Web Services Security

Université Libre de Bruxelles

Mohamed Amine YOUSSEF
Nicolas VANDER AUWERA

26 April 2012

Structure

- Part 1: WS-Security
  - Introduction
  - Challenges..
  - How does it work?
  - Error Handling
  - Exstensions

Structure

- Part 2: REST-based security
  - OpenID
  - OAUTH 1.0
  - OAUTH 2.0
  - Amazon S3

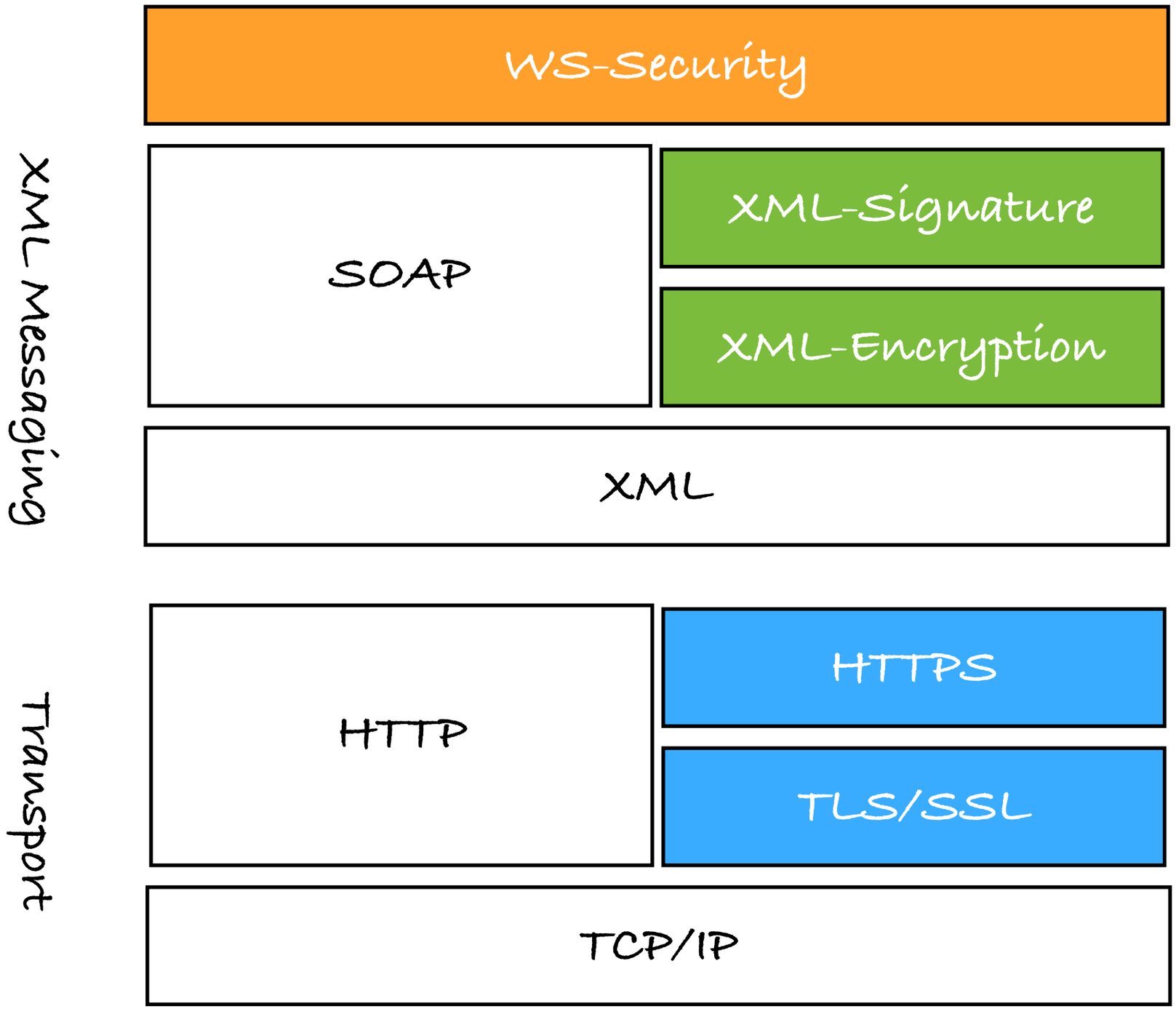
Introduction

Our story so far...

- Web service security requirements:

1. Confidentiality
2. Integrity
3. Authorization
4. Authentication
5. non-repudiation

Our story so far...



What WS-Security is?

WS-Security

«WS-Security is a message level standard definding how to secure SOAP messages».

IBM.

WS-Security

- IBM, Microsoft & verisign.
- In 2002, Specification for WS-Security.
- In 2004, OASIS released WS-Security 1.0.
- In 2006, WS-Security 1.1 by OASIS.

Challenges...

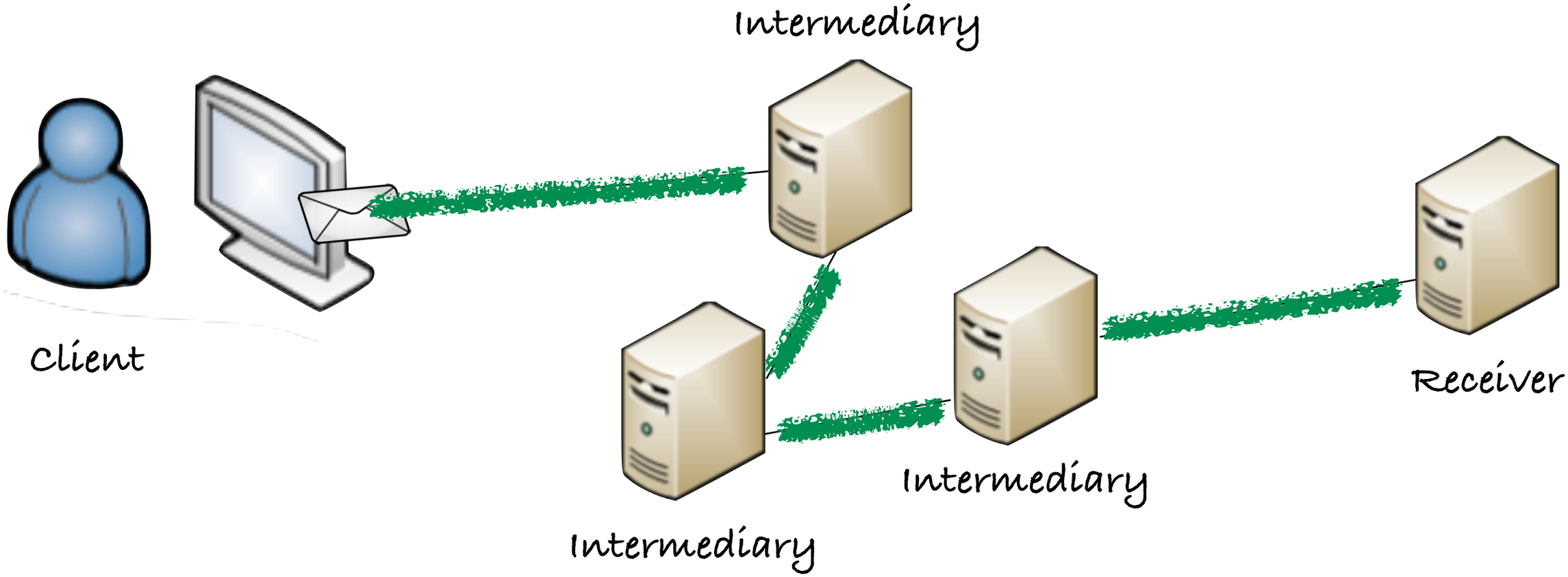
Challenges - 1

Privacy	Solutions	Encryption
Integrity	→	Signatures
Authentication		Security tokens

Challenges - 2

Why not just use HTTPS for transmitting SOAP messages?

Challenges - 2



THE GOAL:

support distributed message processing

THE PROBLEM:

Point to point communication

THE SOLUTION:

WS-Security

Challenges - 2

Message Security	Transport security
Immature standards only partially supported by existing tools	Widely available (SSL, TLS, HTTPS)
Securing XML is complicated	Understood by most system administrators
Different parts of a message can be secured in different ways.	Point 2 Point: The complete message is in clear after each hop
Asymmetric: different security mechanisms can be applied to request and response	Symmetric: Request and response messages must use same security properties
Self-protecting messages	Transport specific

Challenges - 3

Interoperability, how do we manage it?

Challenges - 3

Kerberos

WS-Security extensions

?

SSL

WS-Security

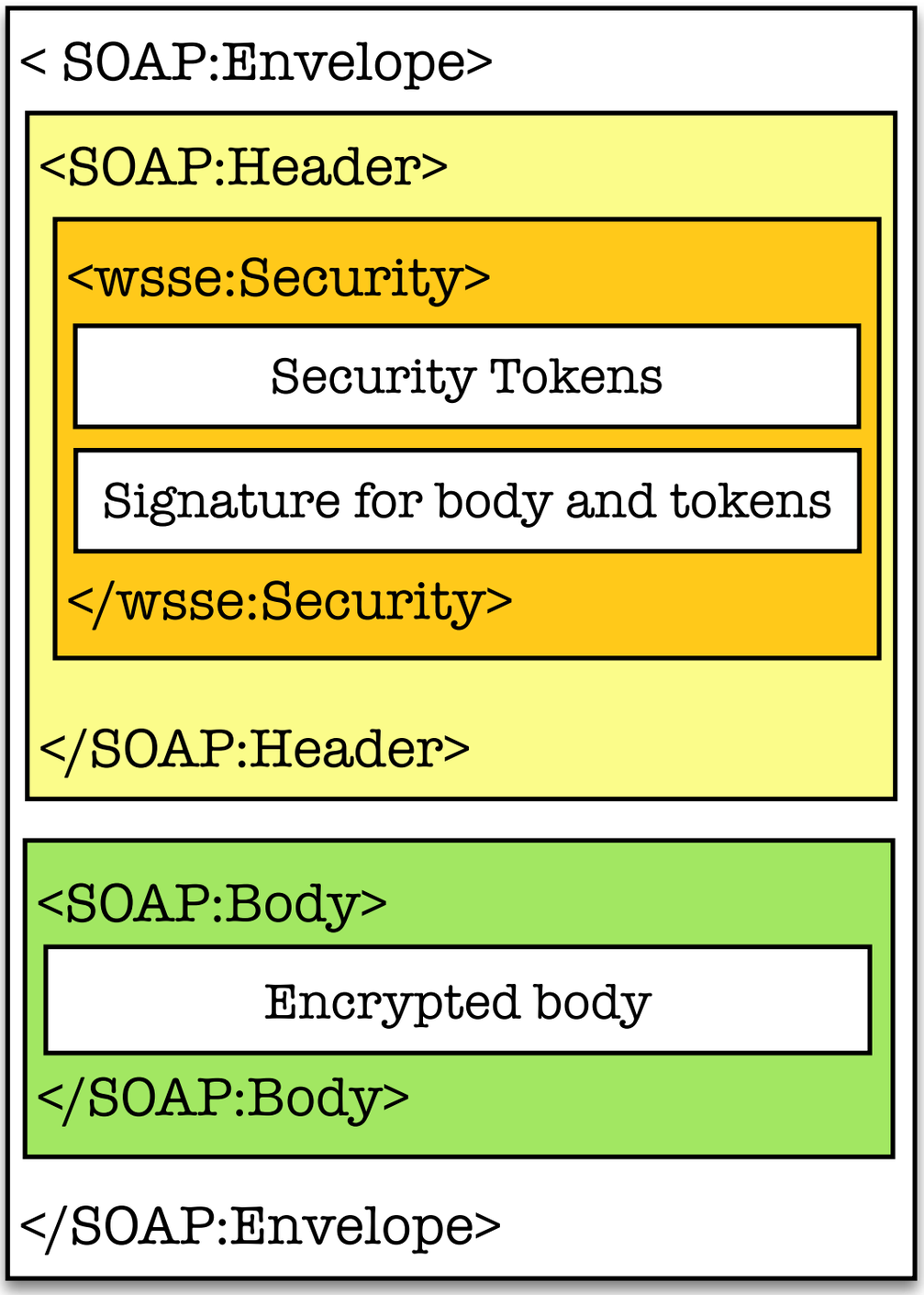
- End2end secure communication.
- A security application-based approach.
- Meet company policy.
- Interoperability.
- Flexibility & efficiency.

WS-Security

- Exchanging security tokens as part of message
- Ensure the integrity and confidentiality of the contents of SOAP messages

How does it works?

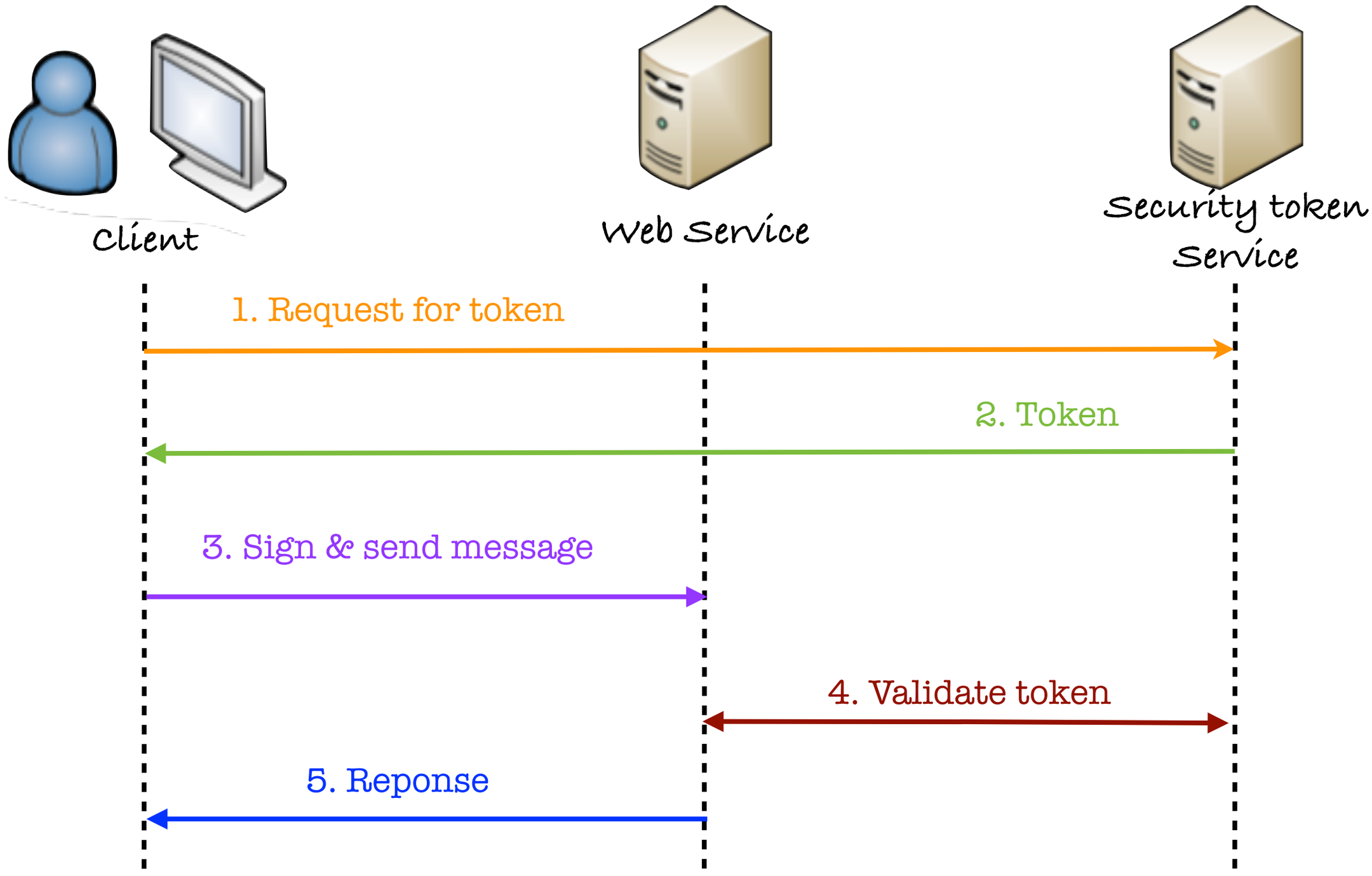
WS-Security Structure



1.Security Tokens

- Security Token express the identity of the source.
- Security Tokens:
  - Simple Token.
  - Binary Tokens.
  - Token reference.

1.Security Tokens



<UsernameToken>

```
....
   <wsse:UsernameToken>
      <wsse:Username>
         anUsername
      </wsse:Username>
      <wsse:Password Type= "wsse:PasswordText">
         aPassword
      </wsse:Password>
   </wsse:UsernameToken>
...
```

<UsernameToken>

```
...
<wsse:Security>
   <wsse:UsernameToken>
      <wsse:Username>anUserName</wsse:Username>
      <wsse:Password Type="wsse:PasswordDigest">
            KE6QugOpkPyT3Eo0SEgT30W4Keg
      </wsse:Password>
      <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg</wsse:Nonce>
      <wsu:Created>
            2012-24-19T00:44:02Z
      </wsu:Created>
   </wsse:UsernameToken>
</wsse:Security>
...
```

<BinarySecurityToken>

```
...
<wsse:Security>
   <wsse:BinarySecurityToken
        ValueType="wsse:X509v3"
        EncodingType="wsse:Base64Binary" >
      aMIYHjClB...
   </wsse:BinarySecurityToken>
</wsse:Security>
...
```

Or wsse:Kerberosv5TGT

Or wsse:Kerberosv5ST

Or wsse:HexBinary

<SecurityTokenReference>

```
...
<wsse:Security>
    <ds:KeyInfo>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#myToken"/>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo>
</wsse:Security>
...
```

2.Confidentiality

- Confidentiality?
- XML-Encryption
- WS-Security uses 2 elements:
  - <ReferenceList>
  - <EncryptedKey>

<ReferenceList>

...

«conf-ReferenceListe.xml»

...

<EncryptedKey>

...

«conf-EncryptedKey.xml»

...

3. Integrity

- Integrity?
- XML-Signature

<Signature>

...
«integ.xml»
...

Error Handling

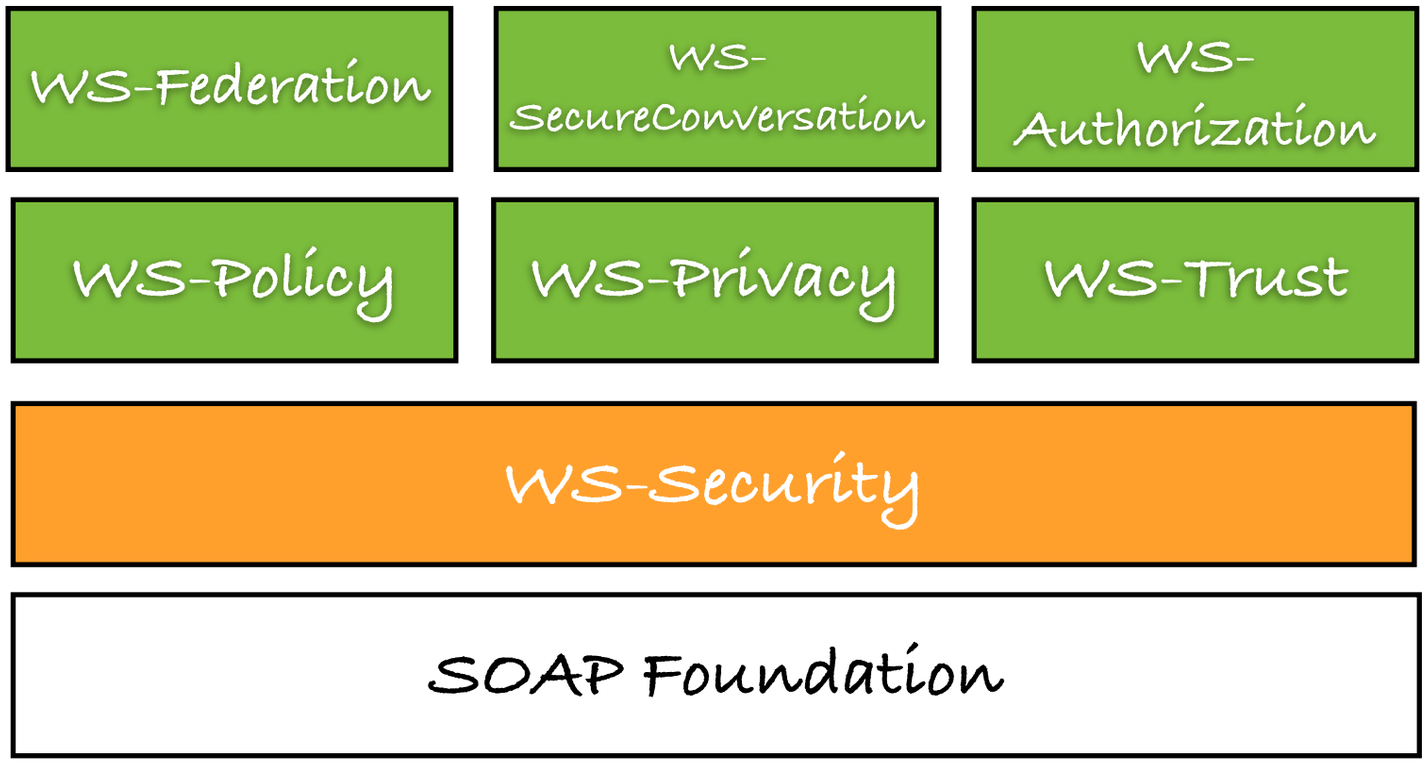
- Errors can occur if:
  - Invalid or unsupported type of security token, signing, or encryption
  - Invalid or unauthenticated security token
  - Invalid signature
  - Decryption failure
  - Referenced security token is unavailable.

Error Handling

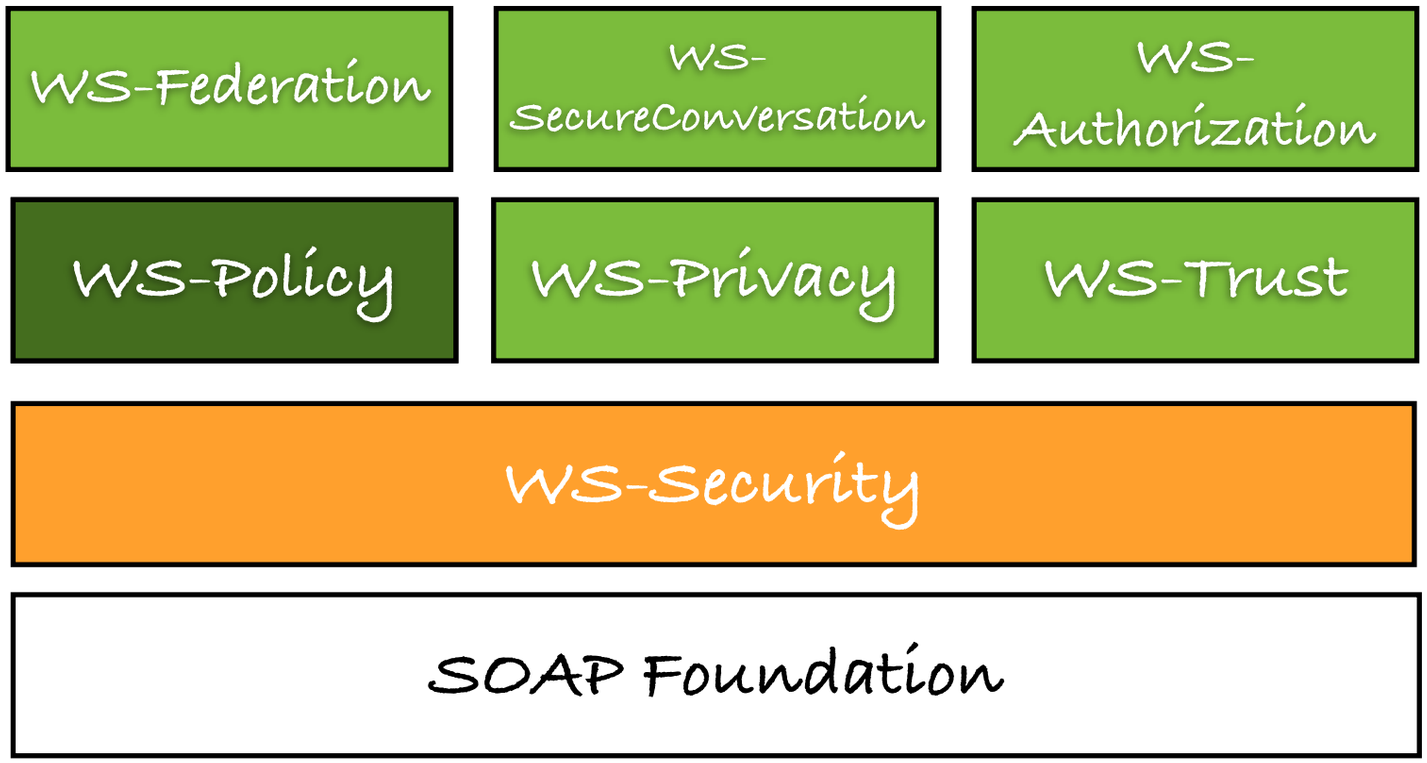
- 2 classes of Errors:
  - Unsupported;
  - Failures;

What are the WS-Security extensions ?

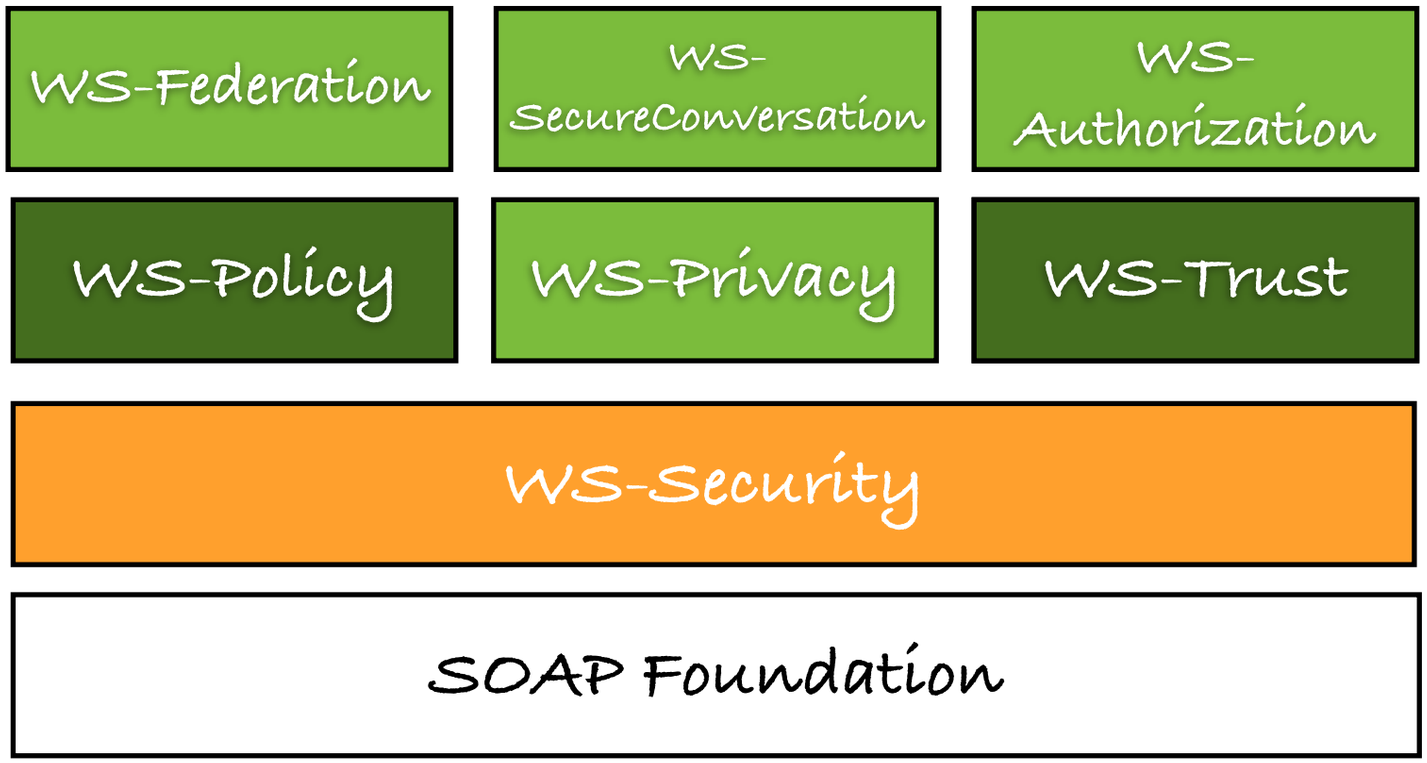
WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

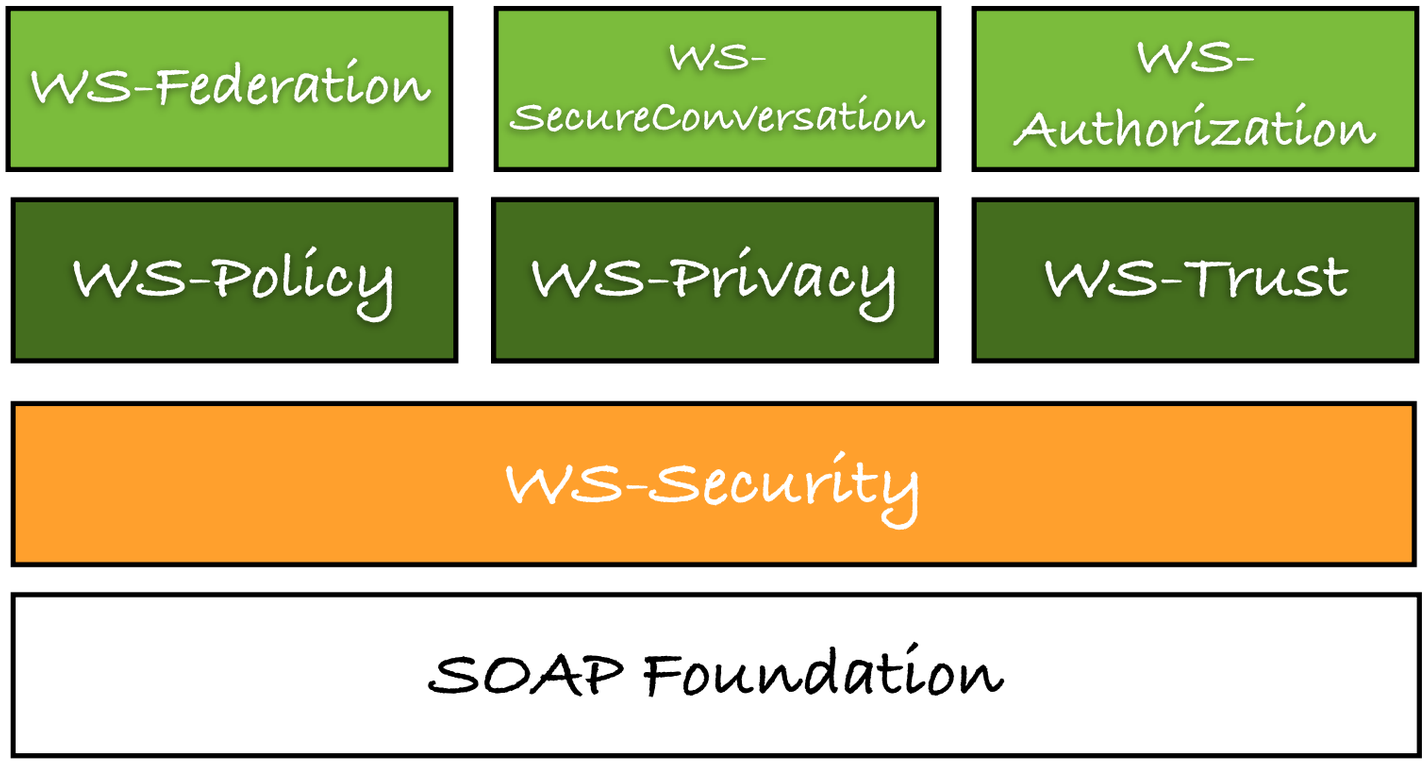
WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

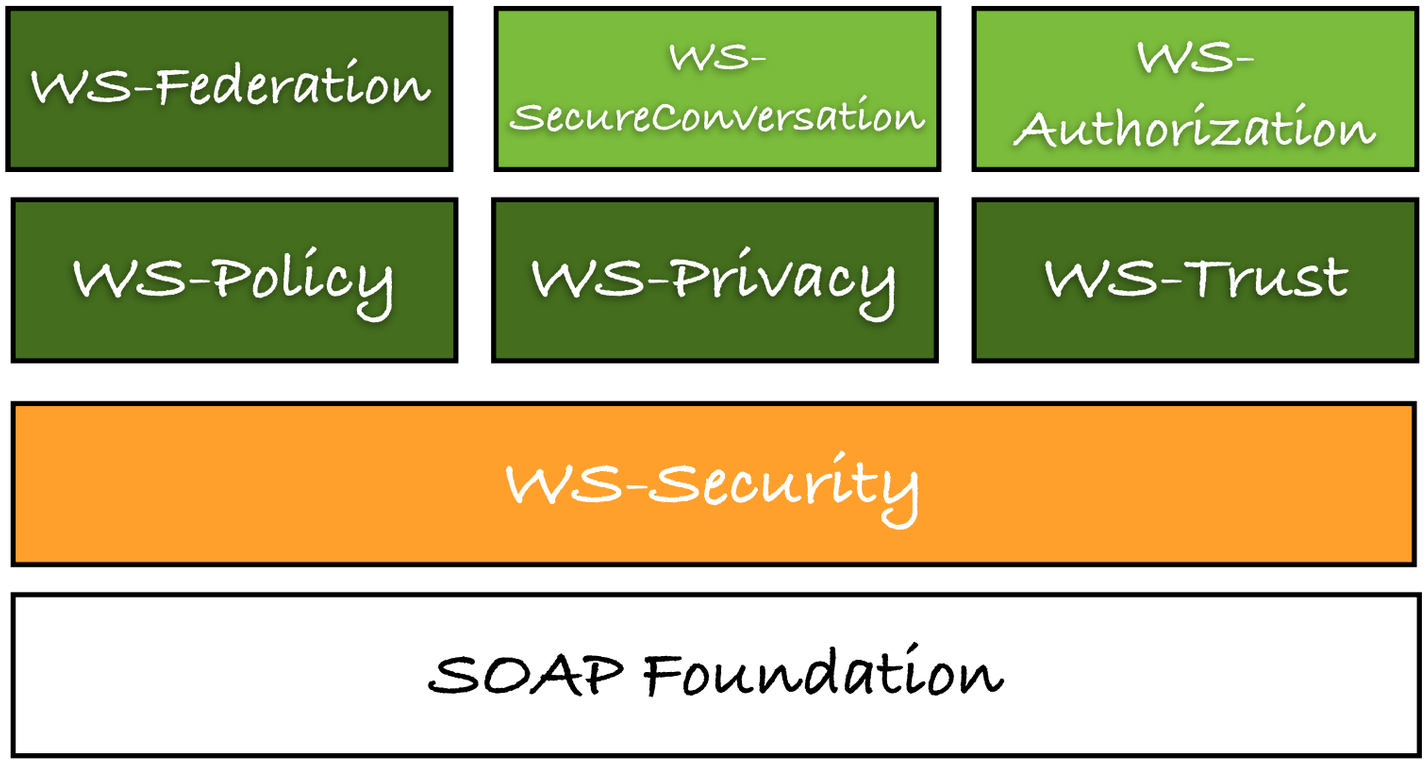
WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

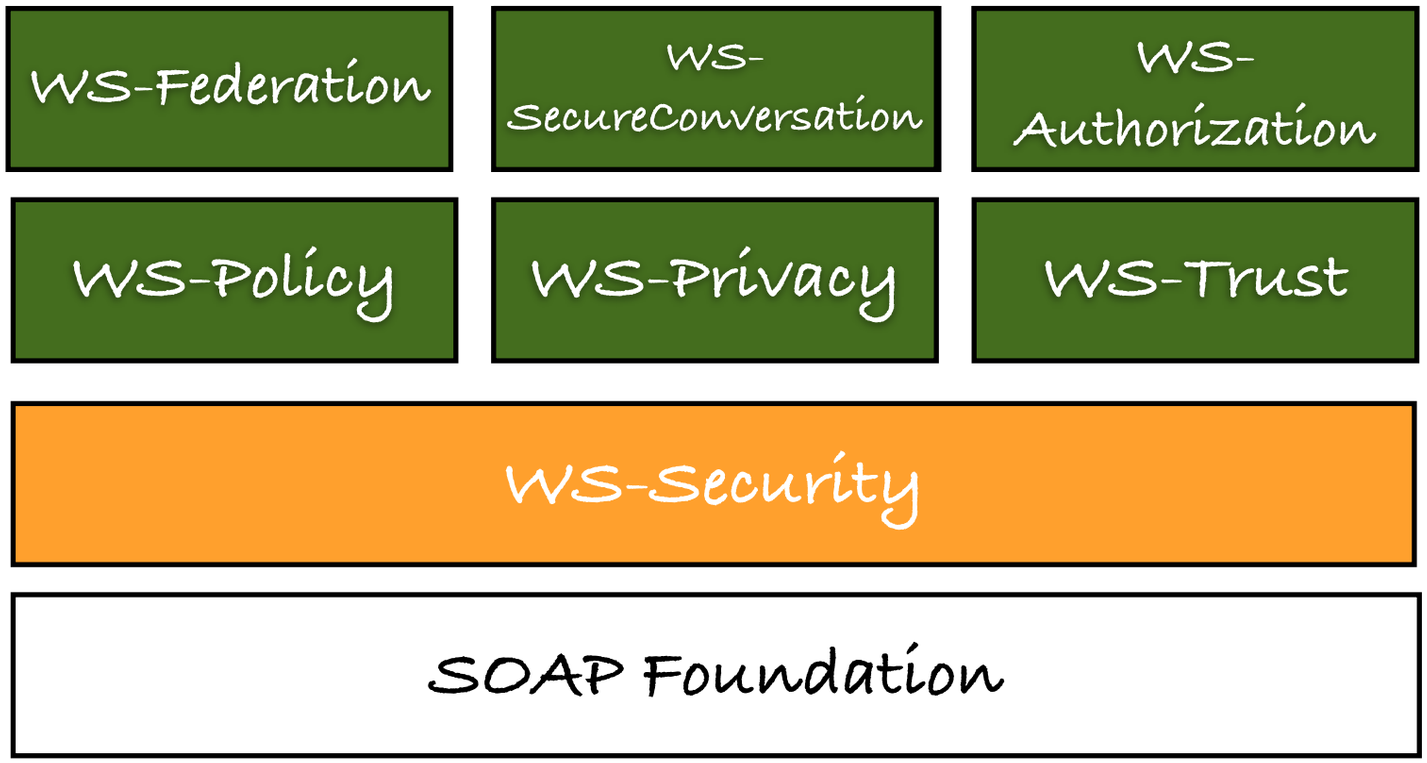
WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

WSS Roadmap


WS-Federation
WS-SecureConversation
WS-Authorization
WS-Policy
WS-Privacy
WS-Trust
WS-Security
SOAP Foundation

WS-Authorization

- Authorization?
- WS-Authorization?

WS-SecureConversation

- problem?
  - Performance
- WS-SecureConversation?

WS-SecureConversation

- How does it works?
  - Secure communication session as SSL
  - Security Context Token (SCT)
- Advantages?
  - Both sides!

WS-SecureConversation

- How to generate the SCT?

- A security token service;
- One of the parties;
- Negociation process.

<SecurityContextToken>

...

«SCT.xml»

...

What about RESTfull-based security ?


OpenID

OpenID

- objectives?
  - one username and one password for multiple applications.
- Definitions:
  - An OpenID is an URL.
  - OpenID is a decentralized mechanism for single sign on.

OpenID protocol

OpenID protocol lets you prove that you own a specific URL

Quick example

My OpenID: http://webserviceulb.myopenid.com/

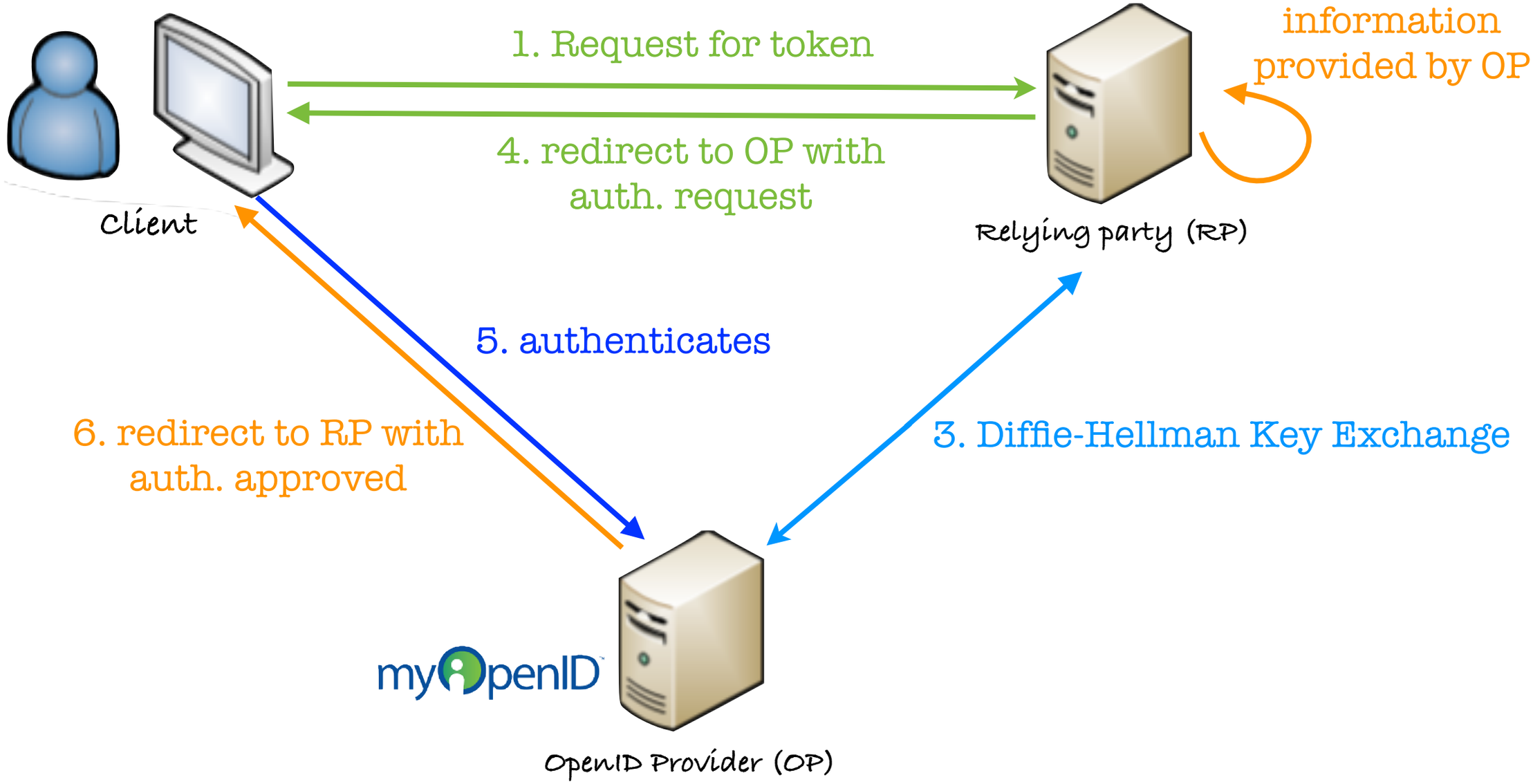
I want to login on **www.sourceforge.net**

OpenID protocol

example.com ➔ http://example.com

2. normalization & discovery

openID = http://webserviceulb.myopenid.com/



OpenID protocol

- Advantages?
  - One pass, many accounts
- Disadvantages?
  - Phishing attack
  - Sniffing

OAUTH

OAUTH - Introduction

- 2 versions
  - OAUTH 1.0 (2006-2007)

flickr Google

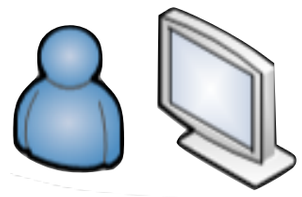
  - OAUTH 2.0 (in developpment)

facebook YAHOO! twitter

Microsoft Google mozilla FOUNDATION

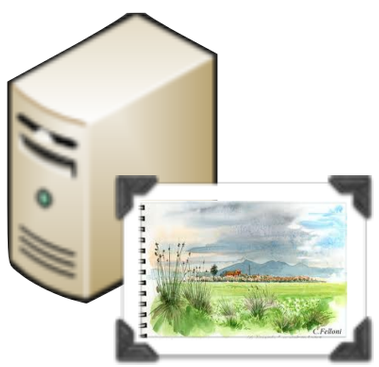
OAUTH 1.0

- authorization grant
  - credential representing resource owner's authorization
- client identifier
  - unique string representing the registration information provided by the client
- redirect uri
  - absolute uri which redirects the user-agent
- native application
  - client on mobile



John → Resource owner

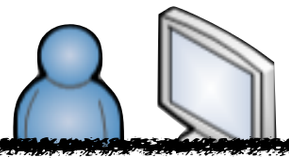
Fnac Service → Client



Flickr → Resource Server

OAUTH 1.0

- authorization grant
  - cred
    reso
- client i
  - unic
    the r
    prov
- redirect
  - absolute URI which redirects the user-agent
- native application
  - client on mobile



Resource owner

Scenario:

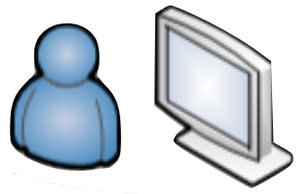
John wants to print photos, stored on Flickr, with Fnac Service

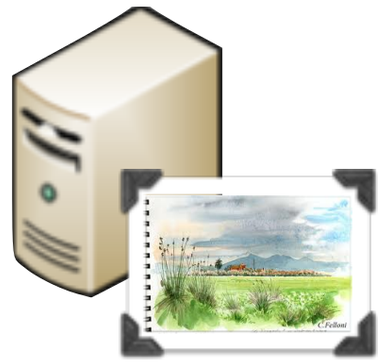
Client

Resource Server

Flickr

Authorization Grant





Resource owner

Client

Resource server

1. Sign up

2. Credential

3. login

4. request **request token**

5. **request token**

6. request grant access

7. login + grand access for Fnac Service with **request token**

8. **access token**

9. requests photos + **access token**

10. requested photos

11. Shows photos

12. selects photos to print

Example of Request

Authentication request

```
GET /photos?size=original&file=vacation.jpg HTTP/1.1
Host: photos.example.net:80
Authorization: OAuth realm="http://photos.example.net/photos",
   oauth_consumer_key="dpf43f3p2l4k3l03",
   oauth_token="nnch734d00sl2jdk",
   oauth_nonce="kllo9940pd9333jh",
   oauth_timestamp="1191242096",
   oauth_signature_method="HMAC-SHA1",
   oauth_version="1.0",
   oauth_signature="tR3%2BTy81lMeYAr%2FFid0kMTYa%2FWM%3D"
```

➔ the client would like to access a protected resource located at http://photos.example.net/photos with the parameters size=original, file=vacation.jpg

Example of Request

Message to sign

GET&http%3A%2F%2Fphotos.example.net%2Fphotos&file%3Dvacation.jpg%26oauth_consumer_key%3Ddpf43f3p2l4k3l03%26oauth_nonce%3Dkllo9940pd9333jh%26oauth_signature_method%3DHMAC-SHA1%26oauth_timestamp%3D1191242096%26oauth_token%3Dnnch734d00sl2jdk%26oauth_version%3D1.0%26size%3Doriginal

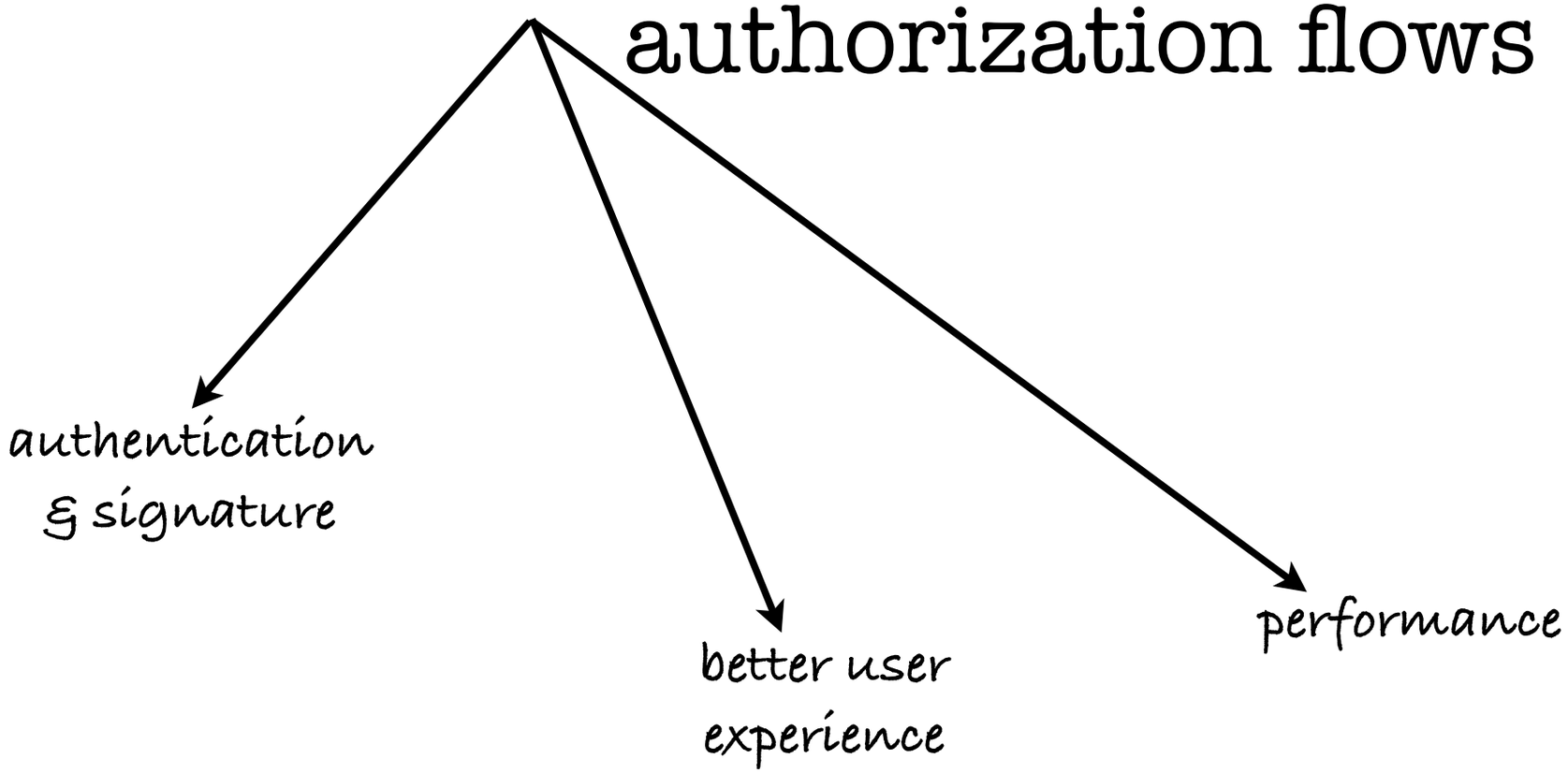
HMAC secrets

client secret + token secret

OAuth 2.0

objectives

client developper simplicity & multiple authorization flows



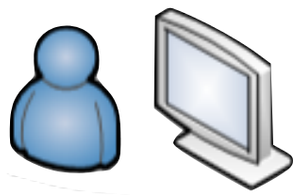
OAuth 2.0

Fnac Service

Client

1. Authorization request

2. Authorization grant



John

Resource owner

3. Authorization grant

4. Access token

Authorization server

5. Access token

6. Protected resource

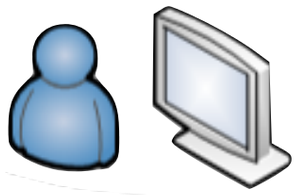
Resource server

Flickr

FLOWS

- ☐ Authorization code
- ☐ Implicit (see later)
- ☐ Resource owner password credential
  - Client access to the resource owner credential (username/password)
- ☐ Client credentials
  - Client needs to access resources not related to a specific resource owner

Authorization code



Resource owner

Client

Authorization server

Resource server

1. Access client

2. Redirect URI

3. User authenticates

4. Redirect to redirect URI with Auth. Code

auth. server can add a refresh token

5. Access to redirect URI

6. send Auth. Code + Client id & secret

7. Access token

8. Access client

9. Request photos + access token

10. Requested photos

11. Show photos

Auth. Code Request/Responses

Client Request Authorization

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Parameter	Required/Optional	Detail
response_type	REQUIRED	must be set to code
client_id	REQUIRED	the client identifier
redirect_uri	OPTIONAL	absolute uri
scope	OPTIONAL	access token scope
state	RECOMMENDED	preventing cross-site request forgery

Auth. Code Request/Responses

End-User Grant Authorization

```
HTTP/1.1 302 Found
 Location: http://example.com/rd#access_token=FJQbwq9&expires_in=3600
```

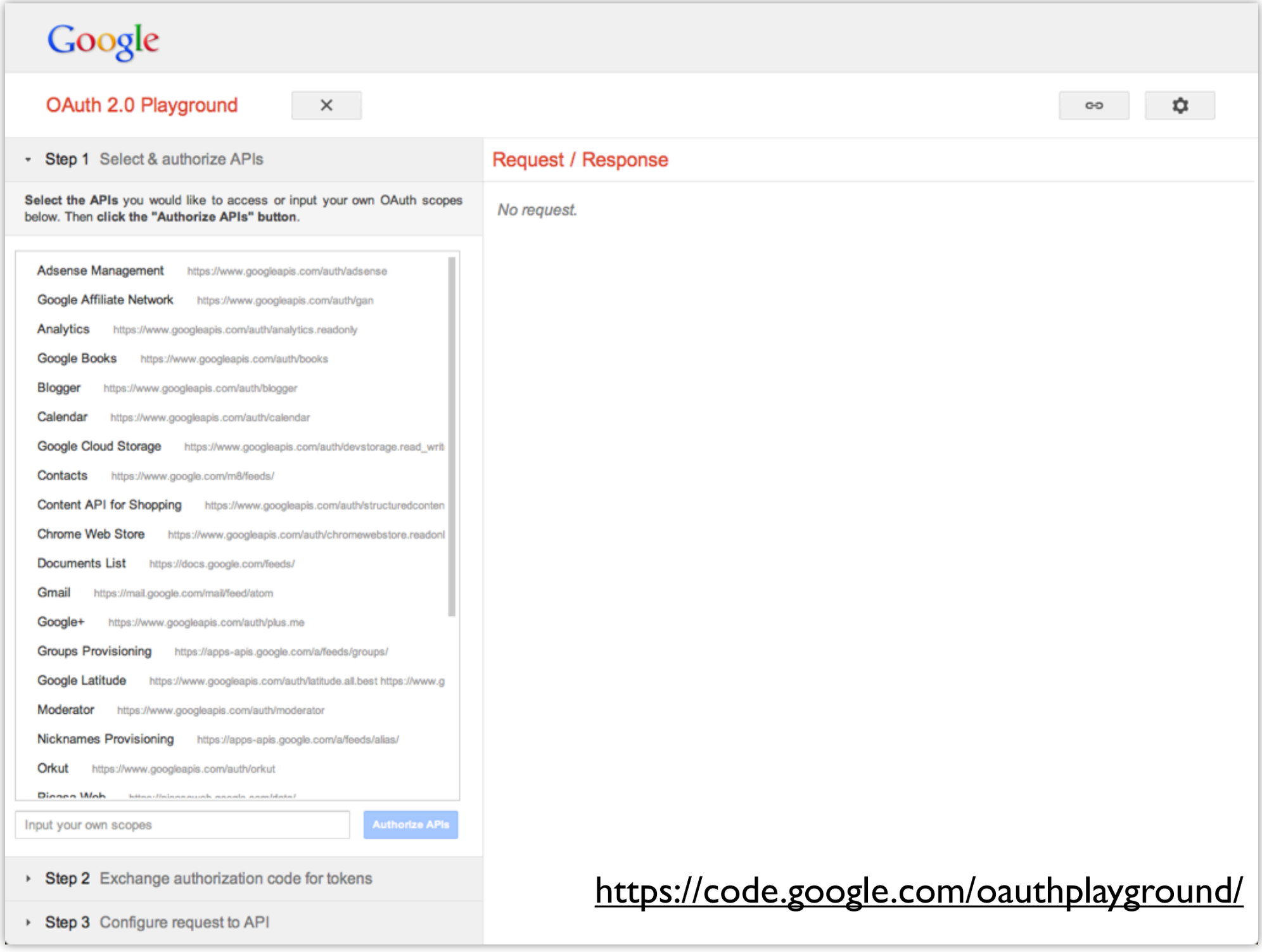
End-User Denies Authorization

```
HTTP/1.1 302 Found
 Location: http://example.com/rd#error=user_denied
```

Client Extracts Access Token

```
GET /rd HTTP/1.1
 Host: example.com
```

Démo.

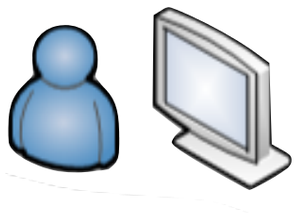


https://code.google.com/oauthplayground/

FLOWS

- [ ] Authorization code
- [ ] Implicit
- [ ] Resource owner password credential
  - Client access to the resource owner credential (username/password)
- [ ] Client credentials
  - Client needs to access resource not related to a specific resource owner

Implicit



Resource owner

Client

Authorization server

Resource server

1. Access app

2. Login via auth server

3. Login

4. Redirect to client redirect URI + **Access token**

6. Access to redirect URI + Access token

7. Logged in

→ not support refresh token
→ for native application

Implicit Request/Responses

Client Request Authorization

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
  Host: server.example.com
```

Access Token Response

```
HTTP/1.1 302 Found
   Location: http://example.com/cb#access_token=2YotnFZFEjr1zCsicMWpAA
        &state=xyz&token_type=example&expires_in=3600
```

Access Error Response

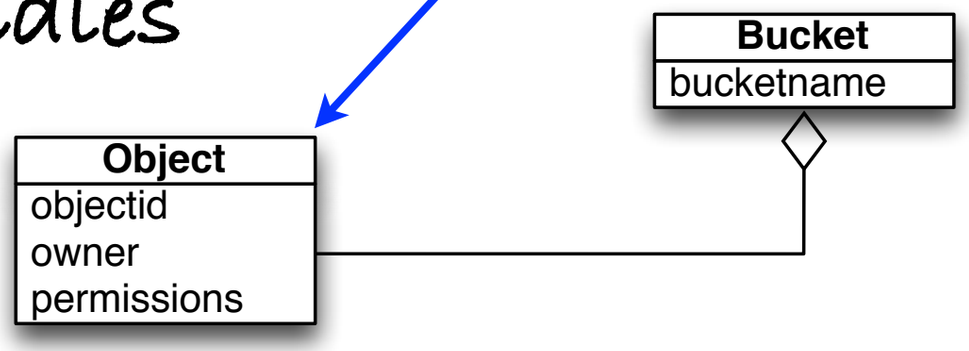
```
HTTP/1.1 302 Found
  Location: https://client.example.com/cb#error=access_denied&state=xyz
```

Amazon

Web service

Amazon S3

- S3 = Simple Store Service
- Allows to store files in a remote storage (DropBox)
- Handles



- Address of an object:
  - http://s3.amazonaws.com/bucketname/objectid

USER authentication

Request

```
GET /photos/puppy.jpg HTTP/1.1
Host: johnsmith.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000
Authorization: AWS OPN5J17HBGZHT7JJ3X82 xXjDGYUmKxnwqr5KXNPGldn5LbA=
```

AWSAccessKeyId

Signature

```
Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of( YourSecretAccessKeyID, StringToSign ) ) );
StringToSign = HTTP-Verb + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Date + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

```
GET\n
\n
\n
Tue, 27 Mar 2007 19:36:42 +0000\n
/johnsmith/photos/puppy.jpg
```

Merci

Dank u

thank you

:-)

Questions ?