Introduction à OpenIDConnect

COURS

ANF Authentification – Mathrice
Angers – 25/09/2014

laurent.facq@math.u-bordeaux1.fr
Institut de Mathématiques de Bordeaux

« OpenID » vu de l'utilisateur

- **L'utilisateur** veut accéder à un **service** sur internet
- Il découvre que ce service est accessible via des **comptes de services tiers** (google, facebook, twitter)
- Il choisit un de ces **fournisseurs d'identité**
  - pour se créer une identité

    ou
  - utiliser une de ses identités déjà existante
- Il **s'authentifie** auprès de ce **fournisseur** (après rebond)

- Authentifié, il revient sur le **service** initial (autre rebond)

« OpenID » vu de l'utilisateur
variante avec attributs

- **L'utilisateur** veut accéder à un **service** sur internet
- Il découvre que ce service est accessible via des **comptes de services tiers** (google, facebook, twitter)
- Il choisit un de ces **fournisseurs d'identité**
  - pour se créer une identité

    ou
  - utiliser une de ses identités déjà existante
- Il **s'authentifie** auprès de ce **fournisseur** (après rebond) et **autorise** la diffuser d'informations le concernant
- Authentifié, il revient sur le **service** initial (autre rebond)
- Des **attributs** le concernant sont propagés vers le **service**

OpenIDConnect en résumé

OpenIDConnect est un
protocole de délégation
{ d'authentification et d'autorisation }

OpenIDConnect
résumé technique

- OpenIDConnect est une **couche d'identité**

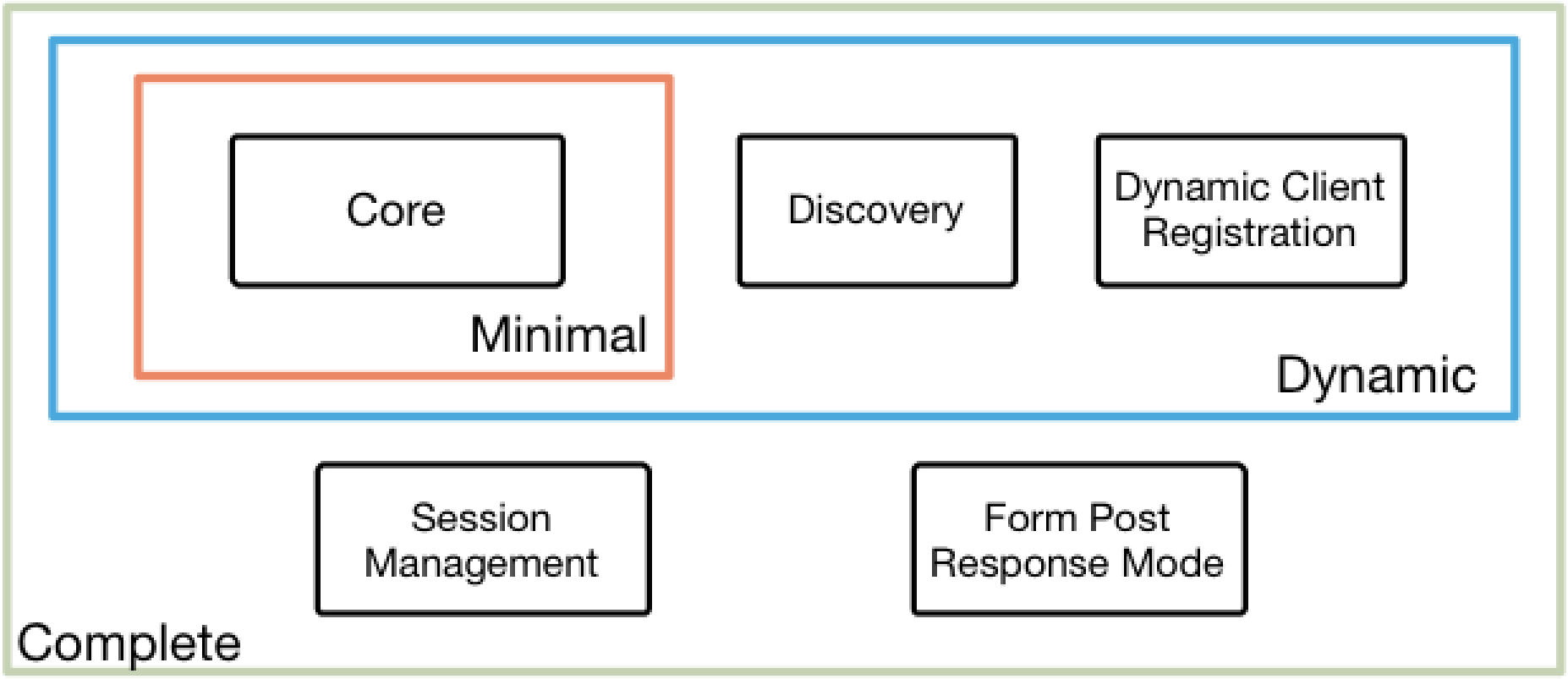
  au dessus du protocole **de délégation d'autorisation** OAuth 2.0

  permettant aux **clients** :

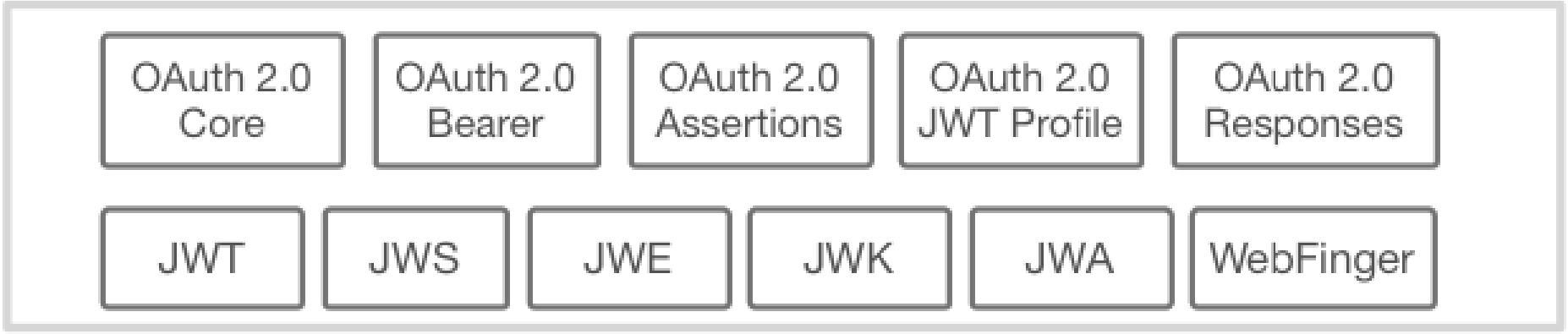
  - de vérifier l'identité de **l'usager**
  - d'obtenir de façon interopérable (avec REST) des informations de base (attributs) concernant le profile de **l'usager**

OpenID Connect Protocol Suite

http://openid.net/connect



Underpinnings



Historique / Versions

1) ~~OpenID [1.0] (authentification déléguée seule) Obsolète 2005~~
   - login = un URL, très simple
   - Trop simple : faiblesses de sécurité, limitations (pas d'attributs)
   - Précurseur, peu répandu
2) ~~OpenID 2.0 (authentification déléguée seule) Obsolète 2007~~
   - Plus complexe (XML), plus sûr, plus complet
   - Extension pour attributs : OpenID Attribute Exchange
3) **OpenIDConnect (authentification et autorisation) 2014**
   - Autorisation : OAuth 2.0 + TLS
   - Identité : plus simple : ~~XML~~ → JSON Web Token (JWT) / REST
   - Sûr, simple (pour les cas simples) et complet(attributs)

OAuth 2.0

Protocole d'Autorisation

Historique / Versions

- Historique (2006) OAuth 1.0 – RFC 5849
  - Complet (Authentification + Autorisation)
  - Simple
- Actuel (2012) : OAuth 2.0
  - OAuth 2.0 (Core) Authorization **Framework** - RFC 6749
  - Bearer Token Usage - RFC 6750
  - Threat Model and Security Considerations - RFC 6819
  - ...

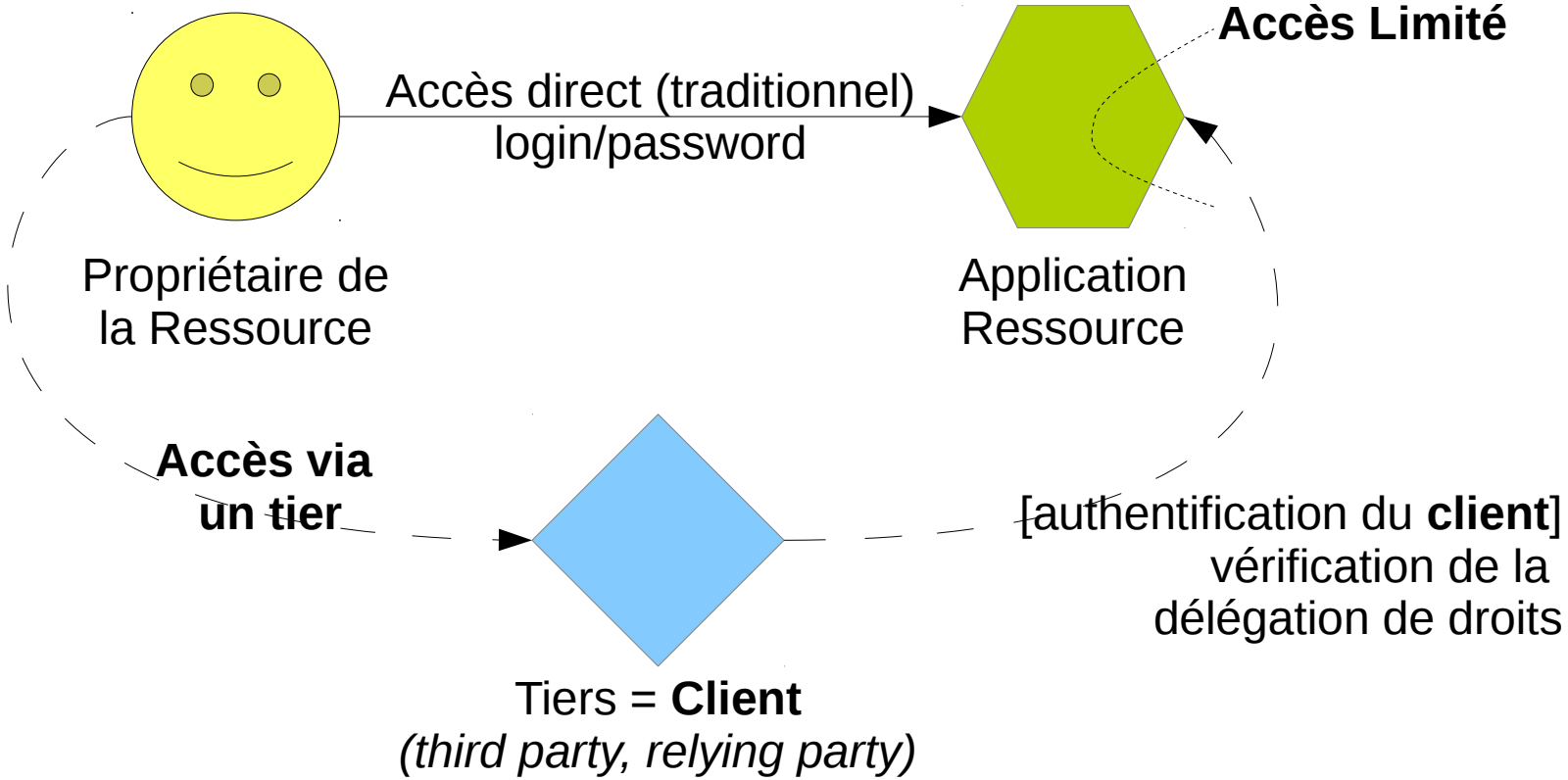
OAuth 1.0 et OAuth 2.0 incompatibles

OAuth 1.0 vs 2.0 (incompatibles)

- **passage à l'échelle**
  - 1.0 : inadapté pour répartition de charge (états centralisés)
  - 2.0 : séparation des rôles / indépendance => répartition sur fermes serveurs
- **support des équipements modernes**
  - 1.0 : supporte uniquement navigateur classique
  - 2.0 : supporte interfaces modernes (tablette, console de jeux, TV, ...)
- **complet → extensible**
  - 1.0 : autonome/complet mais simple & figé
  - 2.0 : framework extensible mais pas utilisable seul
- **protection des échanges simplifié**
  - 1.0 : signature cryptographique (hachage) de chaque requête – empêche de copier-coller les requêtes
  - 2.0 : transport chiffrée obligatoire (HTTPS)

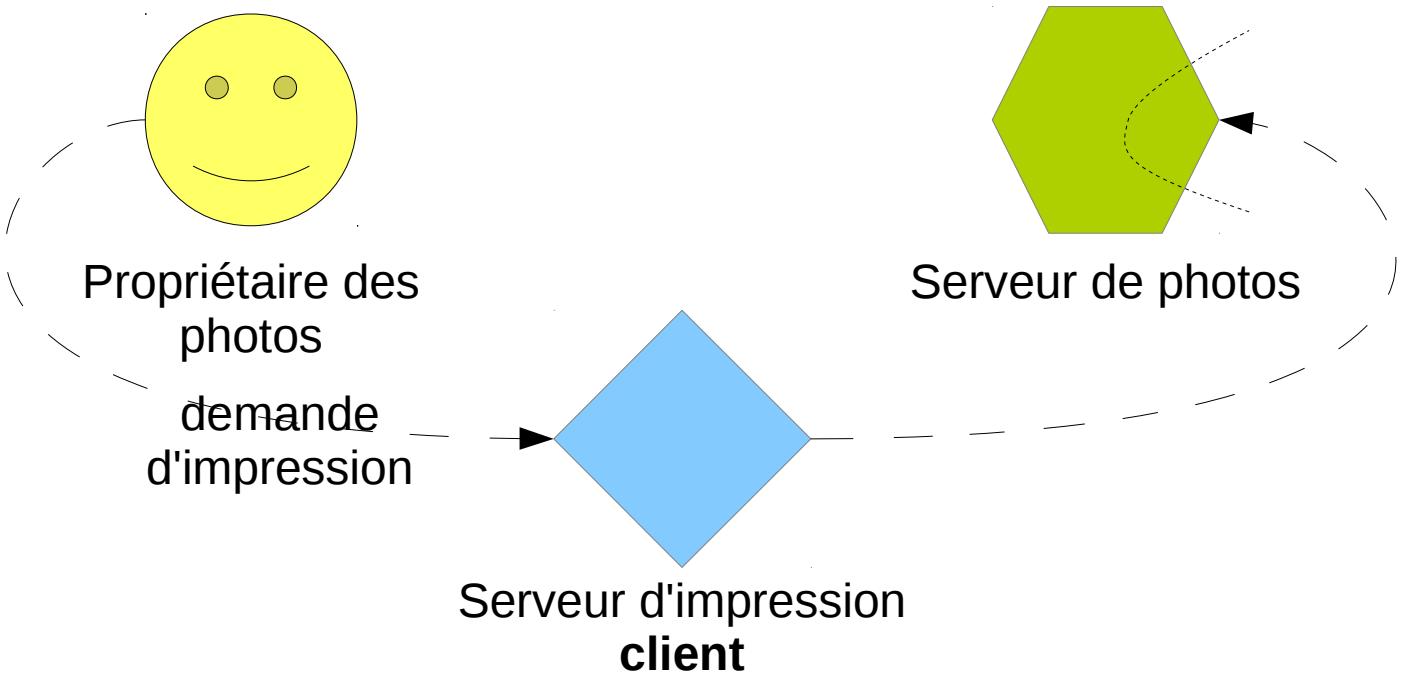
Protocole d'Autorisation
Objectif

Autoriser des accès pour des **tiers** (**clients**), à certaines **ressources**, au nom du **propriétaire de ces ressources,** mais sans donner tous ses droits



Le besoin
Ex : impression de photos

- Entités :
  - le propriétaire des photos
  - le serveur hébergeant les photos
  - le serveur d'impression
- Objectif :
  - Le propriétaire veut que le serveur d'impression puisse récupérer les photos, en son nom, pour les imprimer.
- Contrainte :
  - Il ne veut pas donner tous ses droits (mot de passe/chèque blanc)



Protocole d'Autorisation
Étapes (première approche)

- [découverte par le **client** des **points d'entrée** « web-services »]
- [enregistrement du **client** auprès d'un service central]

*le **propriétaire** souhaite que le **client** accède à **ses ressources***

- le **client** demande une **autorisation** d'accès à **des ressources**

*le **propriétaire** est sollicité et donne (ou pas) son accord*

- le **client** utilise son **autorisation** et récupère un **jeton d'accès** à **une ressource** particulière
- le **client** utilise ce **jeton d'accès** pour accéder à cette **ressource**

*chaque étape correspond à un **point d'entrée***

***autorisation** et **jeton d'accès** sont temporaires, révocables & ciblées*

on passe d'une étape à une autre via des redirections HTTP

redirections HTTP sur des **points d'entrée** (webservices)

- utilisation des mécanismes classiques pour l'authentification déléguée en mode web (CAS, Shibboleth) : **redirection** HTTP du User-Agent du propriétaire des ressources (usager)
- permet de faire le lien entre les différents **point d'entrée** (webservices), propager des données (**attributs, jetons, ...**) d'un serveur à l'autre

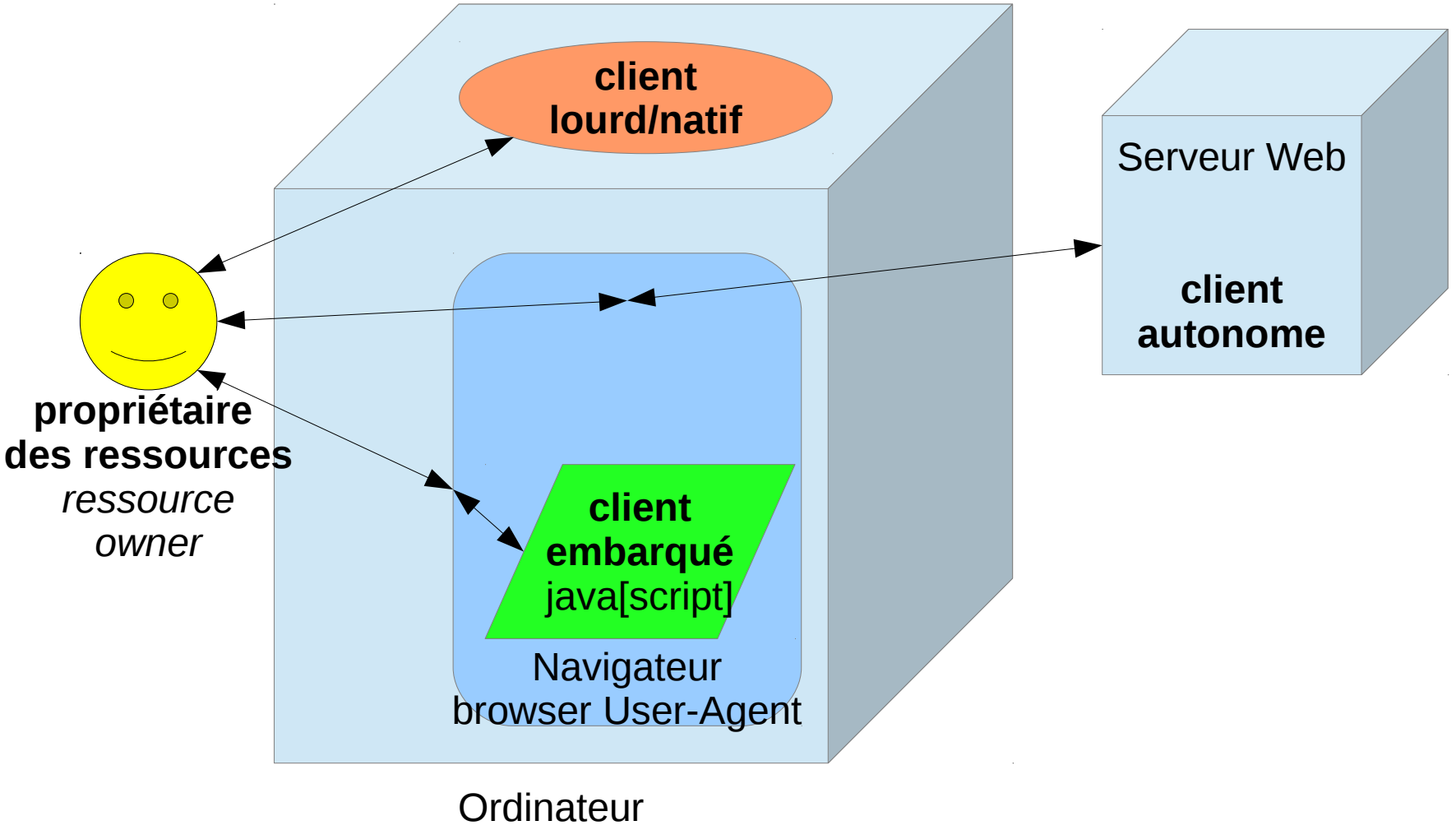
Terminologie (*traductions*) OAuth/OpenIDConnect (1/2)

- *Ressource Owner*
  - *Propriétaire des ressources, Usager Final, Humain ?*
- *Credentials :*
  - « Justificatifs d'identité » : « login/password »
- *Clients = Relaying Party (RP) = Consumer*
  - Clients = « Tiers Mandataire » : agit au nom des propriétaire des ressources, consommateur d'autorisations
- *OpenID Provider = OP*
  - Fournisseur d'identité OpenIDConnect,
- *Authorization Server*
  - Serveur d'Autorisation

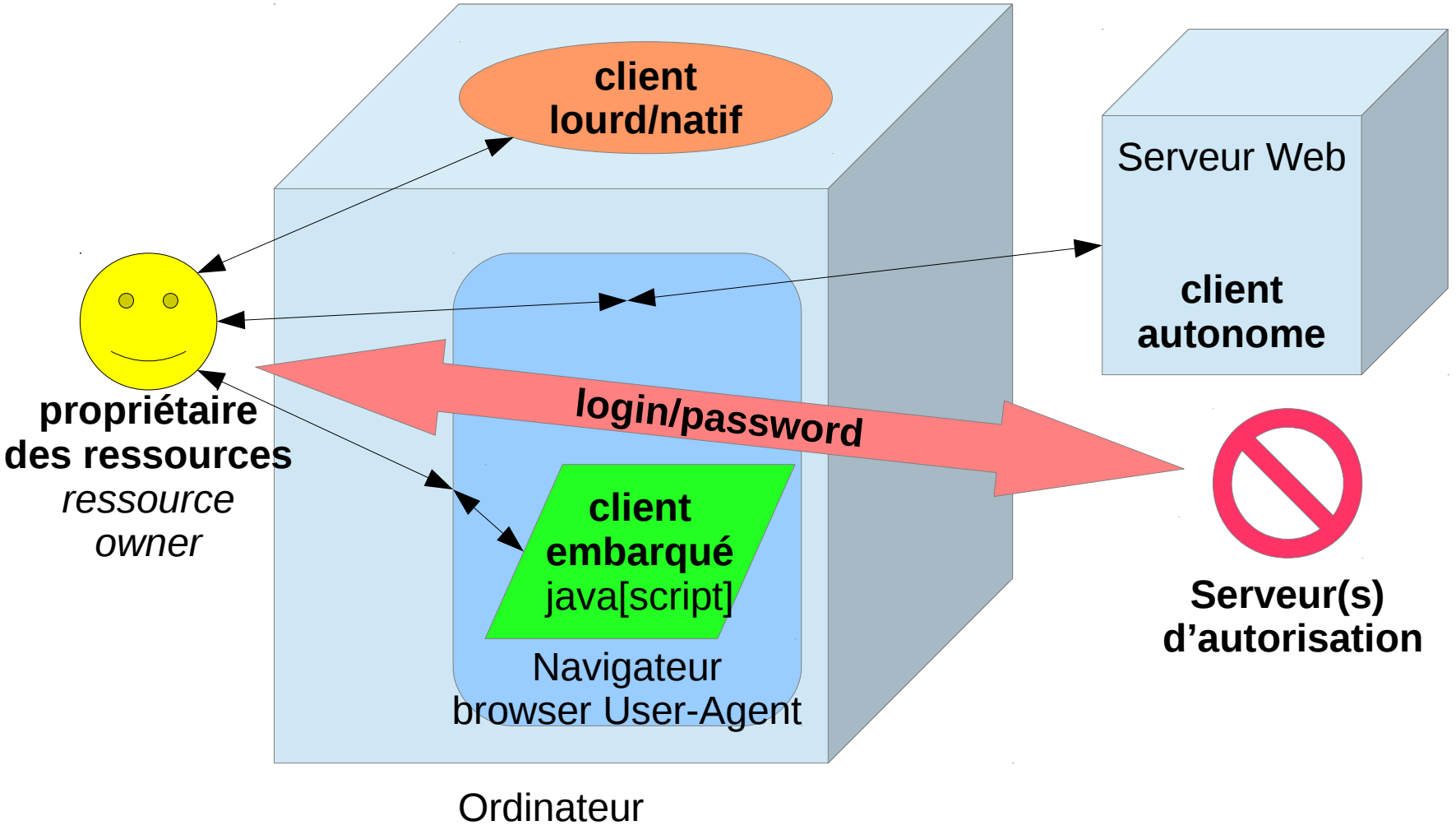
Terminologie (*traductions*) OAuth/OpenIDConnect (2/2)

- *Acces Token* :
  - Jeton d'Accès
- *Authorization Code*
  - Code d'autorisation
- *Authorization Grant* :
  - autorisation accordée, allocation d'autorisation
- *Claims*
  - Affirmations, prétentions, revendication, **assertions**
- *Endpoint*
  - Point d'entrée, Guichet de service, URL de web-service

Qu'est ce qu'un **client** ?



Trajet des **justificatifs d'identit**
« *credentials* » (login/password)



Par défaut, le client est tenu à l'écart

Schéma Général

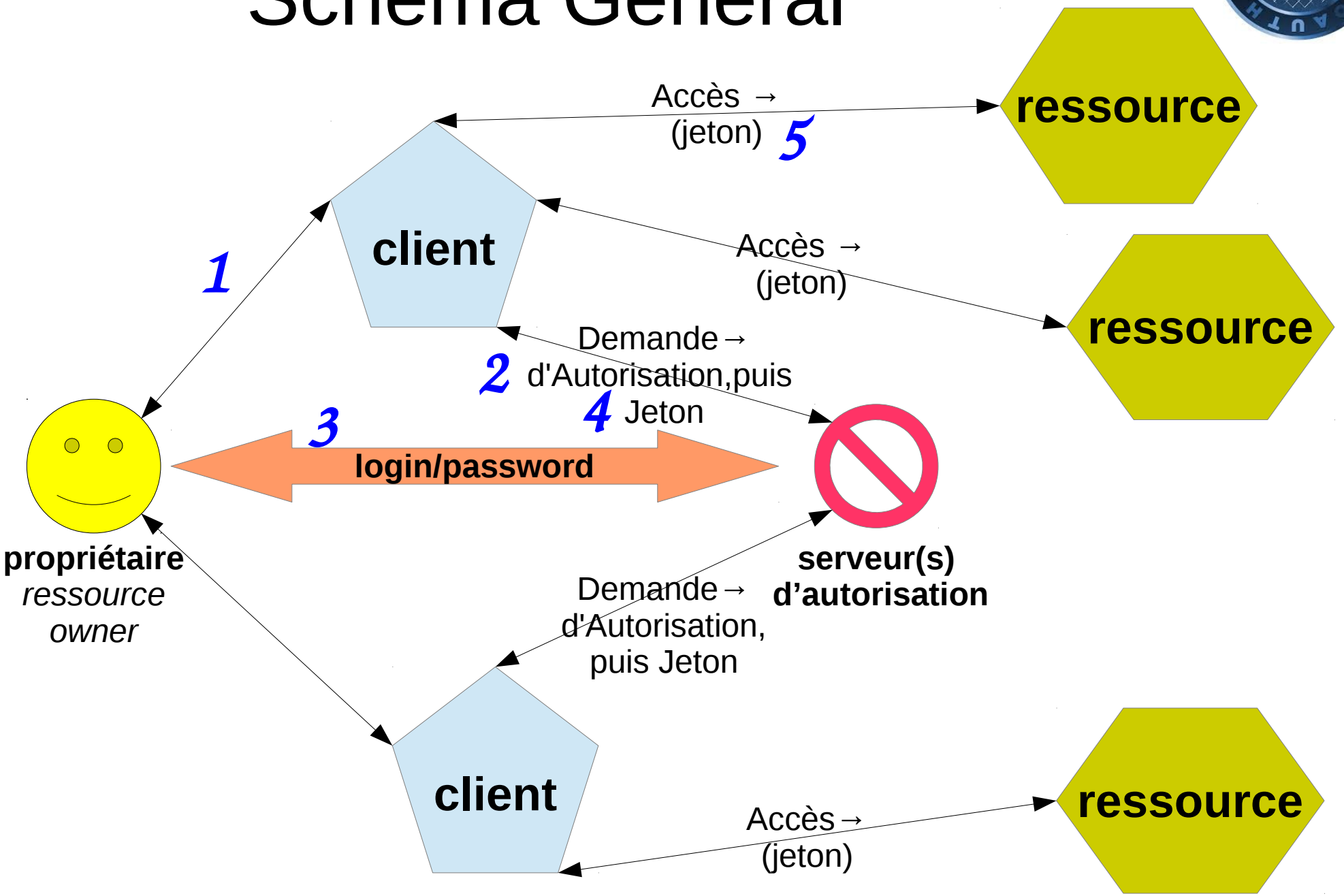
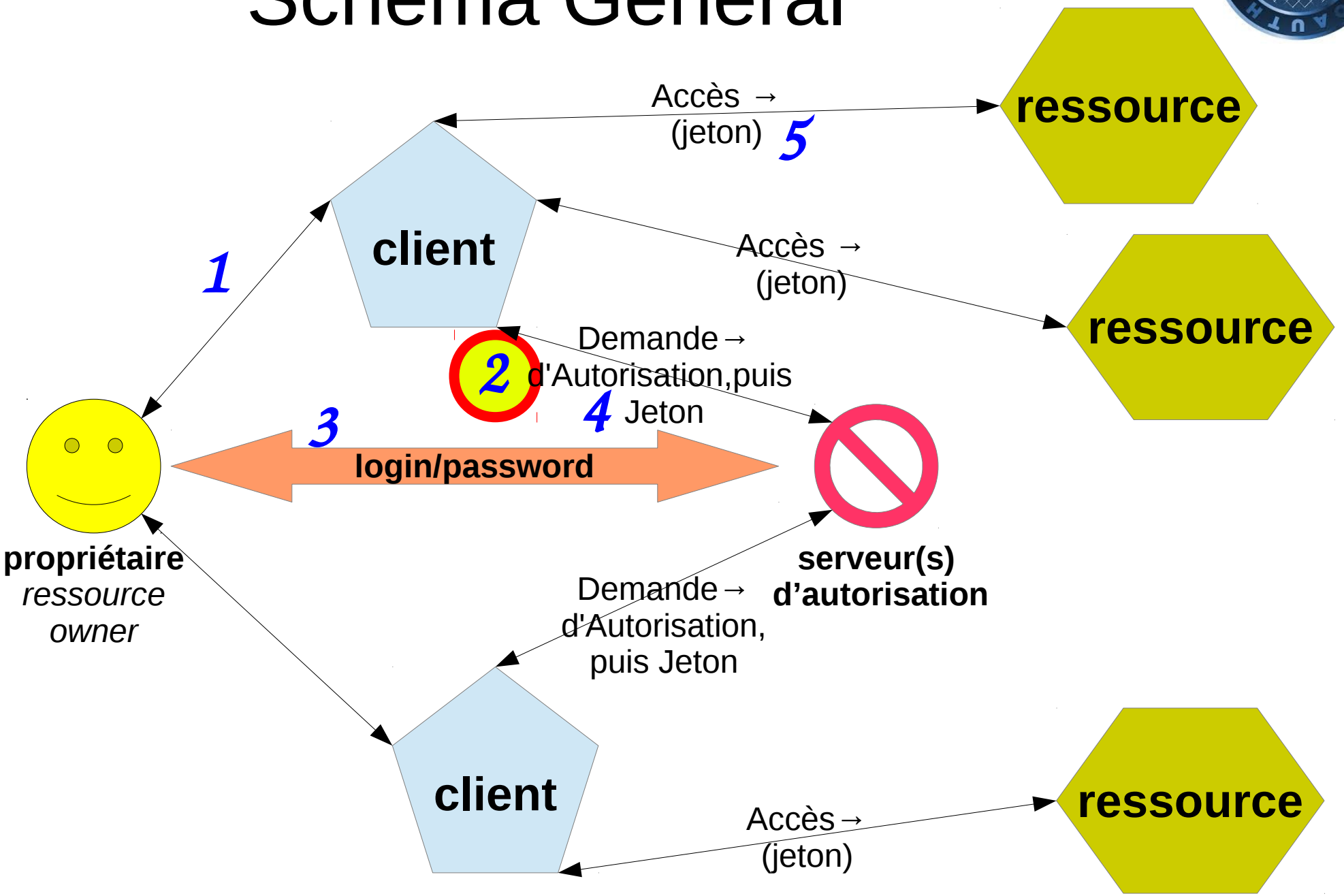


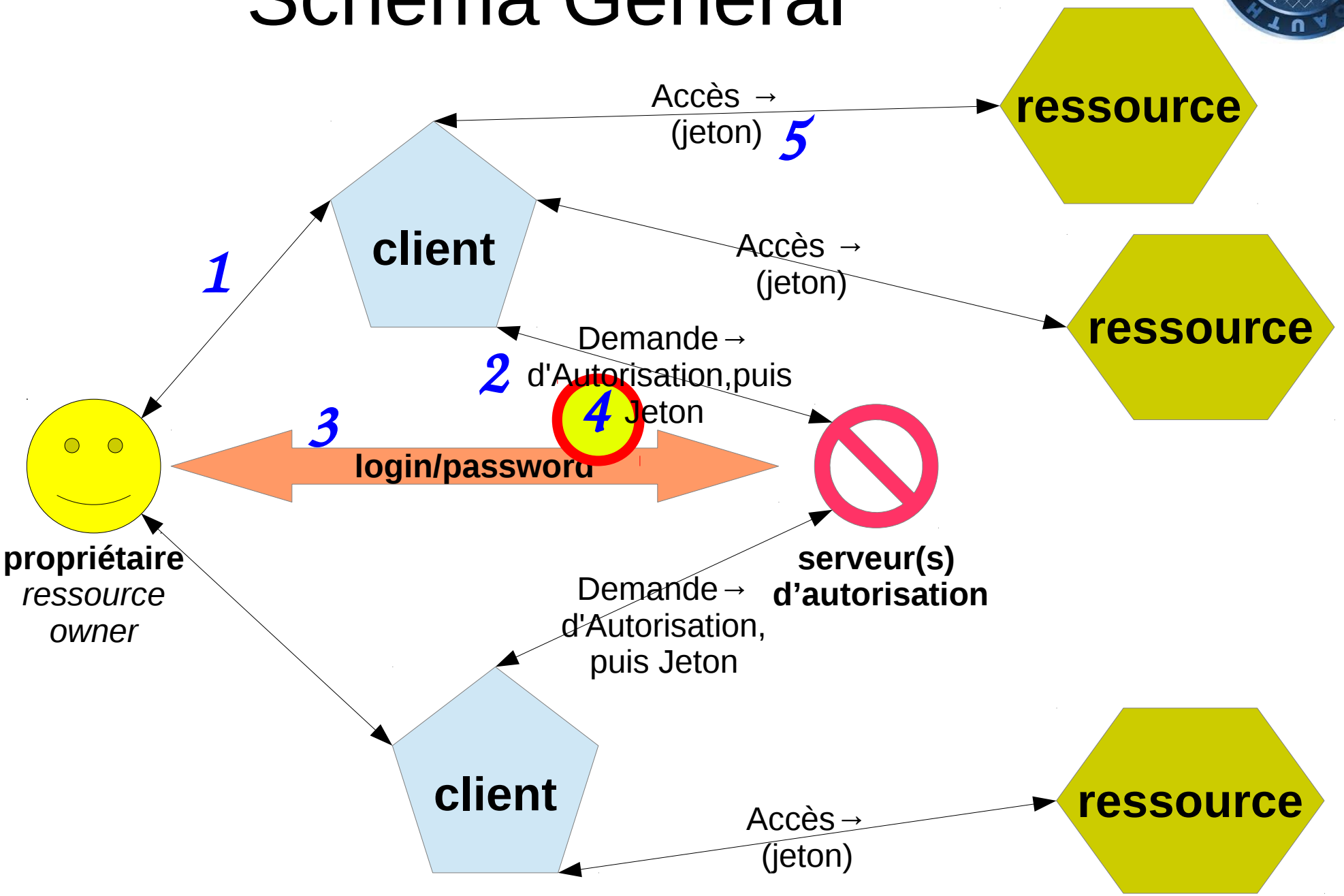
Schéma Général



Étape 2
Code d'Autorisation

- « chaîne de caractère opaque » représentant l'accord d'un **propriétaire de ressource** pour autoriser l'accès à *certaines* (*scope*) de ses ressources à un **client**
- délivré au **client** par un **serveur d'autorisation** ayant reçu **l'accord** du **propriétaire**
- le **code d'autorisation** est ensuite soumis par le **client** a un **serveur d'autorisation** pour obtenir des **jetons d'accès** aux **ressources**
- durée de vie courte (max 10 minutes)

Schéma Général



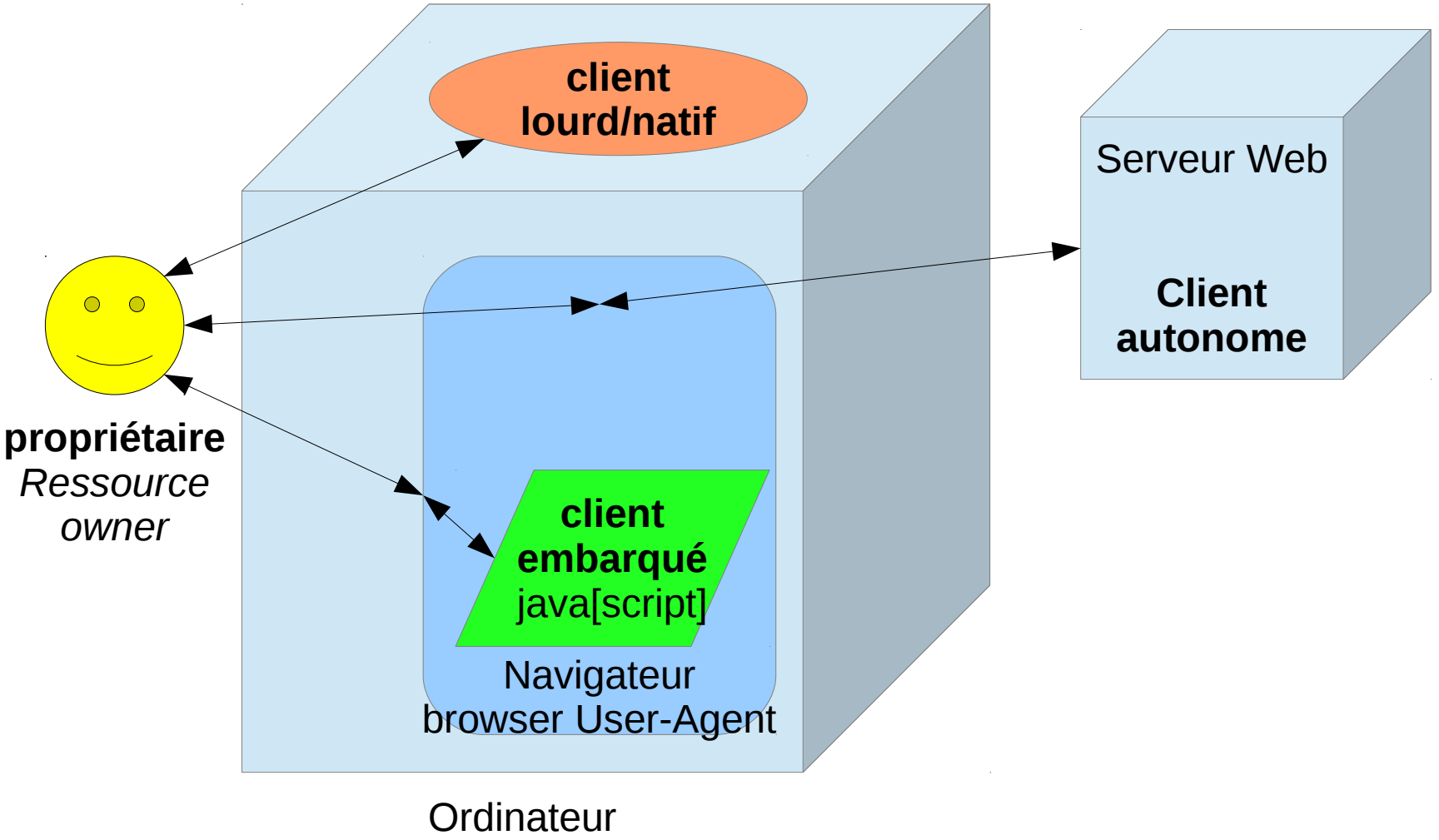
Etape 4
Jeton d'Accès

- « chaîne de caractère opaque » représentant une *autorisation d'accès* pour un **client** à une **ressource** (*scope*) appartenant à un **propriétaire**
- délivré au **client** par un **serveur d'autorisation** en échange d'un **code d'autorisation**
- le jetons d'accès est ensuite soumis par le **client** à la **ressource** pour avoir accès
- le **jetons d'accès** est *révocable*, *temporaire*, avec un *périmètre limité* <> mot passe du propriétaire ou « chèque en blanc »

[Jeton de Renouvellement] *(Refresh Token)*

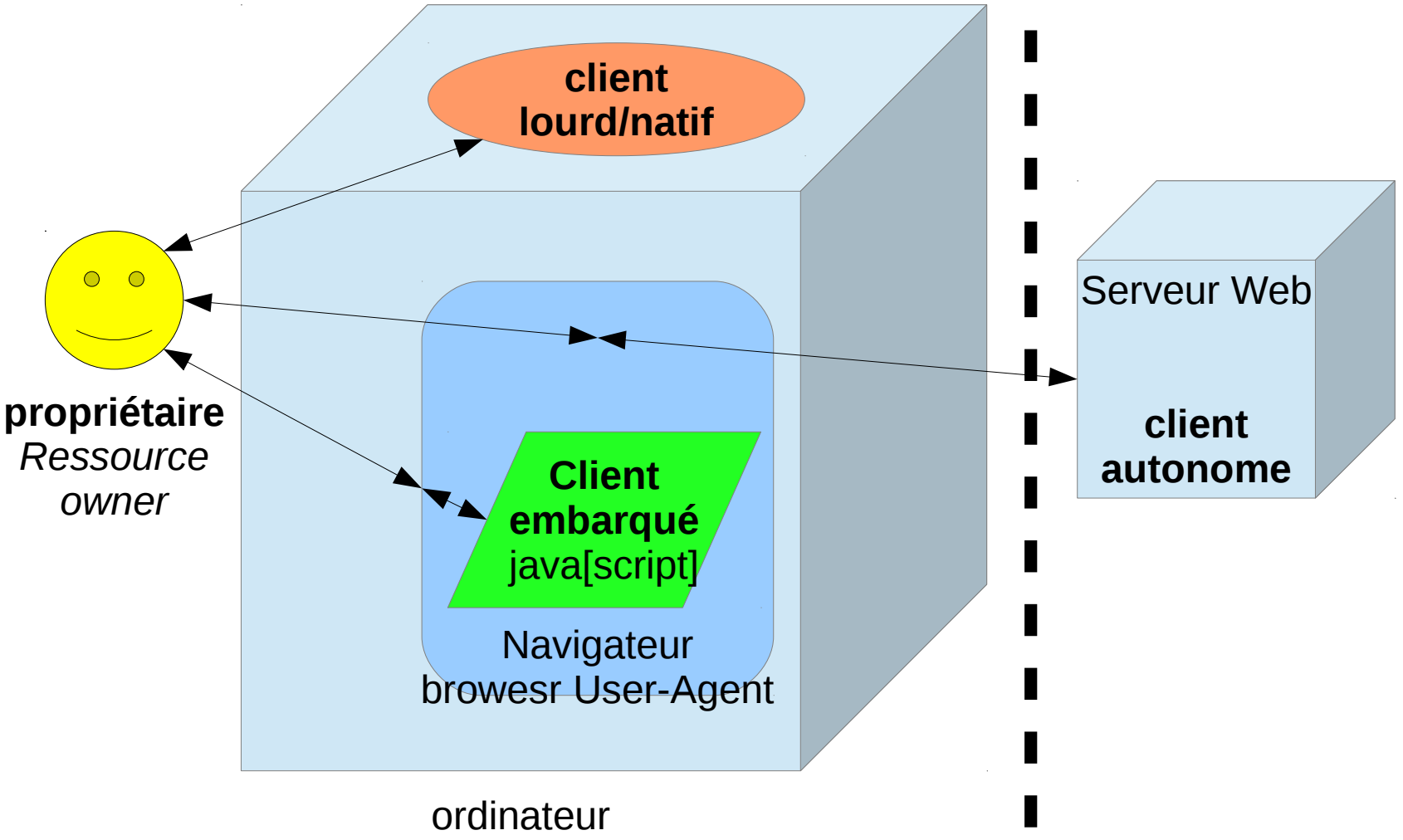
- les **jetons de renouvellement** permettent d'obtenir, des **serveurs d'autorisation**, de nouveaux **jetons d'accès** valides :
  - quand un jeton d'accès à expiré ou à été invalidé
  - pour obtenir des jetons d'accès supplémentaires
  - pour obtenir des jetons d'accès avec un périmètre :
    - identique
    - plus retreint
- les **serveurs d'autorisation** *peuvent* émettre ce type de jetons (comportement optionnel)
  - les **jetons de renouvellement** sont fournis par les **serveurs d'autorisation** en plus (bonus) en réponse à une requête de **jetons d'accès**

Retour sur les différents types de clients



=> différents types d'autorisations

Clients «accessibles» «inaccessibles» & autonome



Contrôle par le propriétaire ou pas

4 types d'autorisation (*Authorization Grant*) 1/2

(dialogue : client ↔ serveur autorisation)

- (4.1) via **code d'autorisation** (cas courant)
  - **Code d'autorisation** → **Accord propriétaire** → **Jeton d'Accès**
- (4.2) Implicite
  - **Accord propriétaire** → **Jeton d'Accès**
  - Simplification dans le cas d'un **client** embarqué dans le navigateur (« java[script] ») = **client** qui n'a pas d'existence propre (non autonome)
  - le **client** récupère directement un **jeton d'accès** (au lieu d'un **code d'autorisation** lui permettant d'avoir ensuite un **jeton d'accès**)

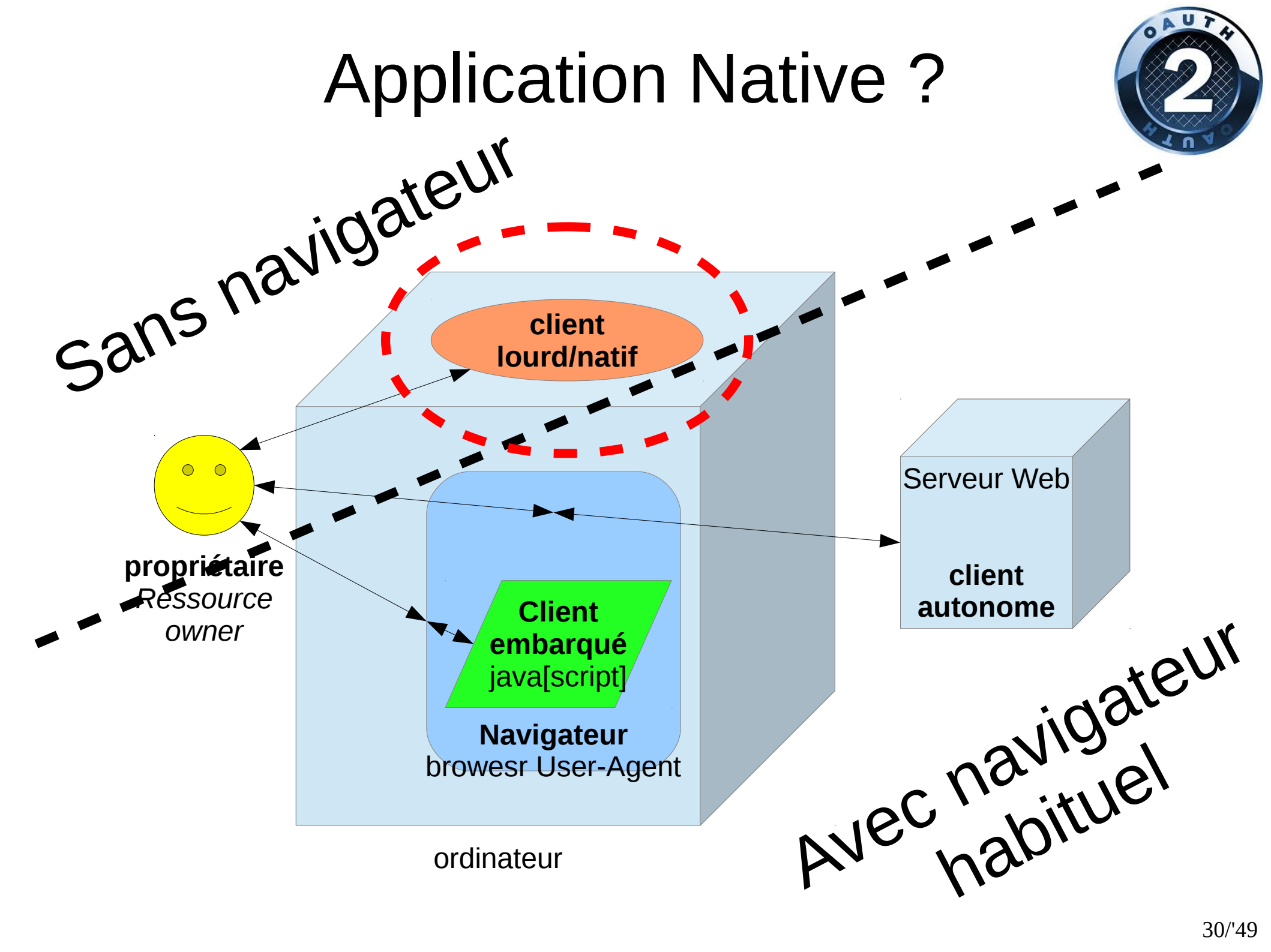
4 types d'autorisation (*Authorization Grant*) 2/2

(dialogue : client ↔ serveur autorisation)

- (4.3) via les identifiants du **propriétaire des ressources**
  - **Justificatifs Identité Usager (mot de passe)** → **Jeton d'Accès**
  - ! Nécessite une grande confiance dans le client (local)
- (4.4) via les Identifiants du **client**
  - **Justificatifs d'Identité client (mot de passe)** → **Jeton d'Accès**
  - Cas du client autonome qui agit pour son propre compte en tant que « propriétaire de ressources »

Cas de l'application native
quel User-Agent utiliser ?

Application Native ?



Cas : application native quel User-Agent ?

- (9) **Application Native** est un **client** s'exécutant sur l'équipement du propriétaire des ressources.
- pour récupérer les **autorisations**, un dialogue « web » est nécessaire entre le **point d'entrée** du **serveur d'autorisation**, ce **client natif** et le **User-Agent (U-A)** du **propriétaire des ressources**. Le **U-A** peut être :
  - **U-A externe** (navigateur) : + : session déjà ouverte, SSO facilités (mémo mdp), *auth-devices*
    - U-A extension (plugin)
    - Redirection sur URI contrôlé par le client, local à l'équipement
    - Web serveur local ou distant (sous contrôle du client)
    - Copier coller manuel
  - **U-A embarqué** dans l'application - : saisie mdp
    - Accès à l'état interne de l'U-A
    - Accès au stockage des cookies

OAuth 2.0
un protocole extensible

- Nouveaux types de jetons d'accès
  - nouveaux types enregistrés dans « le registre »

    ou
  - sous la forme d'un URI absolu et unique
- Nouveaux types d'autorisations
  - nouveaux paramètres enregistrés dans « le registre »
- Nouveaux types de réponses, d'erreur, ...
  - Nouveaux types enregistrés dans « le registre »

OAuth 2.0, seulement un « *Authorization Framework* »

- Pas de mécanisme standard pour obtenir des information sur l'identité de l'utilisateur final !!!
- *They define mechanisms to obtain and use Access Tokens to access resources* ***but do not define standard methods to provide identity information****. Notably, without profiling OAuth 2.0, it is incapable of providing information about the authentication of an End-User. (OpenIDConnect)*

OpenIDConnect

OpenIDConnect + ID Token (Assertion JWT)

- OpenIDConnect définit l'extension « **openid** » pour accéder au profile du **propriétaire** (utilisateur final)
- Le **client** doit requérir le périmètre (*scope*) « **openid** » pour en bénéficier
- Les informations de profile sont retournées dans un **ID Token** : des assertions (*claims*) au format **JWT** (JSON Web Token)

Notion d'Assertion

- Informations organisées sous forme de paires nom/valeur concernant un **sujet** (*subject*) et certifiées par un **émetteur** (*issuer*)
- « Attestation » : ensemble d'informations sur un **sujet** attestées par une **autorité** avec son sceau cryptographique
- Ex :
  - **SAML** Security Assertion Markup Language (2005)
  - **JWT** JSON Web Token (draft 2014)

Le format JWT *[jot]*
JSON Web token (*draft*)

- Format compact pour représenter et échanger des **assertions** (*claims*)
- Les informations de l'assertion sont rangées dans une structure JSON qui est ensuite :
  - Soit chiffrée dans une structure JWE - JSON Web Encryption
  - Soit signée avec une JWS - JSON Web Signature
- Le résultat est ensuite systématiquement compacté et sérialisé sous sa forme finale :
  - JWE → JWE Compact Serialization
  - JWS → JWS Compact Serialization
- Forme finale conçue pour passer sans problème dans les URL (compact et jeu de caractère adapté)

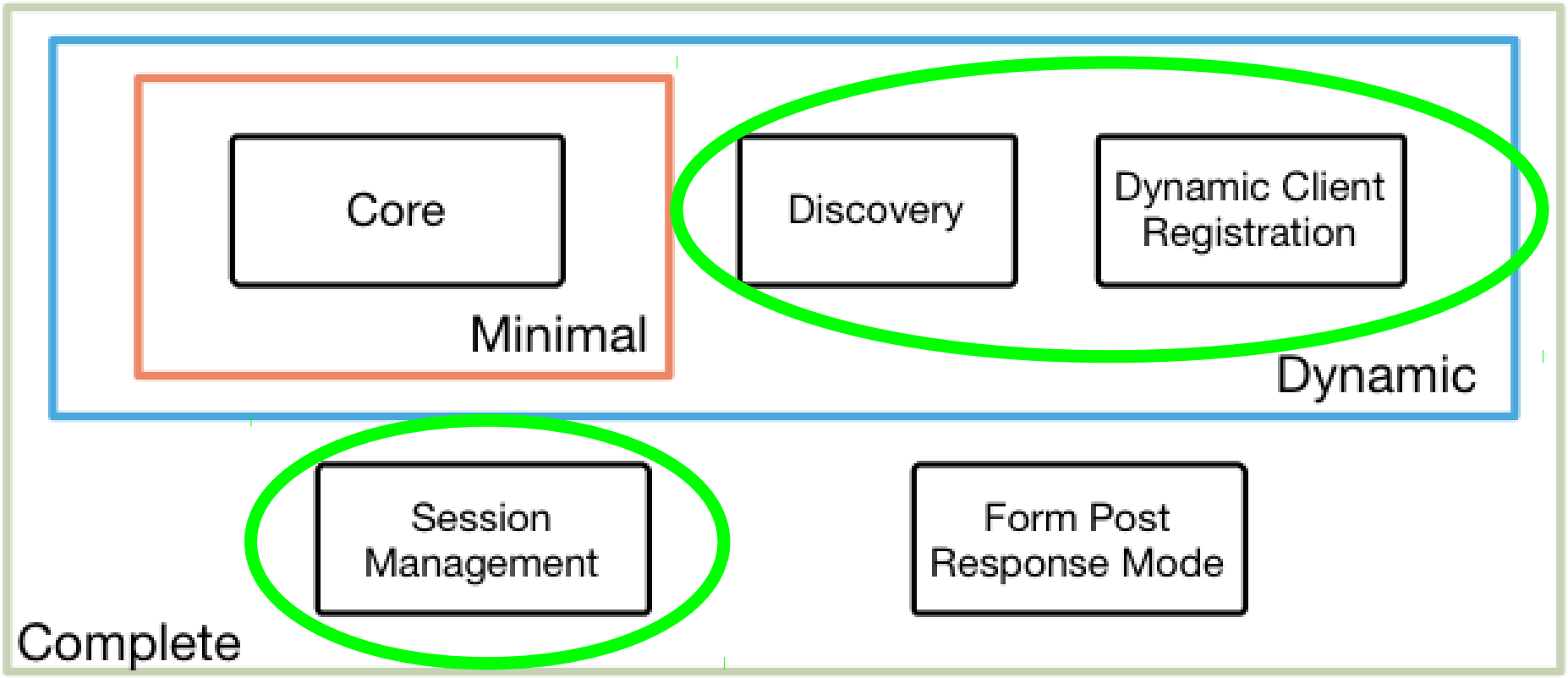
Contenu de base d'un ID Token

- **iss** : issuer Claim : URL identifiant l'émetteur (**autorité**)
- **sub** : subject Claim : identifie de façon unique le sujet (Usager-Final) dans le référentiel émetteur « sub@iss »
- **aud** : audience Claim : liste des *identifiants* des destinataires
- **exp** : expiration time Claim – date d'expiration (*)
- **iat**: issued at Claim – date d'émission (*)
- **auth_time** : instant UTC ou l'usager-final s'est authentifié (*)
- **nonce** : chaîne aléatoire limitant les possibilités de rejeux
- [**acr**] : Authentication Contexte Class Reference - « qualité »
- [**amr**] : Authentication Methode Reference – passwd, otp, ...
- [**azp**] : Authorized party – le destinataire effectif

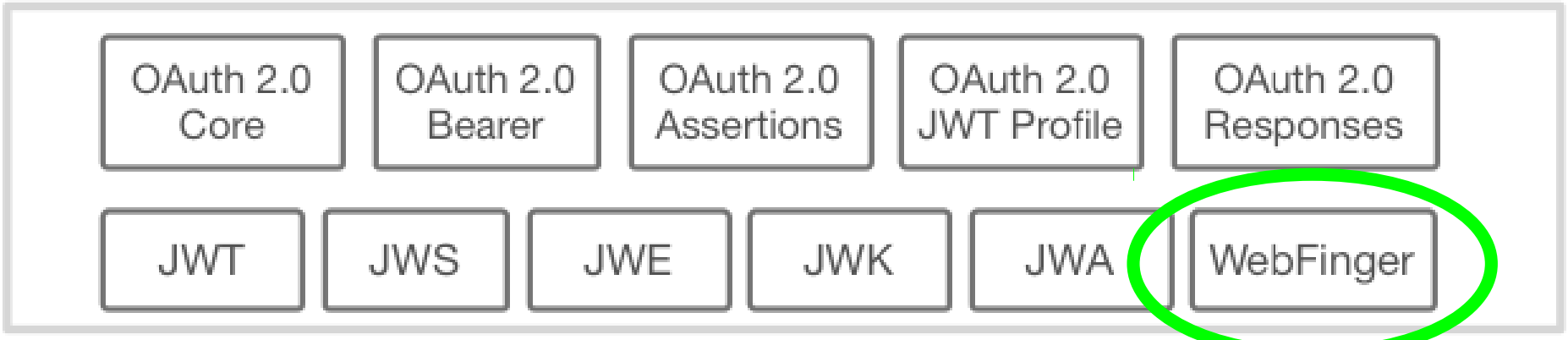
(*) en secondes depuis 1/1/70 UTC

OpenID Connect Protocol Suite

http://openid.net/connect



Underpinnings



Prise en compte d'un OP

OpendIDConnect Provider

- Métadonnées d'un OP utilisées par les **clients** et **ressources** :
  - Ses points d'entrée (URL web service)
  - Ses clefs de chiffrement/signature
  - Ses périmètres (*scopes*)
  - ...
- Intégration des métadonnées de l'OP dans le **client**
  - Publié : Saisie manuelle
  - Autoconfiguration : URL du ***Discovery Service***
- Enregistrement du **client** (l/p) auprès de l'OP
  - Manuel : interface web, copier-coller
  - Automatique : Protocole ***Dynamic Client Registration***

Webfinger : découverte automatique de l'OP d'un user

OpendIDConnect Provider

- L'utilisateur indique son email :

  login@domain.top

- Interrogation du serveur web ***domain.top***

  GET /.well-known/webfinger?....*login*...
  Host: domain.top

- Récupération du *Discovery Service Endpoint* de l'OP de l'utilisateur
- Puis dialogue avec le *Dynamic Client Registration Endpoint*

Configuration totalement automatisable
comme en OpenID 1.0 !

Les points d'entrées
Endoints / web services

- Enregistrement client *(Registration Endpoint)*
  - Extension *Dynamic Client Registration*
- Code d'Autorisations *(Authorization Endpoint)*
- Jetons d'accès *(Token Endpoint)*
- Information utilisateur *(UserInfo Endpoint)*
  - Information sur l'utilisateur authentifié
- Fin de session *(End Session Endpoint)*
  - Extension *Connect Session Management*
  - *Interrogation de l'état de la session (connecté ou pas)*
  - *Gestion du logout distribué*

Récapitulatif 1/2

1) l'**utilisateur/propriétaire** donne au **client** l'email associé à son identité

2) le **client** récupère les métadonnées de l'OP à l'emplacement par défaut

3) le **client** s'enregistre auprès du *Point d'Enregistrement*

4) le **client** prépare sa demande de *Code d'Autorisation*

5) le **client** envoie sans demande de *Code d'Autorisation* au *Point d'entrée d'Autorisation*

6) le *Point d'entrée d'Autorisation* amorce un dialogue avec le **propriétaire**

Récapitulatif 2/2

6) le *Point d'entrée d'Autorisation* amorce un dialogue avec le **propriétaire**

- Il récupère le consentement du **propriétaire**
- Il renvoi le propriétaire vers le **client** avec un *Code d'Autorisation*

2) le **client** utilise son *Code d'Autorisation* pour demande un *Jeton d'Accès* au *Point d'Entrée jetons*

3) le **client** reçoit en réponse un *Jeton d'Identification* (ID Token) et un *Jeton d'Accès*

4) le **client**

- valide le *Jeton d'Identité* et récupère l' Identifiant de l'Usager Final *(End-User's Subject Identifier)*
- utilise son *Jeton d'Accès* pour accéder à la ressource

Auth. Par Code d'Autorisation Requête d'Authentification 1/3

- Paramètres OAuth 2.0
  - *scope* : périmètre des ressources ciblées **(extension openid)**
  - *response_type* : 'code' (type de réponse attendue)
  - *client_id* : ID du client auprès du serveur d'autorisation (compte)
  - *redirect_uri* : URI où rediriger le User-Agent en réponse à cette demande
  - {*state*} : état propre au client
  - [*response_mode*] : mode pour transmettre la réponse
  - [*nonce*] : limite les possibilités de rejeux

Auth. Par Code d'Autorisation Requête d’Authentification 2/3

- Paramètres *extensions OpenIDConnect (interaction user)*
  - *[display] : (**page**/popup/touch/wap)*
    - Définit le mode d'interaction avec l'Usager-Final
  - *[prompt] : (none, login,consent,select_account)*
    - Modalité de l'interaction : **None** : test si Usager-Final déjà authentifié **Login** : force réauthentification, **consent** : demande consentement, **select_account** : Usager-Final ayant plusieurs comptes)
  - *[max_age] :* (secondes) durée de vie de l'authentification courante de l'Usager-Final. Si dépassée, force une réauthentification.
  - *[ui_locales] :* choix de la langue
  - *[id_token_hint] :* Jeton d'Identification précédemment renvoyé par le Serveur d’Autorisation (en particulier avec prompt=none) – indices, val/défaut
  - *[login_hint] :* indice concernant le login de l'Usager-Final, val/défaut.
  - *[acr_values] :* Authentication Context Class Référence - contexte

Auth. Par Code d'Autorisation Requête d'Authentification 3/3

- *Paramètres d'autres extensions OpenIDConnect*
  - *client_secret... :*
  - *[claims] : (cf claims_parameter_supported* Discovery*)*
    - *[userinfo] :* récupération d'attributs (ex : given_name, nickname, email, email_verified, picture, …)
    - *[id_token] : autres assertions auth_time, acr*
  - *[claims_locales] :* langue pour les assertions
  - *[request] :* permet d'activer/gérer signature/chiffrement
  - *[request_uri] :* permet le passage de paramètres par référence

Références Utilisées

- OAuth
  - http://hueniverse.com/2007/10/04/beginners-guide-to-oauth-part-i-overview/
  - RFC 6749 : The OAuth 2.0 Authorization Framework
  - RFC 5849 : The OAuth 1.0 Protocol
  - http://www.bortzmeyer.org/6749.html
  - http://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified
  - http://oauth.net/documentation/
- OpenID Connect
  - http://openid.net/specs/openid-connect-core-1_0.html
  - http://fr.slideshare.net/andrea.chiodoni/openid-and-oauth
- ~~OpenID (Obsolète)~~
  - ~~V1 http://openid.net/specs/openid-authentication-1_1.html~~
  - ~~V2 http://openid.net/specs/openid-authentication-2_0.html~~
  - ~~V2 http://openid.net/specs/openid-attribute-exchange-1_0.html~~

Fin

Vos questions ?