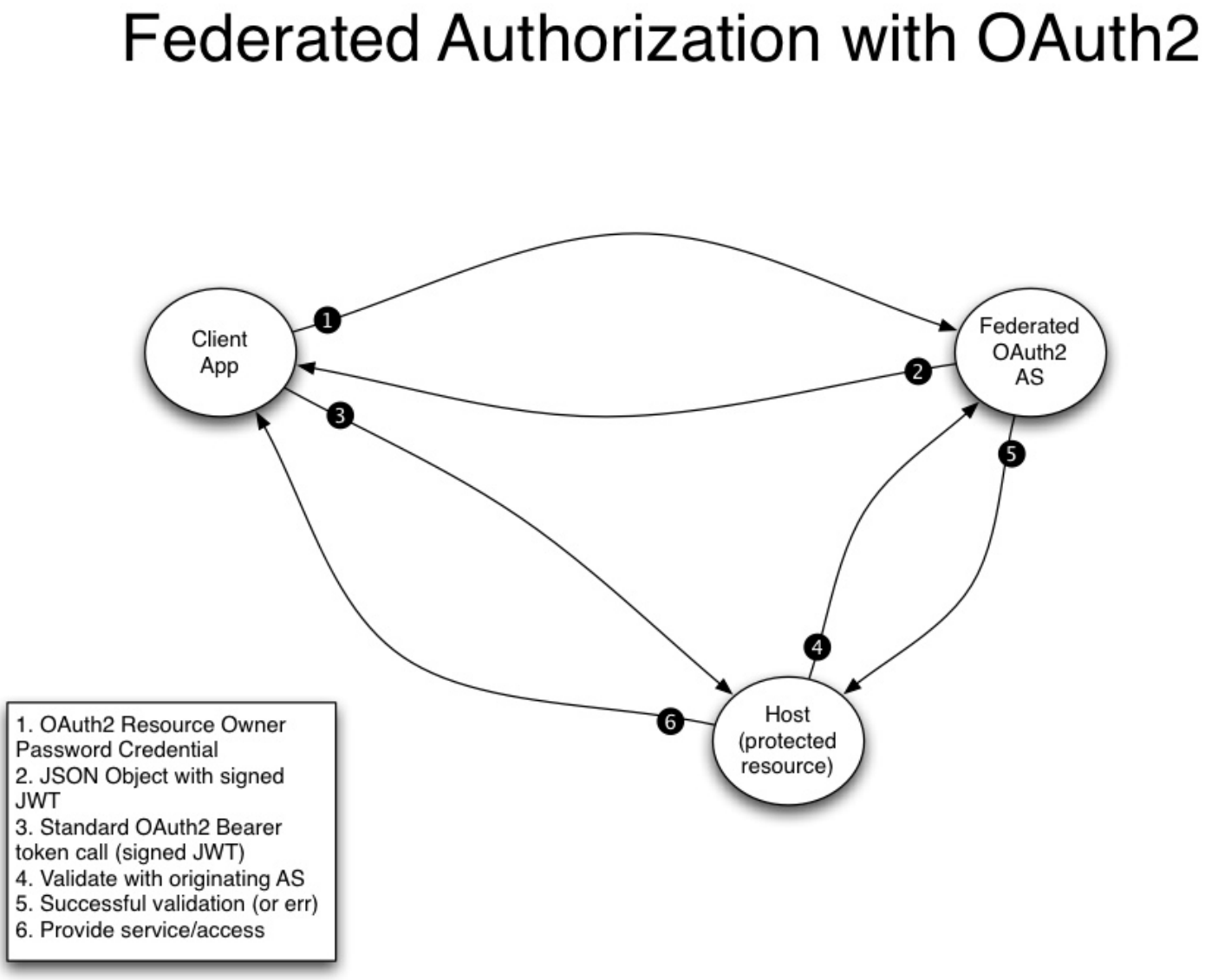
Overall flow for supporting Federated OAuth2 tokens

Enabling support of federated OAuth2 tokens comprises three main steps:
1. Get an OAuth2 token from the user's Authorization Server
2. Present the token to the federated resource server
3. Federated resource server validates the OAuth2 token with the token issuer



Step 1: Get an OAuth2 token from the user's Authorization Server (OAuth2 Resource Owner Password Credential flow)

```
POST /token HTTP/1.1
Host: as.oauth.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=password&client_id=s6BhdRkqt3&
username=johndoe&password=A3ddj3w&scope="user-info"
```

Step 2: Receive token response from Authorization Server

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{

 "access_token":"ewogaXNzIDogaHR0cHM6Ly9hcy5vYXV0aC5leGFtcGxlLmNvbSwKIHVpZCA6IGJzZ
HNhc2RmOGFhZmhzc2Q5ZGZnaHNzZGZncywKIGFjY2Vzc190b2tlbiA6IGFzZGZhZXJnMmdkZGYzN
DY1MzV0d2RmZ3c0Nndnd2V0MzQsCiB2YWxpZGF0ZVVSTCA6IGh0dHBzOi8vYXMub2F1dGguZXhh
bXBsZS5jb20vdmFsaWRhdGUKfQ",
   "token_type":"bearer",
   "expires_in":3600,
   "refresh_token":"9as23fvsdf34t"
}
```

Step 3: Present access_token to federated protected resource (host)

```
GET /api/myService?params=asdfasdf&... HTTP/1.1
Host: resource.example.net

Authorization: Bearer
ewogaXNzIDogaHR0cHM6Ly9hcy5vYXV0aC5leGFtcGxlLmNvbSwKIHVpZCA6IGJzZHNhc2RmOGFhZ
mhzc2Q5ZGZnaHNzZGZncywKIGFjY2Vzc190b2tlbiA6IGFzZGZhZXJnMmdkZGYzNDY1MzV0d2RmZ3
c0Nndnd2V0MzQsCiB2YWxpZGF0ZVVSTCA6IGh0dHBzOi8vYXMub2F1dGguZXhhbXBsZS5jb20vdm
FsaWRhdGUKfQ
Content-Type: application/x-www-form-urlencoded
```

Step 4: Validate OAuth2 token with token issuer

```
POST /validate HTTP/1.1
Host: as.oauth.example.com

Authorization: Bearer
ewogaXNzIDogaHR0cHM6Ly9hcy5vYXV0aC5leGFtcGxlLmNvbSwKIHVpZCA6IGJzZHNhc2RmOGFhZ
mhzc2Q5ZGZnaHNzZGZncywKIGFjY2Vzc190b2tlbiA6IGFzZGZhZXJnMmdkZGYzNDY1MzV0d2RmZ3
c0Nndnd2V0MzQsCiB2YWxpZGF0ZVVSTCA6IGh0dHBzOi8vYXMub2F1dGguZXhhbXBsZS5jb20vdm
FsaWRhdGUKfQ
```

Step 5: Validation response from token issuer

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
      “uid”: “bsdsasdf8aafhssd9dfghssdfgs”,
}
```

Step 6: Return requested resource or service. Response is resource/application defined

Possible structure of the Signed JWT returned as the OAuth2 bearer token in Step 2

```
{
  "iss" : "https://as.oauth.example.com",
  “uid” : “bsdsasdf8aafhssd9dfghssdfgs”,
  "access_token" : "asdfaerg2gddf346535twdfgw46wgwet34",
  "validateURL" : "https://as.oauth.example.com/validate"
}
```

Instead of requiring the host (protected resource) to validate all federated OAuth2 tokens with the token issuer, it’s possible to use short lived signed JWTs (JWS) that can be validated locally by the host. Determining which model to use depends on the risk profile of the use case being implemented.