Open Authentication

Saravanan

Authentication & Authorization

Authentication

- Establishment of a binding of confidence between and entity and an identity

Authorization

- Process of establishing the rights for the authenticated user

Why AuthN & AuthZ

- To avoid insecure resource access
- To give finer control on the resource access
- To track the various actions performed on resources by the doer’s
- Increasing variations in resource consumers
- Overcoming security breaches

Ways to Achieve

- Authentication
  - Username / Password
  - Certificates
  - Access tokens / established identity etc…
  - Finger print / Retina Scan etc…
- Authorization
  - Roles
  - Policies

Authorization Background

- Policy Phases
  - Definition
  - Enforcement
- Access Control Lists / Capability
  - Principle of least privilege
- Tokens
  - Anonymous identity support

Need for OAuth

Problem

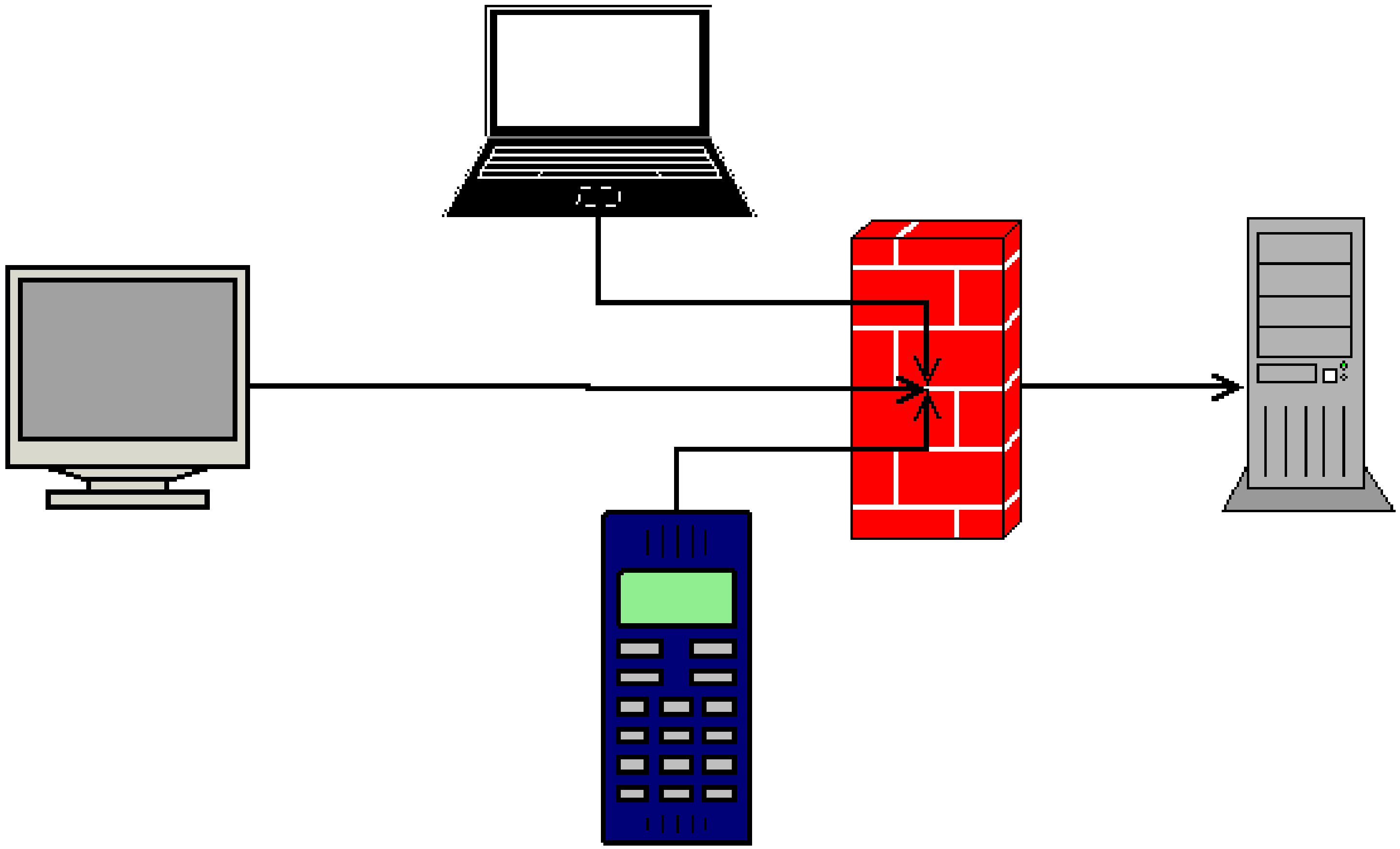
- Present day has Multitudes of
  - Applications
  - Identities
- Hard to remember authentication information among above

Solution

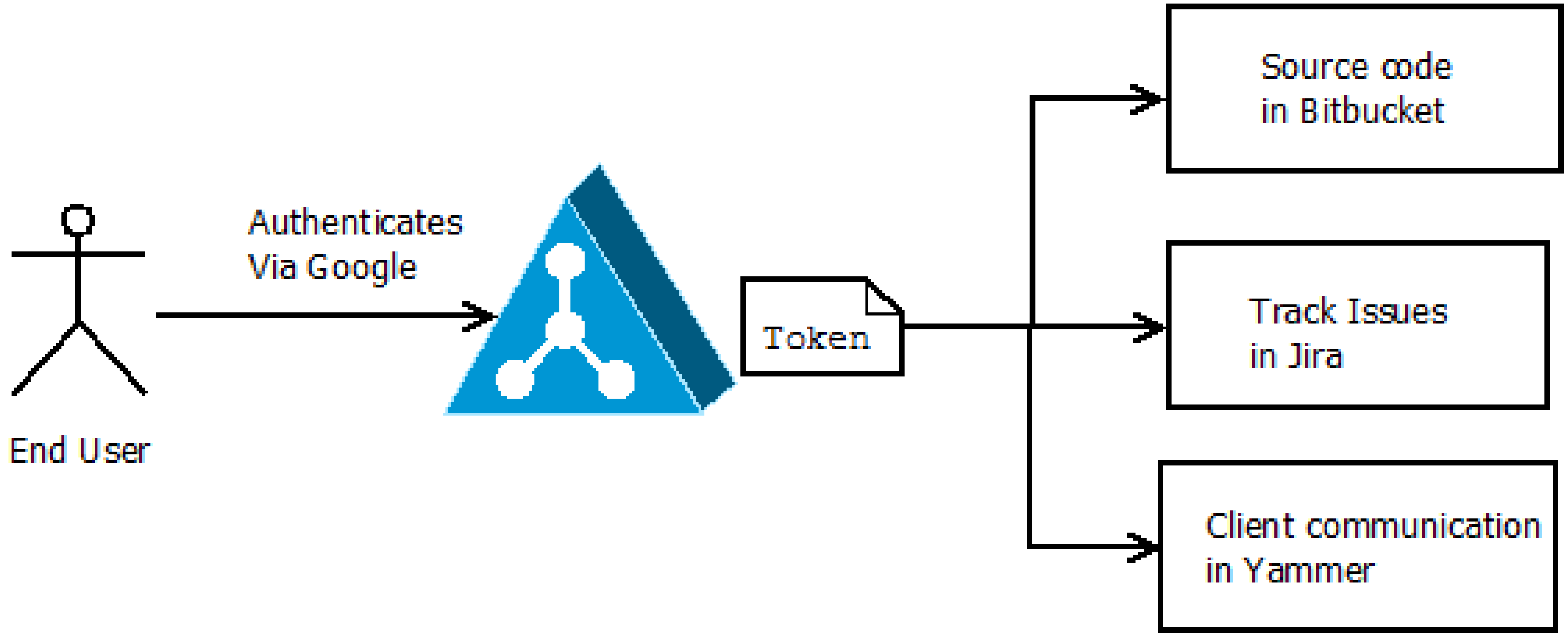
- Delegated Authentication & Authorization

Use Case

Multitude of devices for accessing 1 application



SSO Use Case


Authenticates
Via Google
End User
Token
Source code
in Bitbucket
Track Issues
in Jira
Client communication
in Yammer

Problems Addressed in OAuth

The Problem

1. Credentials Sharing
2. Unrestricted Access
3. Servers are required to handle authentication & authorization
4. Difficulty in revoking
5. Huge chain of dependencies
6. Security breach

Solution

1. Abstracting the authorization layer from the client & server
2. No more password sharing
3. Access Tokens / Valet Keys with lifetimes
4. Takes place over HTTPS / SSL
5. Concealed / isolated identity

OAuth 2.0

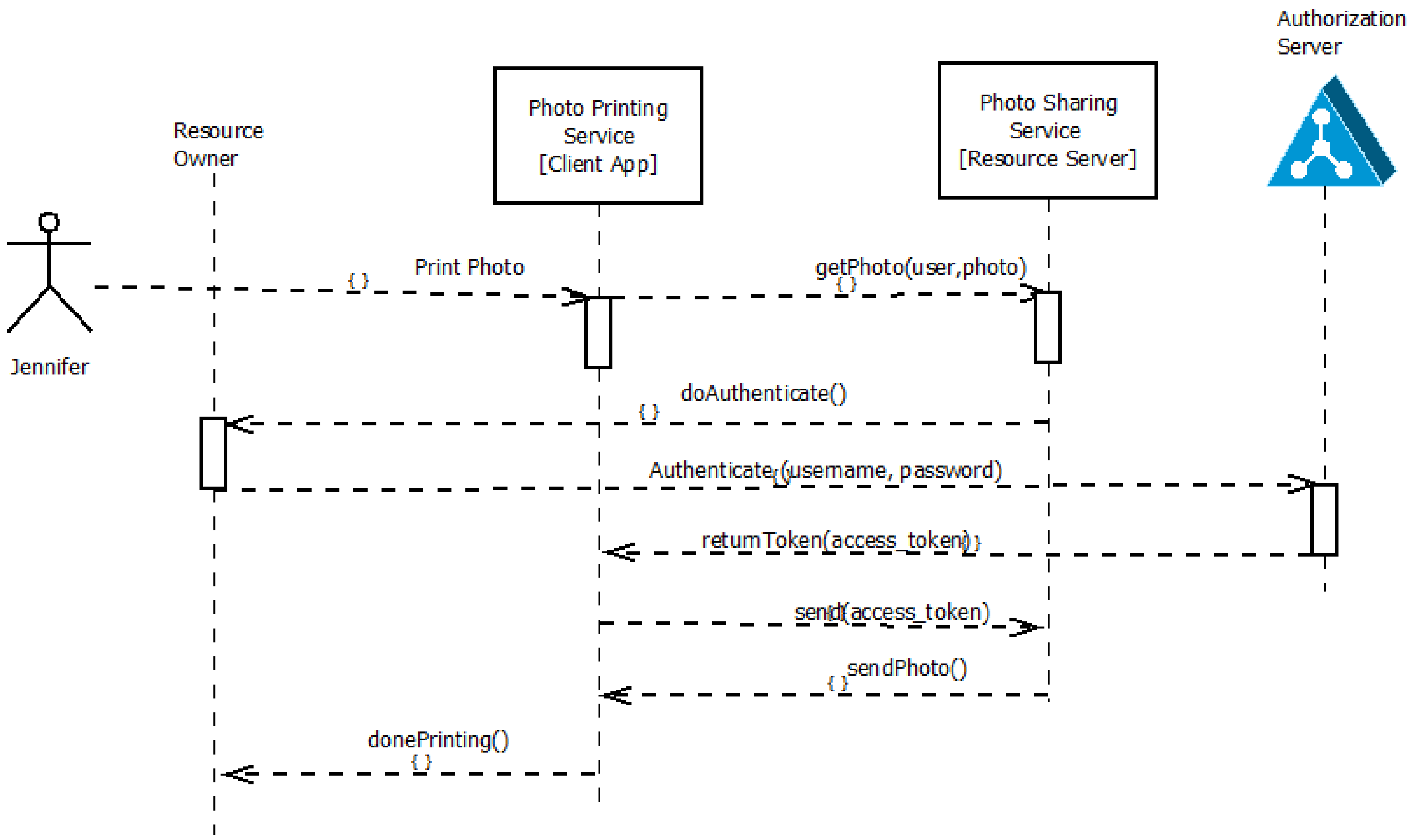
What is OAuth

- OAuth 2.0 is an Authorization Framework
- Framework specifying
  - Authentication & Authorization delegation
  - Interactions in the delegation process

Specification

- Google, Yammer & Bitbucket all speak through OAuth.
- Developed in 2006 by Twitter & Ma.gnolia
- Evolved from 1.0 to 2.0
- Main problem targeted by OAuth is Access Delegation

Use Case


Authorization Server
Resource Owner
Photo Printing Service [Client App]
Photo Sharing Service [Resource Server]
Jennifer
Print Photo
getPhoto(user,photo)
doAuthenticate()
Authenticate (username, password)
returnToken(access_token)
send(access_token)
sendPhoto()
donePrinting()

Roles in OAuth2.0

Resource [R]

- A HTTP Resource / Service / App

Resource Owner [RO]

- Entity that is capable of granting access to a resource

Resource Server [RS]

- Protected resource Host

Client Application [CA]

- Application making request to RS on behalf of RO to gain access to R

Authorization Server [AS]

- Generates tokens after authenticating the RO and obtaining authorization

Flow

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Authorization Grant

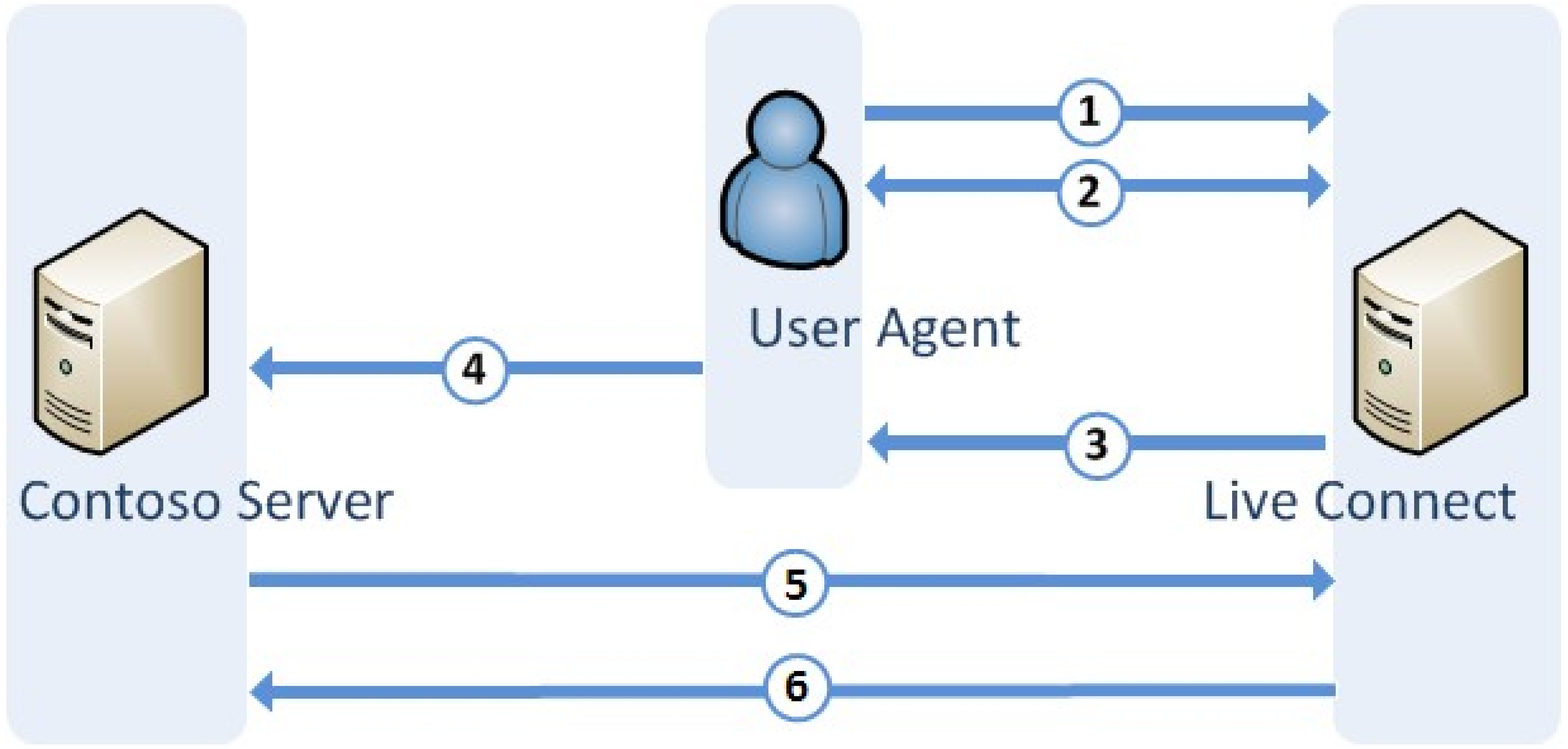
Server-side Scenarios

Target Applications

1. Any app that is web enabled / Desktop
2. Application that can access a browser

```
     +----------+
     | Resource |
     |   Owner  |
     |          |
     +----------+
          ^
          |
         (B)
     +----|-----+          Client Identifier      +---------------+
     |         -+----(A)-- & Redirection URI ---->|               |
     |  User-   |                                 | Authorization |
     |  Agent  -+----(B)-- User authenticates --->|     Server    |
     |          |                                 |               |
     |         -+----(C)-- Authorization Code ---<|               |
     +-|----|---+                                 +---------------+
       |    |                                         ^      v
      (A)  (C)                                        |      |
       |    |                                         |      |
       ^    v                                         |      |
     +---------+                                      |      |
     |         |>---(D)-- Authorization Code ---------'      |
     |  Client |          & Redirection URI                  |
     |         |                                             |
     |         |<---(E)----- Access Token -------------------'
     +---------+       (w/ Optional Refresh Token)
```

Microsoft Implementation

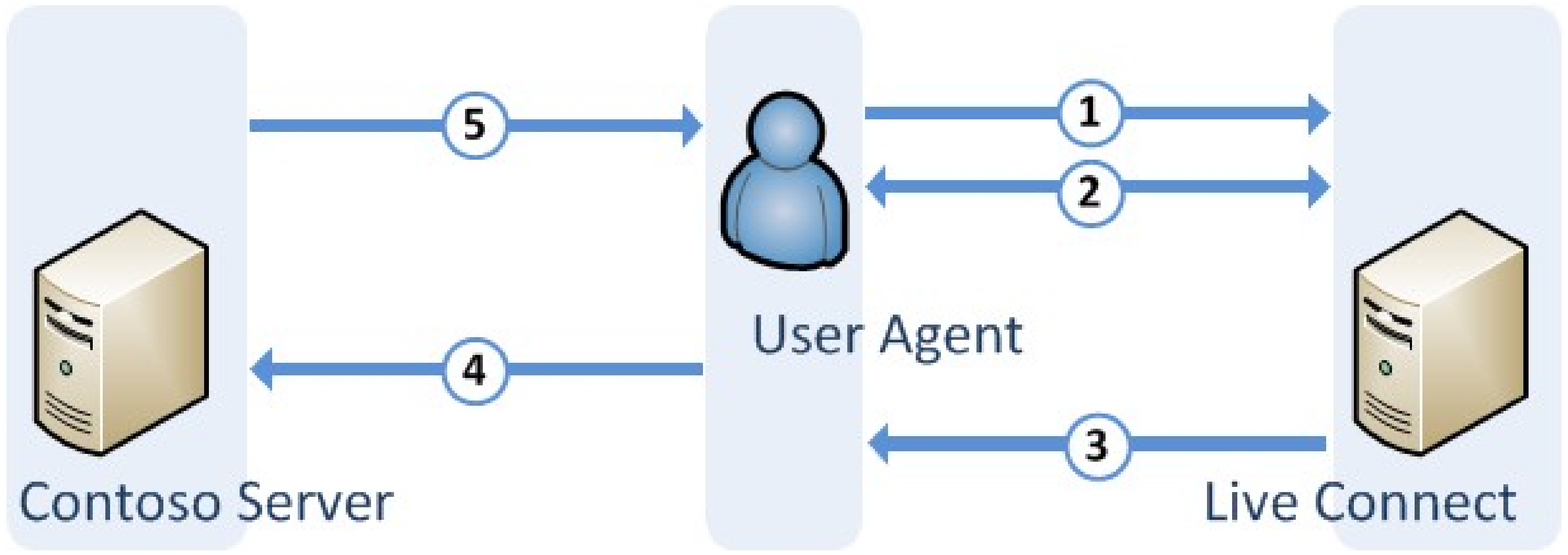


Implicit Grant

- Scripted client access
  - Ex: Google Ad services API
- For well known clients
- No client validation happens
- Access Token sent as a fragment in the response

```
+----------+
| Resource |
|  Owner   |
|          |
+----------+
     ^
     |
    (B)
+----|-----+          Client Identifier     +---------------+
|         -+----(A)-- & Redirection URI --->|               |
|  User-   |                                | Authorization |
|  Agent  -|----(B)-- User authenticates -->|     Server    |
|          |                                |               |
|          |<---(C)--- Redirection URI ----<|               |
|          |          with Access Token     +---------------+
|          |            in Fragment
|          |                                +---------------+
|          |----(D)--- Redirection URI ---->|   Web-Hosted  |
|          |          without Fragment      |     Client    |
|          |                                |    Resource   |
|     (F)  |<---(E)------- Script ---------<|               |
|          |                                +---------------+
+-|--------+
  |    |
 (A)  (G) Access Token
  |    |
  ^    v
+---------+
|         |
|  Client |
|         |
```

Microsoft Implementation



Resource Owner Flow

- Fully trusted applications
- Not very secure
- Maintained for backward compatibility
- Use of existing data to generate the access tokens

```
+----------+
| Resource |
|  Owner   |
|          |
+----------+
     v
     |    Resource Owner
    (A) Password Credentials
     |
     v
+---------+                                  +---------------+
|         |>--(B)---- Resource Owner ------->|               |
|         |         Password Credentials     | Authorization |
| Client  |                                  |     Server    |
|         |<--(C)---- Access Token ---------<|               |
|         |    (w/ Optional Refresh Token)   |               |
+---------+                                  +---------------+
```

Sample

OpenID Connect

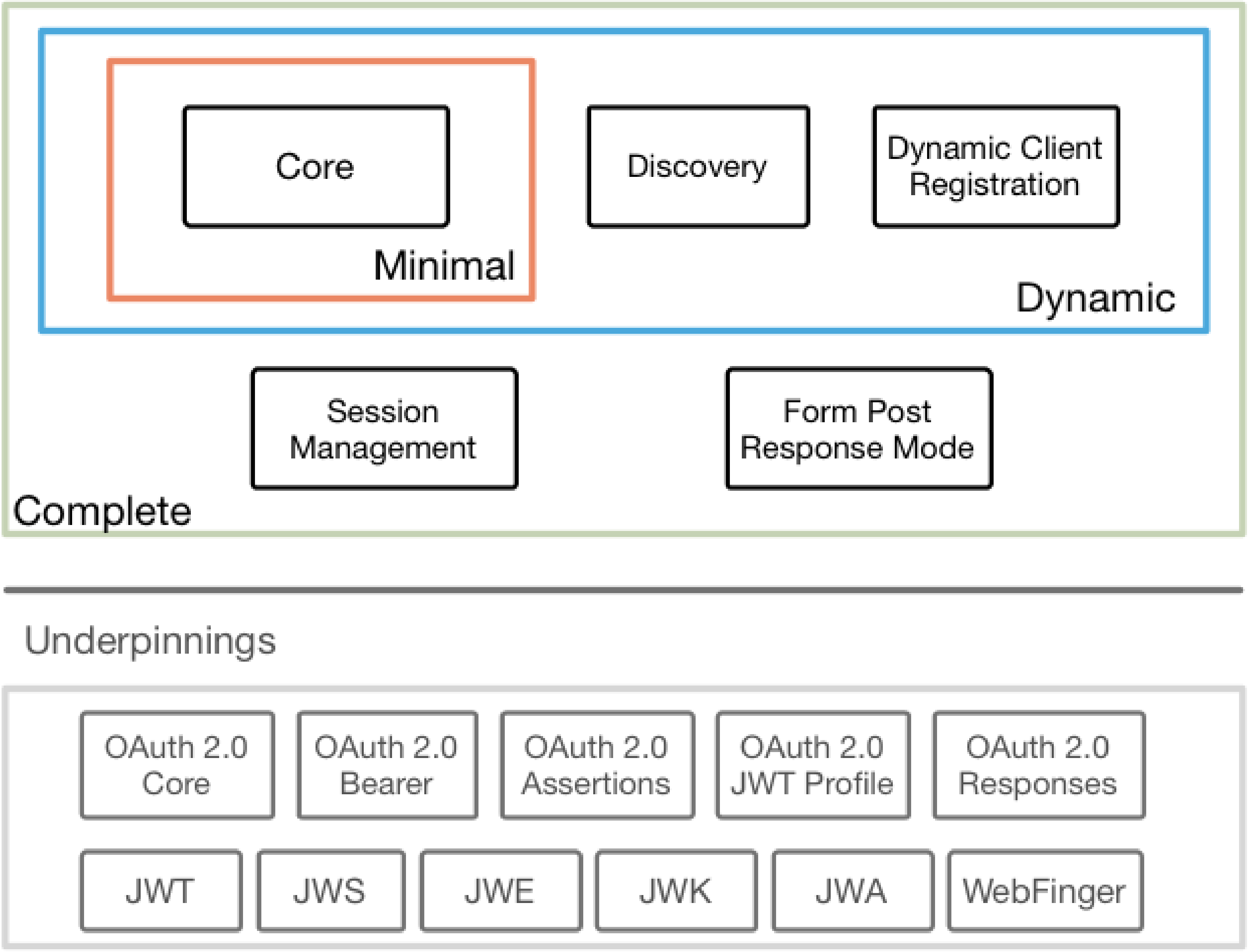
OpenID Connect

- Why OpenID Connect
  - No responsibility of apps to maintain passwords
  - Uses Claims to transfer profile information across diverse apps
- How does it work
  - (Identity, Authentication) + OAuth 2.0 = OpenID Connect
- System-level support
  - Android OS
  - Windows Server 2012 – R2 [ADFS 3.0]
- OpenID makes use of OAuth 2 flows to establish identity

4 Feb 2014

http://openid.net/connect

OpenID Connect Protocol Suite



OpenID 2.0 & OpenID Connect

- Compared to OpenID2.0, OpenID Connect uses
  - JWT Data Structures
  - Simplified signing of tokens
  - No XML
  - Highly interoperable

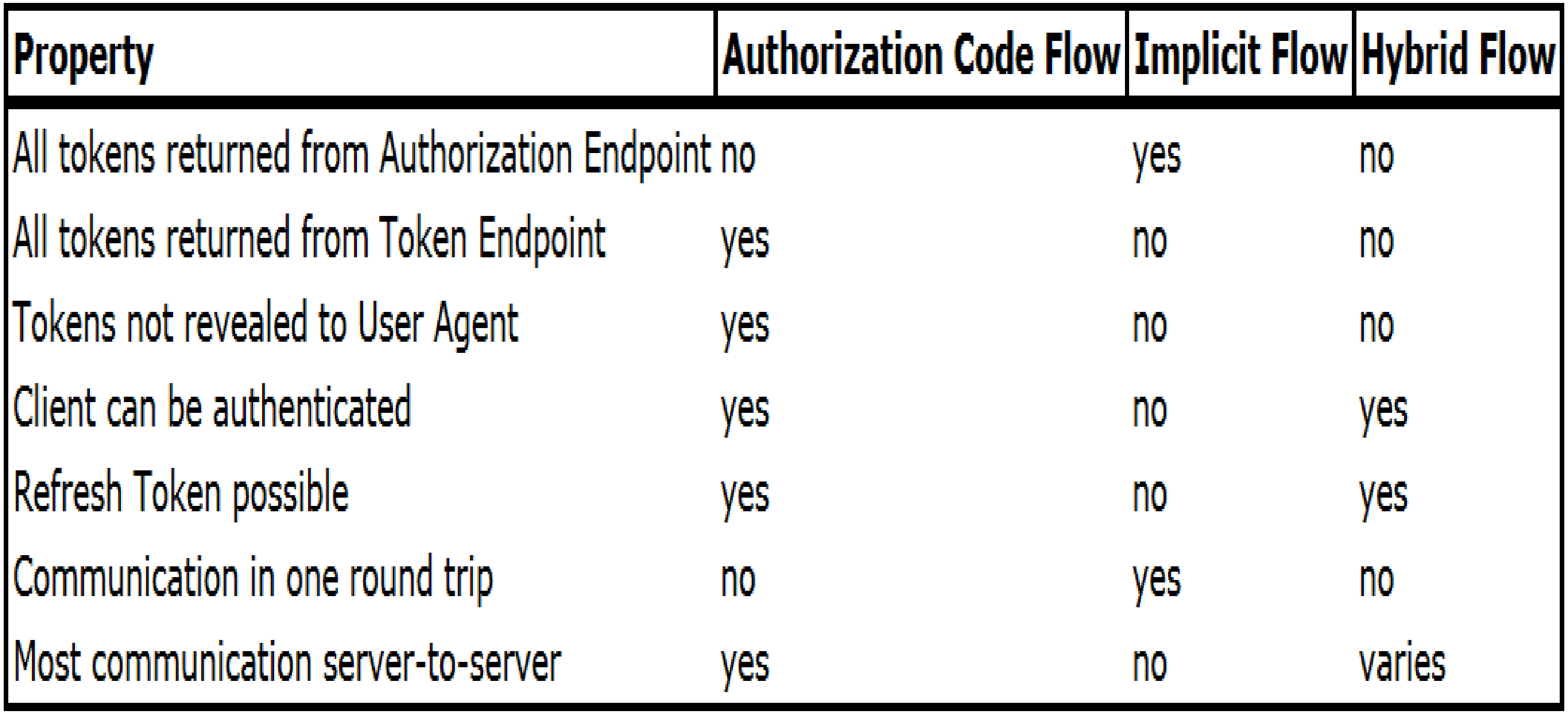
OpenID Connect Glossary

- IDP [AaaS]
  - Any service that provides identity and authentication
- RP
  - App that out sources its authentication to an IDP
- OP
  - The OpenID provider
- Claims
  - Piece of information about an entity / identity

Flow

```
+--------+                                   +--------+
|        |                                   |        |
|        |---------(1) AuthN Request-------->|        |
|        |                                   |        |
|        |  +--------+                       |        |
|        |  |        |                       |        |
|        |  |  End-  |<--(2) AuthN & AuthZ-->|        |
|        |  |  User  |                       |        |
|   RP   |  |        |                       |   OP   |
|        |  +--------+                       |        |
|        |                                   |        |
|        |<--------(3) AuthN Response--------|        |
|        |                                   |        |
|        |---------(4) UserInfo Request----->|        |
|        |                                   |        |
|        |<--------(5) UserInfo Response-----|        |
|        |                                   |        |
+--------+                                   +--------+
```

Authentication Flows



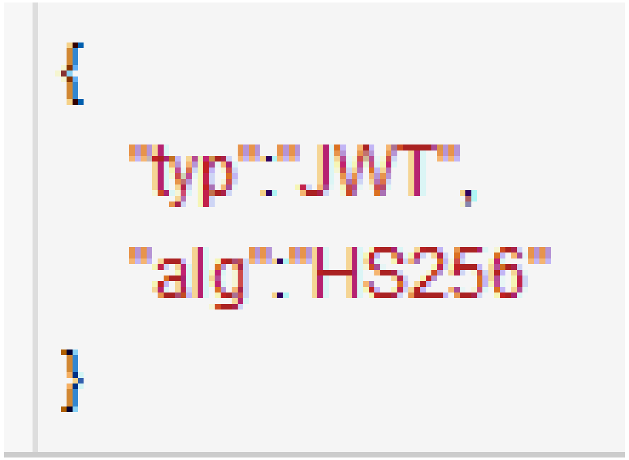
Property	Authorization Code Flow	Implicit Flow	Hybrid Flow
All tokens returned from Authorization Endpoint	no	yes	no
All tokens returned from Token Endpoint	yes	no	no
Tokens not revealed to User Agent	yes	no	no
Client can be authenticated	yes	no	yes
Refresh Token possible	yes	no	yes
Communication in one round trip	no	yes	no
Most communication server-to-server	yes	no	varies

Sample JWT

- eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjEz
  ODY4OTkxMzEsImlzcyI6ImppcmE6MTU0ODk1OTUiLC
  Jxc2giOiI4MDYzZmY0Y2ExZTQxZGY3YmM5MGM4YW
  I2ZDBmNjIwN2Q0OTFjZjZkYWQ3YzY2ZWE3OTdiNDY

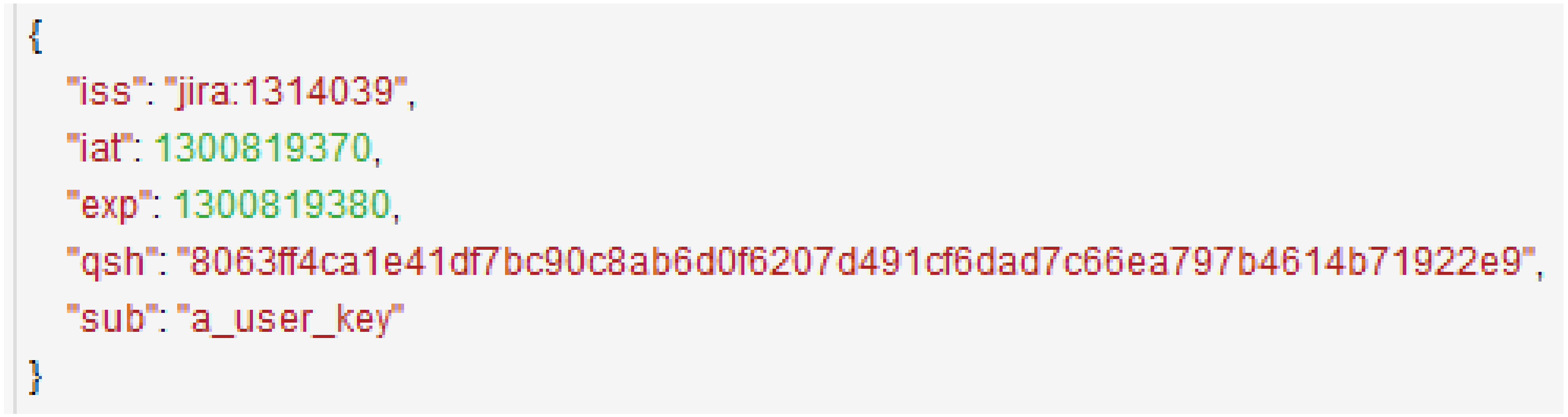
  xNGI3MTkyMmU5IiwiaWF0IjoxMzg2ODk4OTUxfQ.uKq
  U9dTB6gKwG6jQCuXYAiMNdfNRw98Hw_IWuA5MaMo
- <base64-encoded header>.<base64-encoded claims>.<base64-encoded signature>

JWT Header



```
{
   "typ":"JWT",
   "alg":"HS256"
}
```

JWT Claims



```
{
   "iss": "jira:1314039",
   "iat": 1300819370,
   "exp": 1300819380,
   "qsh": "8063ff4ca1e41df7bc90c8ab6d0f6207d491cf6dad7c66ea797b4614b71922e9",
   "sub": "a_user_key"
}
```

OpenID & SAML

- SAML
  - For web based apps
  - Uses XML
- OpenID Connect
  - JSON
  - REST
  - Any app [Native, Mobile, Web]

Realtime Implementation

- Authorization Server in TechCello
  - OpenID Connect 1.0
  - OAuth 2.0

- Supported Modes
  - Social Logins [MSFT, GOOG, FB, TWT]
  - WAAD
  - ADFS 3.0
  - LDAP
  - Proprietary Authentication exposed as an OP

Points to Ponder Upon

- Automated OP Discovery
- Automated Client Registration

THANK
YOU
for
LISTENING
ANY QUESTIONS?