OAuth 2.0

Ein Standard wird erwachsen

Uwe Friedrichsen (codecentric AG) – Berlin Expert Days 2013 – 4. April 2013

Uwe Friedrichsen

@ufried

<session>
<no-code>
<motivation />
<history />
<solution />
<extensions />
<criticism />
<tips />
</no-code>
<code>
<authzorization />
<token />
<resource />
</code>
<wrap-up />
</session>

```
{ „session“ : {
        „no-code“ : [
                „motivation“,
                „history“,
                „solution“,
                „extensions“,
                „criticism“,
                „tips“
        ],
        „code“ : [
                „authorization“,
                „token“,
                „resource“
        ],
        „wrap-up“ : true
}
```

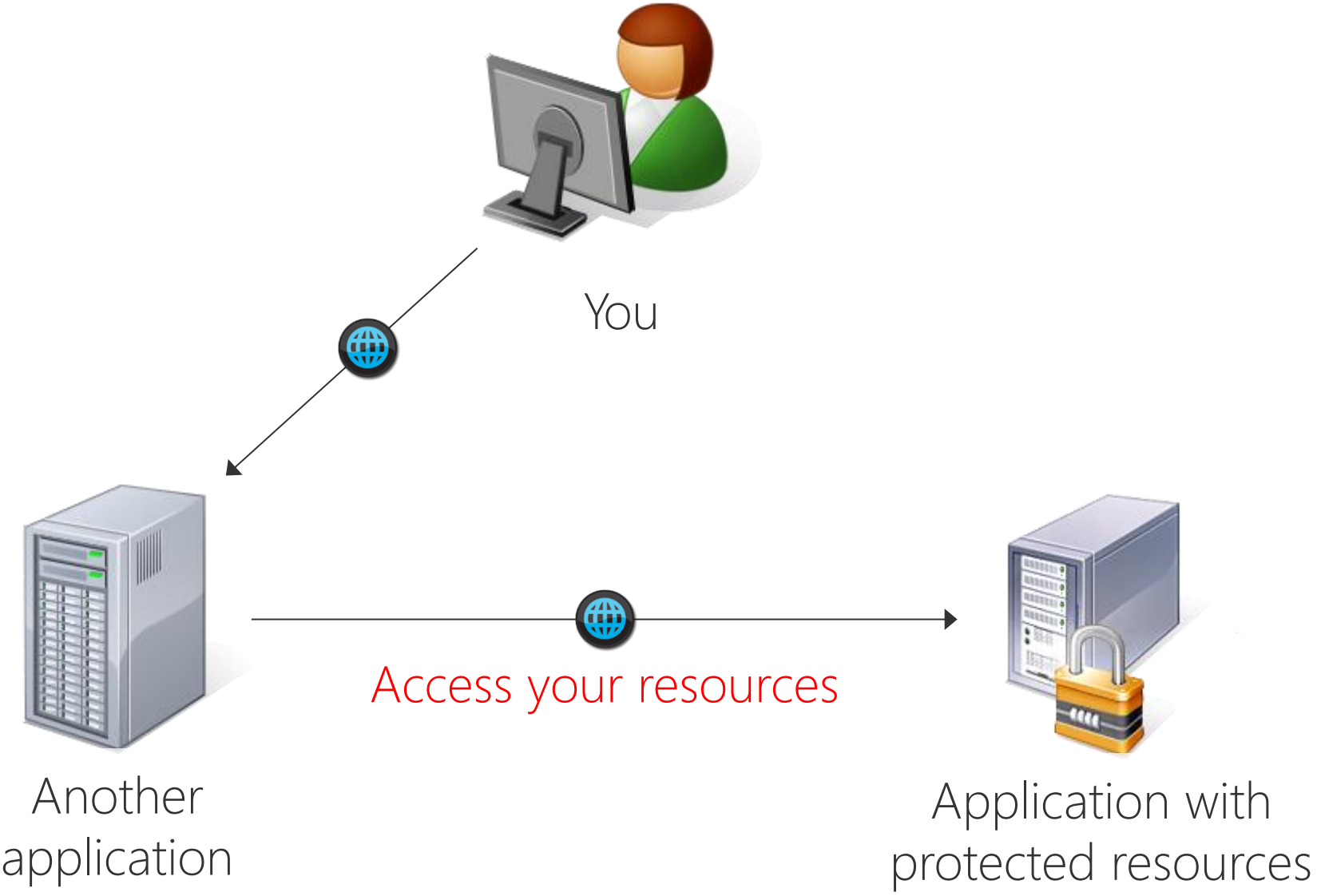
Players

You

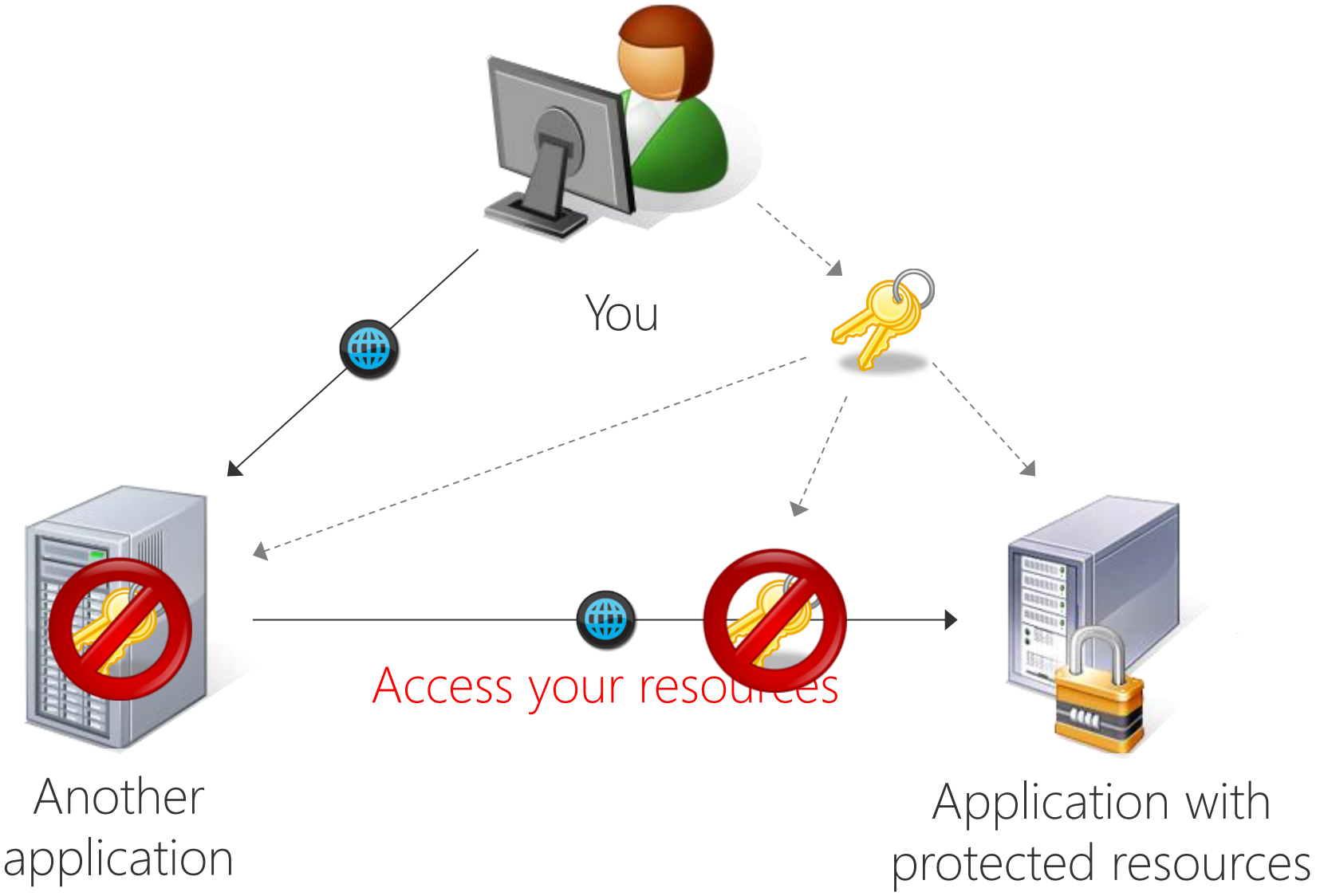
Another application

Application with protected resources

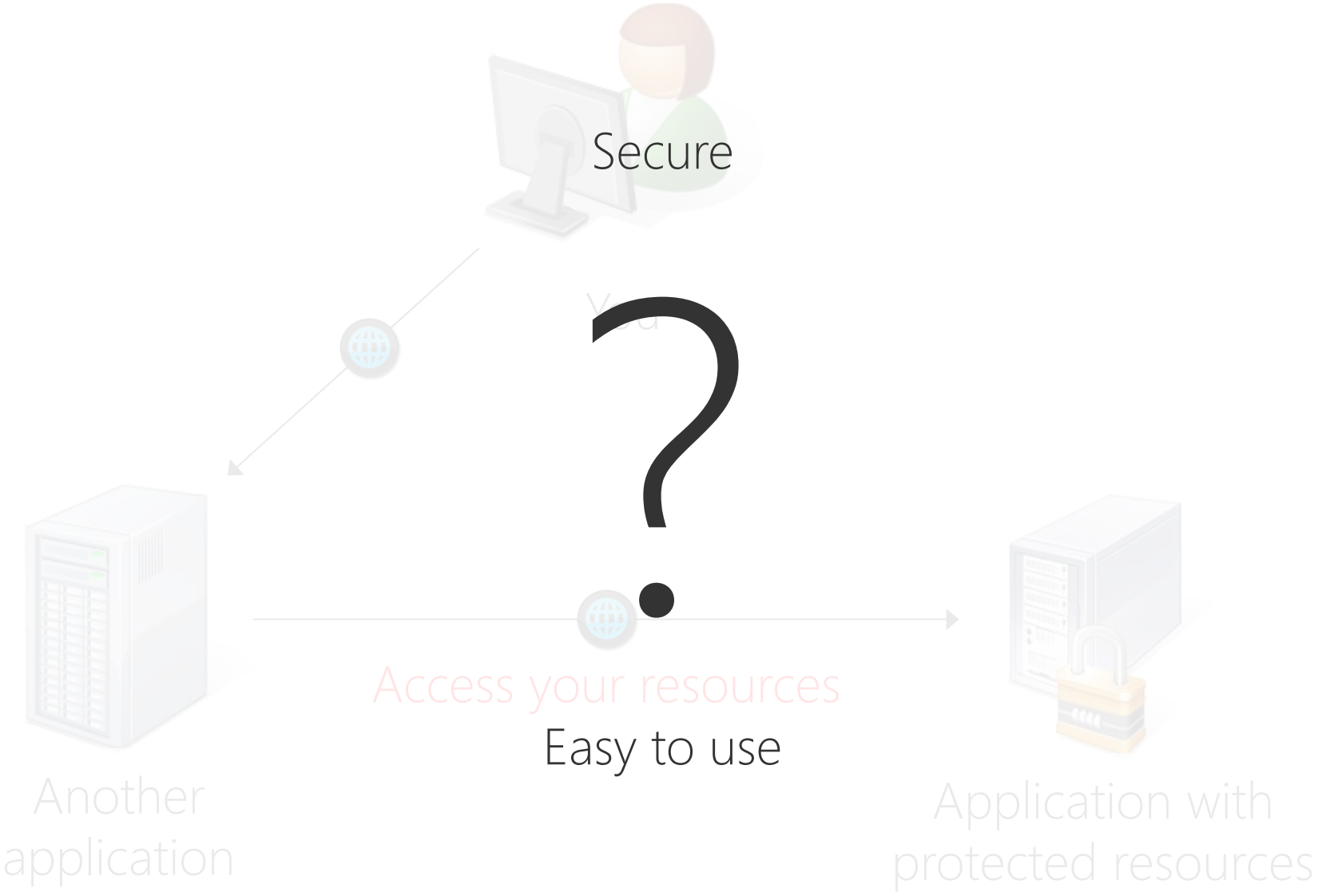
Assignment


You
Access your resources
Another application
Application with protected resources

Problem


You
Access your resources
Another application
Application with protected resources

Challenge


Secure
?
Access your resources
Easy to use
Another application
Application with protected resources

OAuth 1.0

- Started by Twitter in 2006
- 1st Draft Standard in 10/2007
- IETF RFC 5849 in 4/2010

- Widespread

- Complex Client Security Handling
- Limited Scope
- Not extendable
- Not „Enterprise-ready“

OAuth 2.0

- Working Group started 4/2010
- 31 Draft Versions
- Eran Hammer-Laval left 7/2012 *
- IETF RFC 6749 in 10/2012

* http://hueniverse.com/2012/07/oauth-2-0-and-the-road-to-hell/

Players revisited

You

Another application

Application with protected resources

Players revisited

You

Authorization Server

Client Application

Resource Server

Solution (Step 1)

You

2.Client XYZ wants an authorization code

3. User: „Yes, it's okay"

4. Here is an authorization code for client XYZ

Authorization Server

1. I want an authorization code

5. Here you are

Client Application

Resource Server

Solution (Step 2)

You

6. I want to trade my authorization code for an access token

Authorization Server

7. Here you are

Client Application

Resource Server

Solution (Step 3)

You

Authorization Server

Client Application

Give me some resources.
Here is my access token, btw.

...

Resource Server

A few more Details

- TLS/SSL
- Endpoints
- Client Types
- Client Identifier
- Client Authentication
- Redirect URI
- Access Token Scope
- Refresh Token
- Client State


2.Client XYZ wants an authorization code
3. User: „Yes, it's okay"
4. Here is an authorization code for client XYZ
You
Authorization Server
1. I want an authorization code
5. Here you are
GET /authorize?↵
response_type=code&↵
client_id=s6BhdRkqt3&↵
state=xyz&↵
redirect_uri=https%3A%2F%2Fclient%2E↵
example%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
Client Application
Resource Server

2.Client XYZ wants an authorization code

3. User: „Yes, it's okay"

4. Here is an authorization code for client XYZ

You

Authorization Server

1. I want an authorization code

5. Here you are

```
HTTP/1.1 302 Found
Location: ↵
  https://client.example.com/cb?↵
  code=SplxlOBeZQQYbYS6WxSbIA&↵
  state=xyz
```

Client Application

Resource Server

You

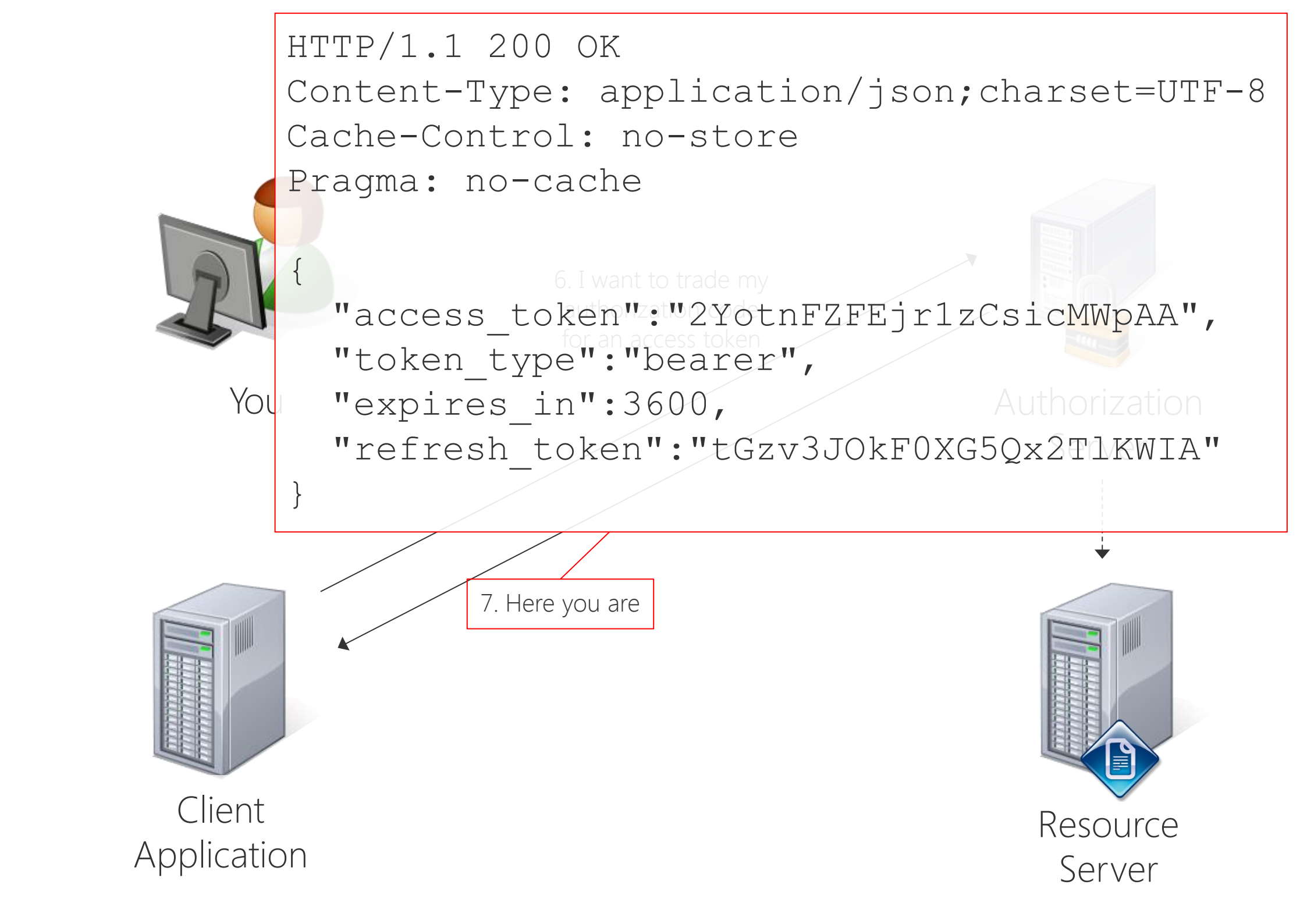
6. I want to trade my authorization code for an access token

Authorization Server

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&↵
  code=SplxlOBeZQQYbYS6WxSbIA&↵
  redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

Client Application

Server


HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"bearer",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
7. Here you are
Client Application
Resource Server

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

Client Application

Give me some resources.
Here is my access token, btw.

...

Resource Server

More flows & Extensions

- Implicit Grant
- Resource Owner Password Credentials Grant
- Client Credentials Grant
- Refresh Token Grant
- Standard & custom Extensions
- Standards based on OAuth 2.0

Criticism

- Too many compromises
- No built-in security
- Relies solely on SSL
- Bearer Token
- Self-encrypted token

Tips

- Turn MAY into MUST
- Use HMAC Tokens
- Use HMAC to sign Content
- No self-encrypted token
- Always check the SSL Certificate

How does the code feel like?

using Apache Amber 0.22


2.Client XYZ wants an authorization code
3. User: „Yes, it's okay"
4. Here is an authorization code for client XYZ
You
Authorization Server
1. I want an authorization code
5. Here you are
GET /authorize?↵
response_type=code&↵
client_id=s6BhdRkqt3&↵
state=xyz&↵
redirect_uri=https%3A%2F%2Fclient%2E↵
example%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
Client Application
Resource Server

Authorization Endpoint (1)

```
@Path("/authorize")
public class AuthorizationEndpoint {
    @Context
    private SecurityDataStore securityDataStore;

    @GET
    @Consumes(OAuth.ContentType.URL_ENCODED)
    public Response authorize(@Context HttpServletRequest request) {
        // Do the required validations
        OAuthAuthzRequest oauthRequest = wrapAndValidate(request);
        validateRedirectionURI(oauthRequest);

        // Actual authentication not defined by OAuth 2.0
        // Here a forward to a login page is used
        String loginURI = buildLoginURI(oauthRequest);
        return Response.status(HttpServletResponse.SC_FOUND)
            .location(new URI(loginUri)).build();
    }

    ...
```

Authorization Endpoint (2)

```
...

private OAuthAuthzRequest wrapAndValidate(HttpServletRequest req) {
    // Implicitly validates the request locally
    return new OAuthAuthzRequest(req);
}

...
```

Authorization Endpoint (3)

```
...

private void validateRedirectionURI(OAuthAuthzRequest oauthReq) {
    String redirectionURISent = oauthReq.getRedirectURI();
    String redirectionURIStored = securityDataStore
        .getRedirectUriForClient(oauthReq.getClientId());

    if (!redirectionURIStored
            .equalsIgnoreCase(redirectionURISent)) {
        OAuthProblemException oAuthProblem =
            OAuthProblemException
                .error(OAuthError.CodeResponse.ACCESS_DENIED,
                       "Invalid Redirection URI");
        oAuthProblem.setRedirectUri(redirectionURISent);
        throw oAuthProblem;
    }
}

...
```

Authorization Endpoint (4)

```
    ...

    private String buildLoginURI(OAuthAuthzRequest oauthRequest) {
        String loginURI = getBaseLoginURI();  // As an example

        loginURI += "&" + OAuth.OAUTH_RESPONSE_TYPE + "=“
            + oauthRequest.getParam(OAuth.OAUTH_RESPONSE_TYPE);
        loginURI += "?" + OAuth.OAUTH_CLIENT_ID + "=“
            + oauthRequest.getClientId();
        loginURI += "&" + OAuth.OAUTH_REDIRECT_URI + "=“
            + redirectUri;
        loginURI += "&" + OAuth.OAUTH_SCOPE + "=“
            + getParam(OAuth.OAUTH_SCOPE);
        loginURI += "&" + OAuth.OAUTH_STATE + "=“
            + getParam(OAuth.OAUTH_STATE);

        return loginURI;
    }
}
```

2.Client XYZ wants an authorization code

3. User: „Yes, it's okay"

4. Here is an authorization code for client XYZ

You

Authorization Server

1. I want an authorization code

5. Here you are

```
HTTP/1.1 302 Found
Location: ↵
  https://client.example.com/cb?↵
  code=SplxlOBeZQQYbYS6WxSbIA&↵
  state=xyz
```

Client Application

Resource Server

Login page handler

```
private void getAndSendAuthorizationCode(HttpServletRequest req,
                                         HttpServletResponse resp) {
    // Assuming login was successful and forwarded
    // parameters can be found in the request
    String userId = (String) request.getAttribute("userId");
    String clientId =
        (String) request.getAttribute(OAuth.OAUTH_CLIENT_ID);

    // Create a new authorization code and store it in the database
    String authzCode =
        securityDataStore.getAuthorizationCode(userId, clientId);

    // Redirect back to client
    String redirectUri =
        (String) req.getAttribute(OAuth.OAUTH_REDIRECT_URI);
    redirectUri += "?" + OAuth.OAUTH_CODE + "=" + authzCode);
    redirectUri += "&" + OAuth.OAUTH_STATE + "="
        + request.getAttribute(OAuth.OAUTH_STATE);
    resp.sendRedirect(redirectUri);
}
```

You

6. I want to trade my authorization code for an access token

Authorization Server

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&↵
  code=SplxlOBeZQQYbYS6WxSbIA&↵
  redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

Client Application

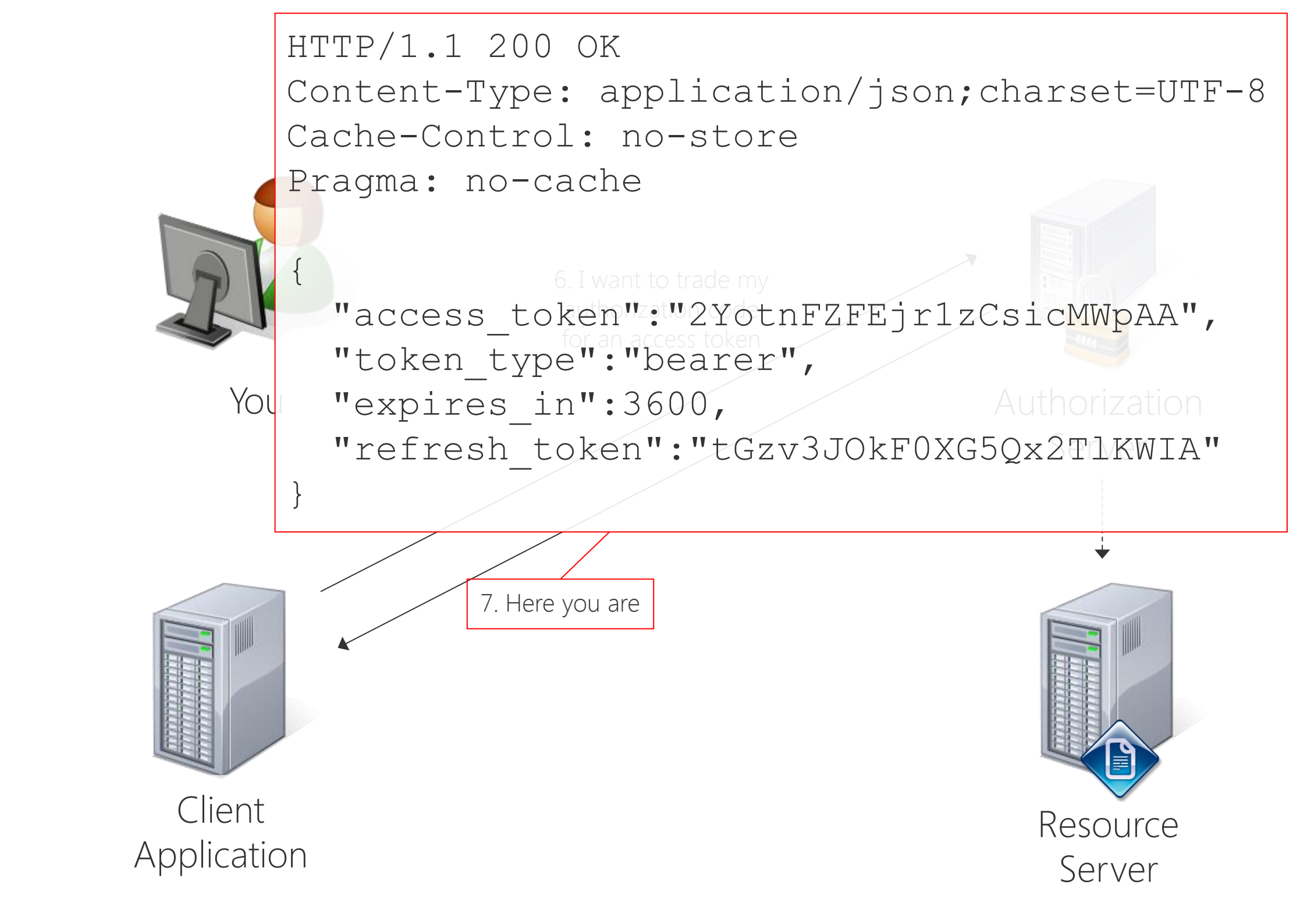
Server

Token Endpoint (1)

```
@Path("/token")
public class TokenEndpoint {
    ...
    @POST
    public Response authorize(@Context HttpServletRequest request,
        @HeaderParam(AUTHORIZATION) String authorizationHeader) {
        // Do the required validations
        validateClient(authorizationHeader);
        OAuthTokenRequest oauthRequest = new OAuthTokenRequest(request);
        validateRedirectionURI(oauthRequest);

        OAuthToken token = securityDataStore
            .exchangeAuthorizationCodeForAccessToken(oauthRequest);

        OAuthResponse oauthResponse = buildOAuthResponse(token);
        return Response.status(oAuthResponse.getResponseStatus())
            .entity(oAuthResponse.getBody()).build();
    }

    ...
```

Token Endpoint (2)

```
...

private void validateClient(String authorizationHeader) {
    Pattern headerPattern = Pattern.compile("\\s+");
    String[] headerParts = headerPattern.split(authorizationHeader);

    byte[] encoded = headerParts[1].getBytes();
    String decoded = new String(Base64.decode(encoded),
                                Charset.forName("UTF-8"));
    String[] clientParts = StringUtils.split(decoded, ":", 2);

    String clientId = clientParts[0];
    String clientSecret = clientParts[1];

    if (!securityDataStore.isValidClient(clientId, clientSecret)) {
        ...  // Create and throw an OAuthProblemException
    }
}

...
```


HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"bearer",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
7. Here you are
Client Application
Resource Server

Token Endpoint (3)

```
    ...

    private OAuthResponse buildOAuthResponse(OAuthToken token) {
        return OAuthASResponse
            .tokenResponse(HttpServletResponse.SC_OK)
            .setAccessToken(token.getAccessToken())
            .setTokenType(TokenType.BEARER)
            .setExpiresIn(token.getExpiresIn())
            .setRefreshToken(token.getRefreshToken())
            .setScope(token.getScope())
            .buildJSONMessage();
    }
}
```

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

Client Application

Give me some resources.
Here is my access token, btw.

...

Resource Server

Resource Filter (1)

```
public class AuthorizationFilter implements ContainerRequestFilter {
    @Context
    private SecurityDataStore securityDataStore;

    @Context
    private HttpServletRequest httpServletRequest;

    @Override
    public ContainerRequest filter(ContainerRequest request) {
        String accessToken = extractAccessToken();
        validateAccessToken(accessToken);
        return request;
    }

    ...
```

Resource Filter (2)

```
...

private String extractAccessToken() {
    OAuthAccessResourceRequest oauthRequest =
        new OAuthAccessResourceRequest(httpServletRequest);
    return oauthRequest.getAccessToken();
}

...
```

Resource Filter (3)

```
    ...

    private void validateAccessToken(String accessToken) {
        if (!securityDataStore.isValidAccessToken(accessToken)) {
            throw new AuthorizationFailedException(
                "Unknown or expired token!");
        }
    }
}
```

Summary

- OAuth 2.0 is ready for use
- Quite easy to use
- Don‘t go for least security

Uwe Friedrichsen

@ufried

uwe.friedrichsen@codecentric.de

http://www.slideshare.net/ufried/
http://blog.codecentric.de/author/ufr