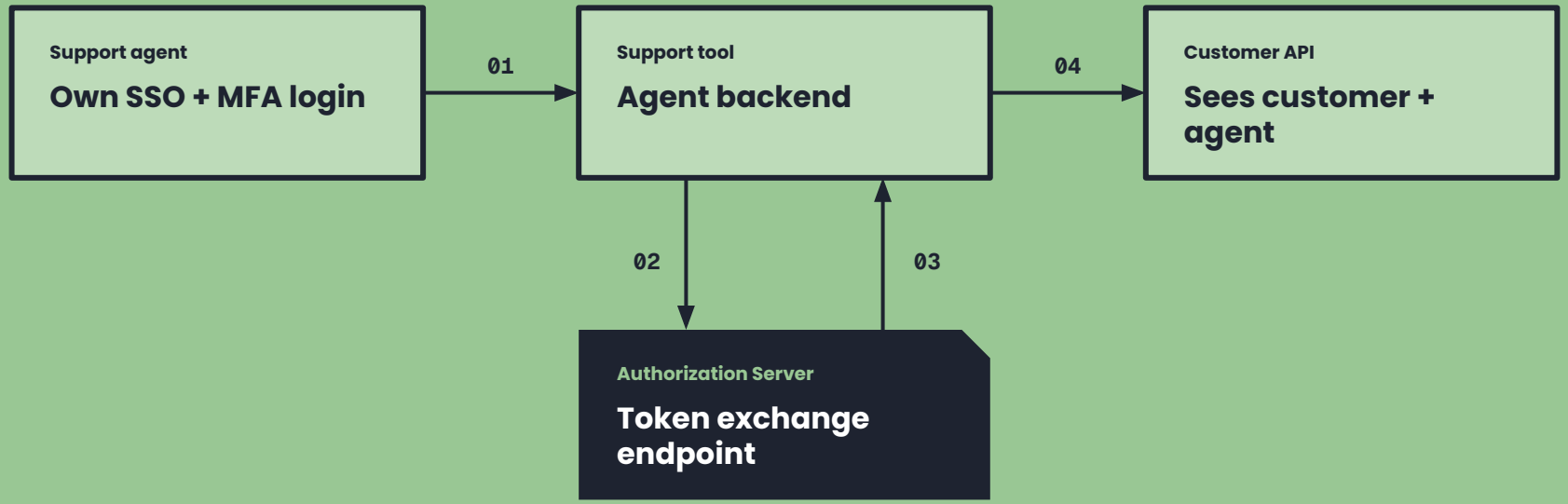
↳ Names who is acting



Pattern 1

On Behalf Of

On Behalf Of

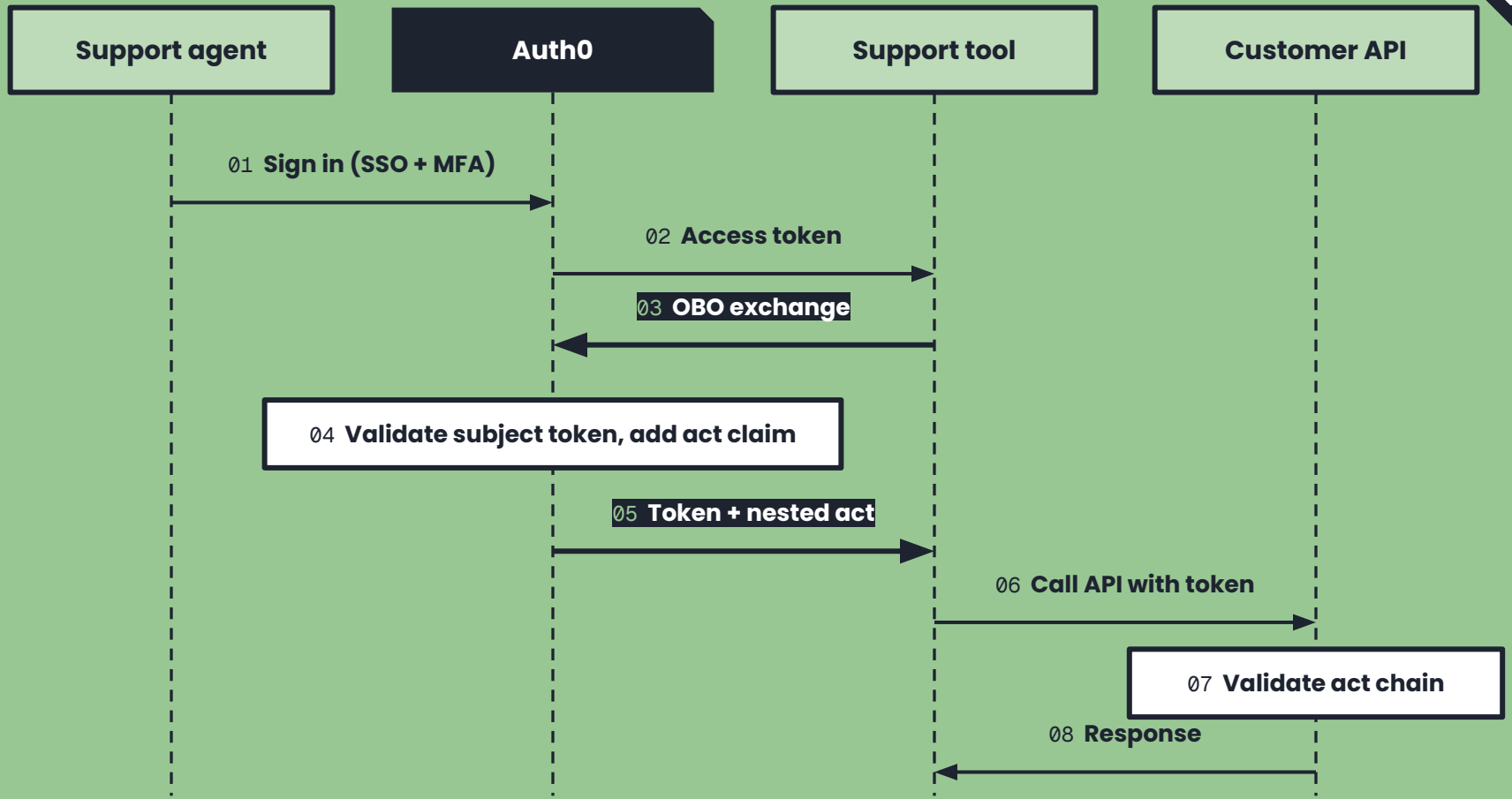


01 Agent's token reaches the tool

02 Tool asks AS: customer + agent

03 AS checks policy, issues token

04 API call carries both identities


Support agent
Auth0
Support tool
Customer API
01 Sign in (SSO + MFA)
02 Access token
03 OBO exchange
04 Validate subject token, add act claim
05 Token + nested act
06 Call API with token
07 Validate act chain
08 Response

On Behalf Of

REQUEST

```
POST /oauth/token
grant_type=...:token-exchange
subject_token=eyJhbGci...  (customer)
subject_token_type=...:jwt
actor_token=eyJhbGci...  (agent)
actor_token_type=...:id_token
audience=https://api.example
scope=read:orders
```

ISSUED TOKEN

```
{
  "iss": "https://as.example",
  "sub": "cust_123",
  "aud": "https://api.example",
  "scope": "read:orders",
  "act": { "sub": "agent_42" }
}
```

Nested actors: the delegation chain

```
{
  "sub": "user_123",
  "aud": "https://flights-api.example",
  "act": {
    "sub": "booking-subagent",
    "act": {
      "sub": "planner-agent",
      "act": {
        "sub": "assistant-app"
      }
    }
  }
}
```

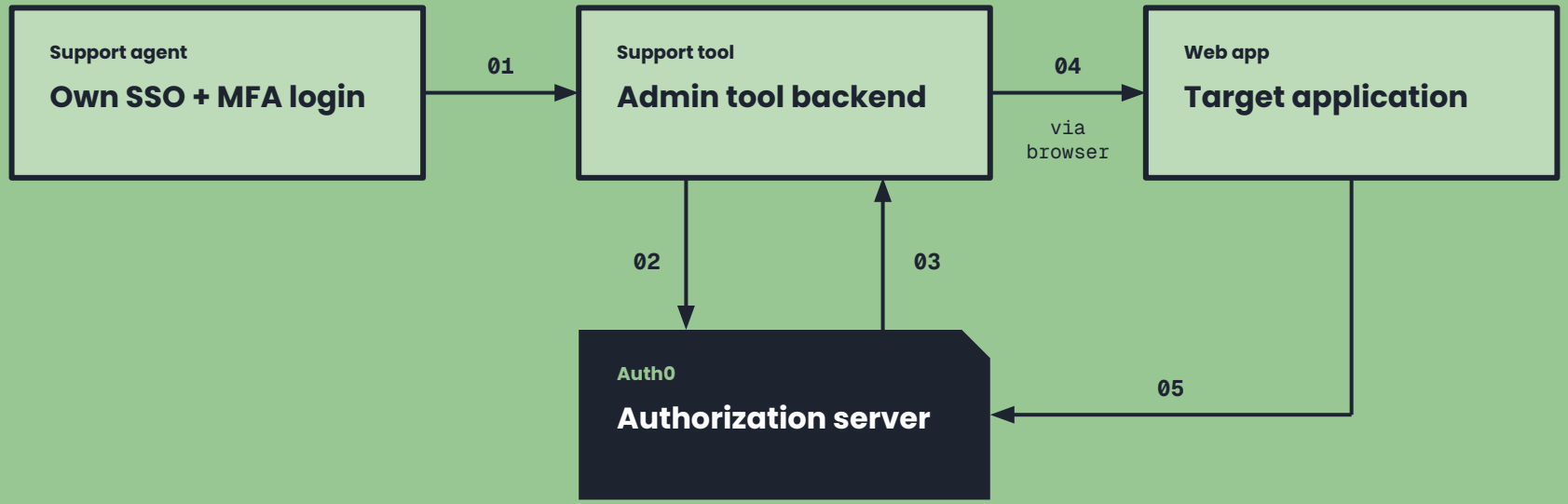
1. Who it is about
2. Acting now
3. Acted before it
4. Where it started



Pattern 2

Session Delegation

Session Delegation



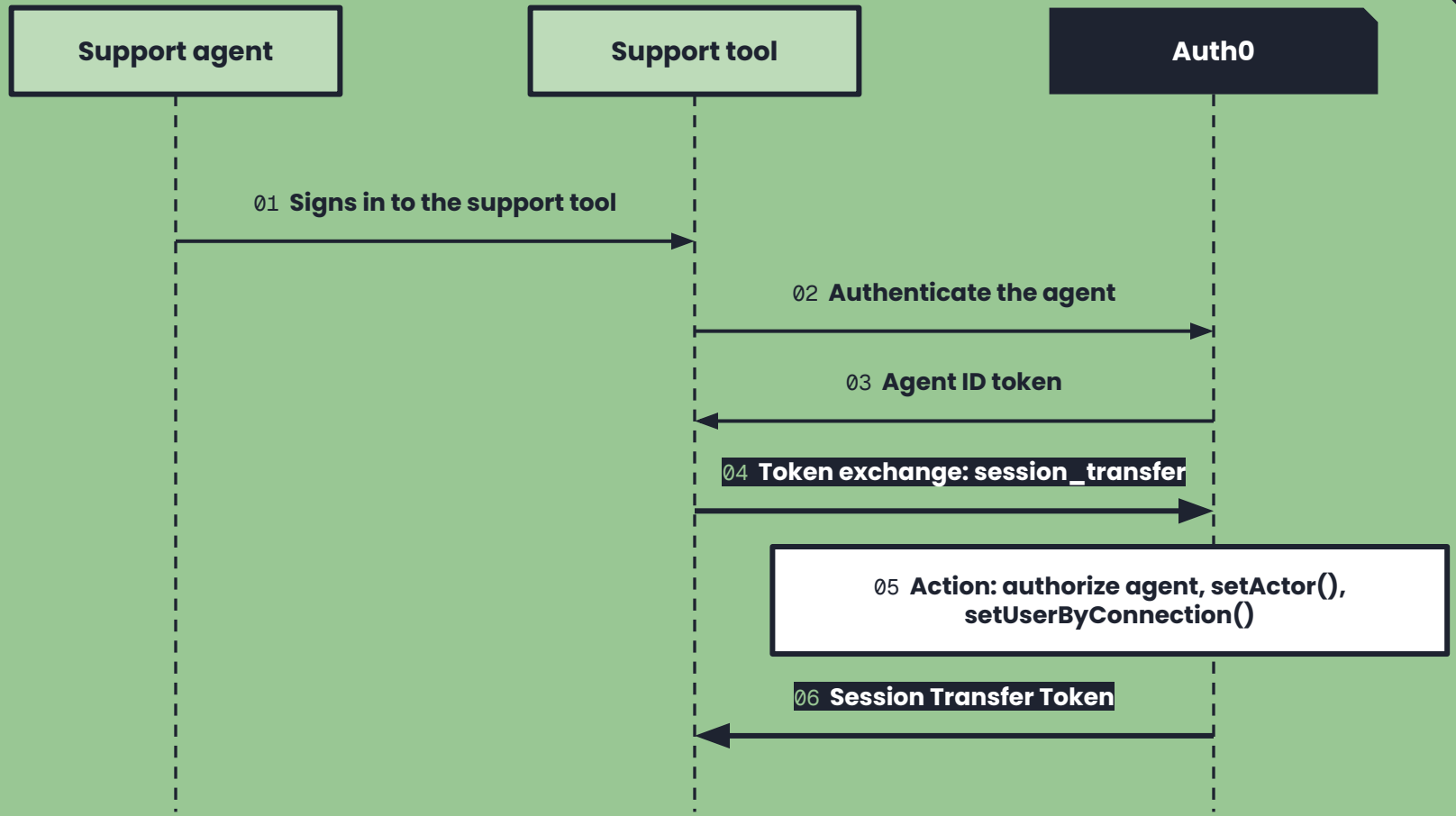
01 **Agent signs in**

02 **Tool requests exchange**

03 **Transfer token returned**

04 **Redirect to the web app**

05 **App redeems at /authorize**


Support agent
Support tool
Auth0
01 Signs in to the support tool
02 Authenticate the agent
03 Agent ID token
04 Token exchange: session_transfer
05 Action: authorize agent, setActor(), setUserByConnection()
06 Session Transfer Token

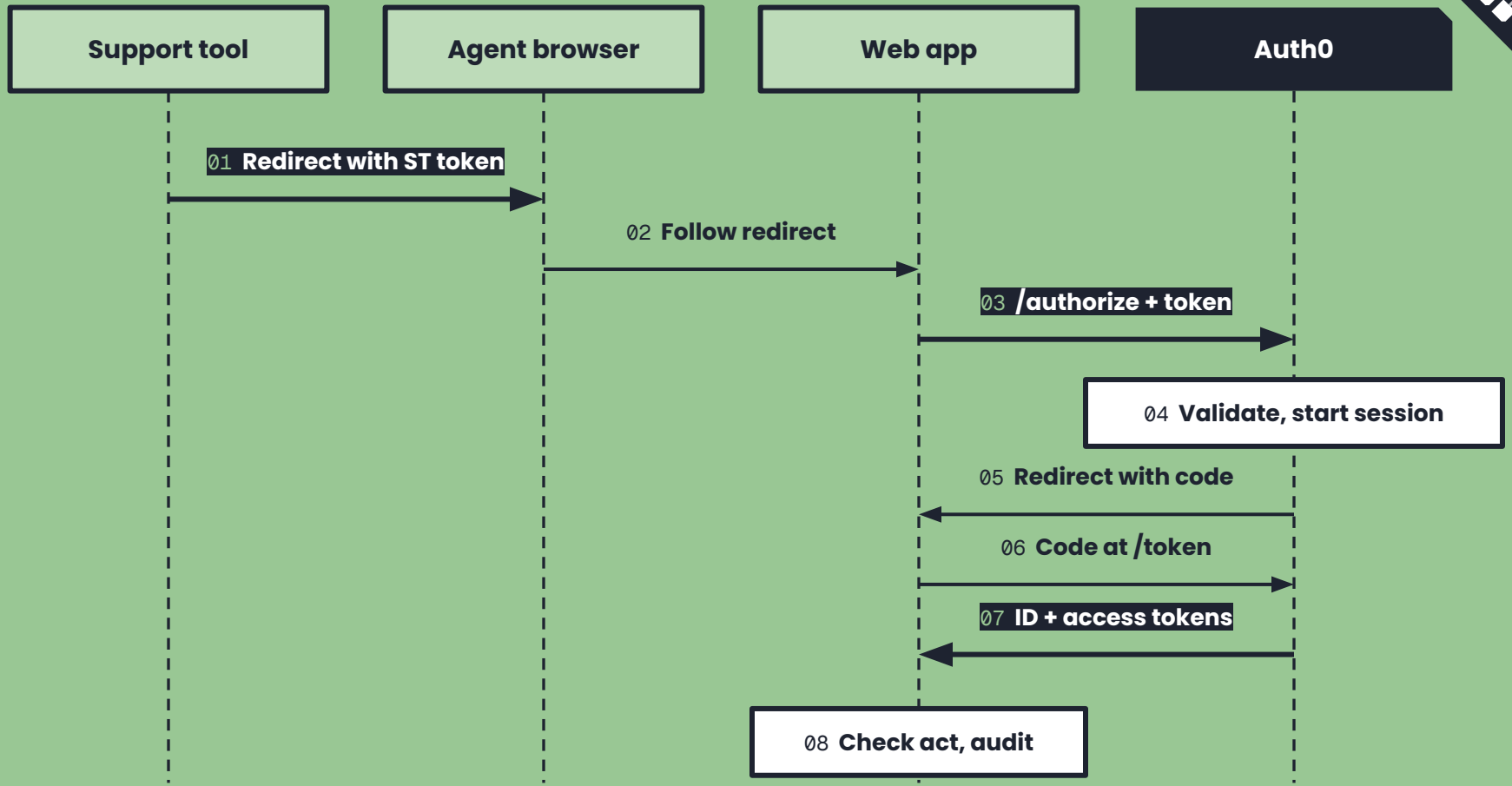
Get Session Transfer Token

REQUEST

```
POST /oauth/token
grant_type=...:token-exchange
subject_token=eyJhbGci...  (customer)
subject_token_type=urn:acme:support
actor_token=eyJhbGci...  (agent)
actor_token_type=...:id_token
audience=urn:<tenant>:session_transfer
client_id=support-tool
```

RESPONSE

```
{
  "access_token":
    "<session transfer token>",
  "issued_token_type":
    "...:session_transfer_token",
  "expires_in": 60
}
```


Support tool
Agent browser
Web app
Auth0
01 Redirect with ST token
02 Follow redirect
03 /authorize + token
04 Validate, start session
05 Redirect with code
06 Code at /token
07 ID + access tokens
08 Check act, audit

Get Delegated Access Token

REQUEST

```
GET /authorize
  ?session_transfer_token=...
  &client_id=...
  &redirect_uri=...
  &response_type=code
```

TOKENS AFTER CODE EXCHANGE

```
{
  "sub": "auth0|end_user_id",
  "act": {
    "sub": "auth0|agent_id",
    "sub_profile": "human",
    "role": "support"
  }
}
```

Same exchange, different token

More patterns.

ID-JAG DRAFT

Cross-App Access

App-to-app access through a trusted IdP

IETF DRAFT

Transaction Tokens

User and workload identity along an internal call chain

IETF DRAFT

Identity Chaining

Identity and authorization across trust domains



Sam Bellen

Principal Developer Advocate at Auth0

@sambego · sambego.tech

Why you have to move.

RFC 9700 · JAN 2025

MUST NOT

Exposes user credentials to the client

Not designed for 2FA or multi-step login

OAUTH 2.1 · DRAFT-17

Not specified

ROPC and Implicit are left out

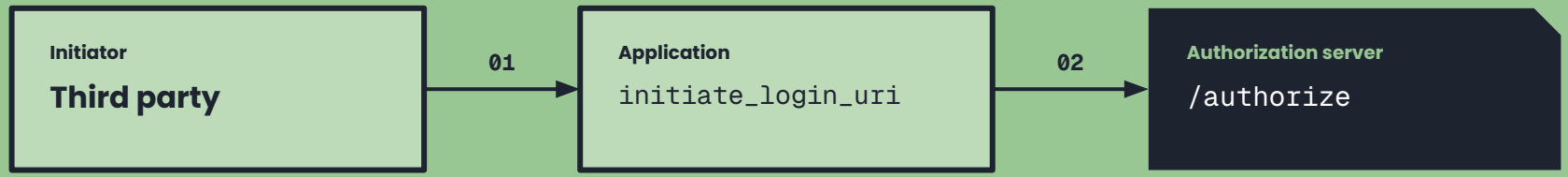
Three grants remain: auth code + PKCE, client credentials, refresh token



RFC 8693

Initiating Login

OpenID Connect: Initiating Login



OIDC CORE §4

```
GET /login
  ?iss=https://as.example
  &login_hint=...
  &target_link_uri=...
```

SESSION DELEGATION

```
GET /initiate-login
  ?session_transfer_token=...
  &organization=...
```