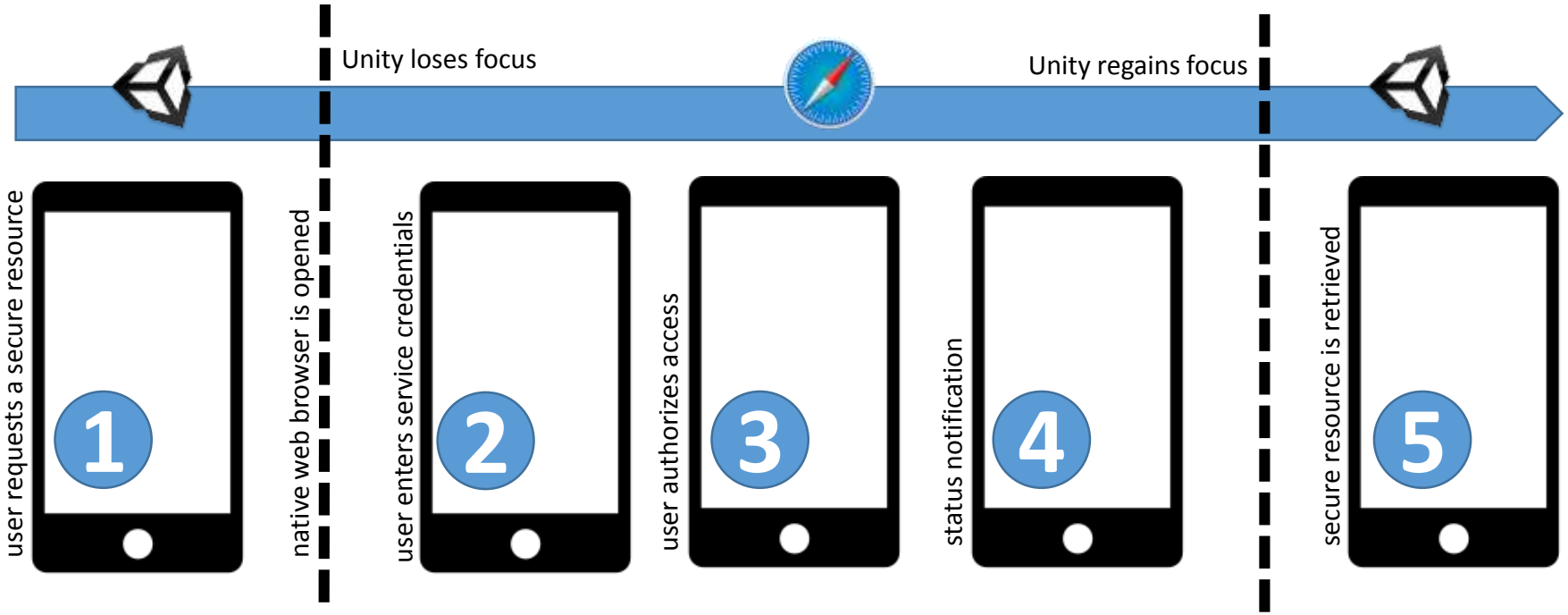
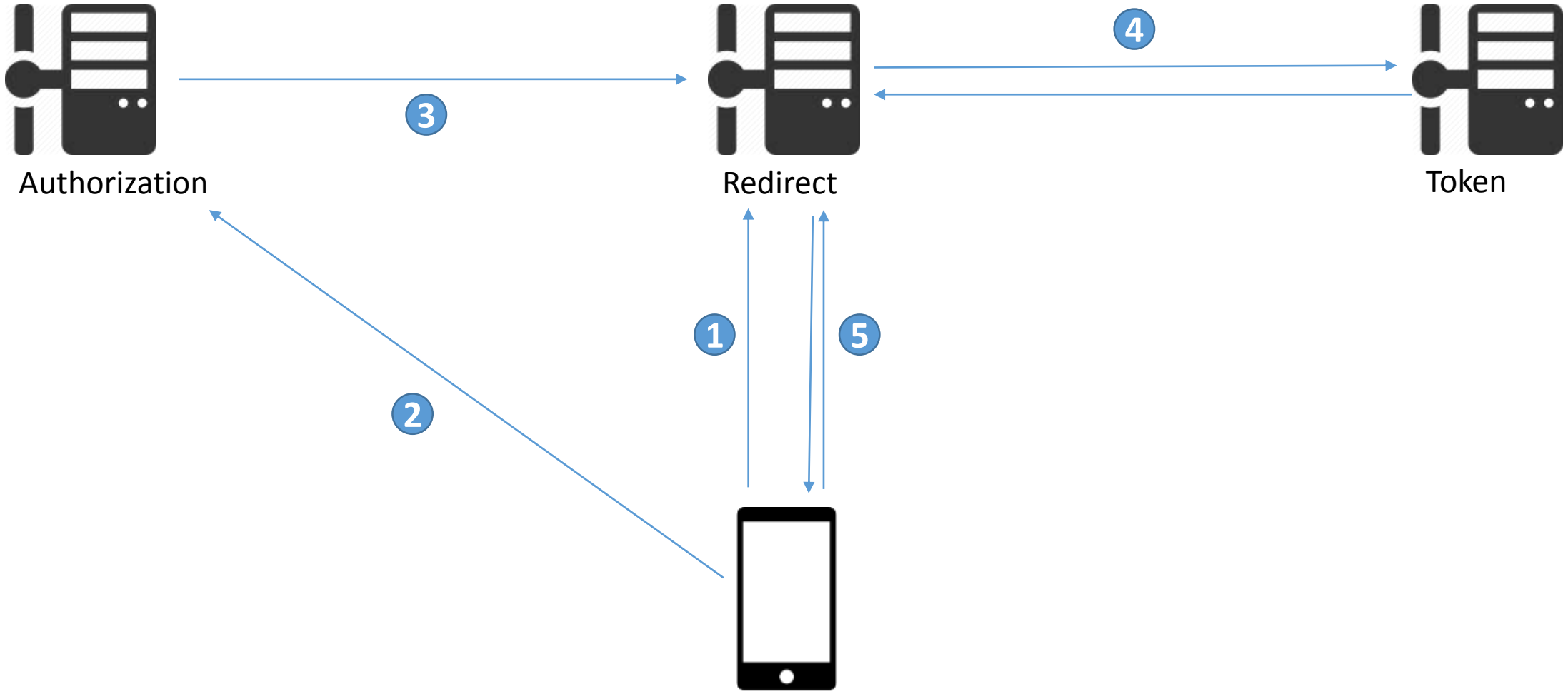
Web Server Authorization Code Grant Flow

https://tools.ietf.org/html/rfc6749



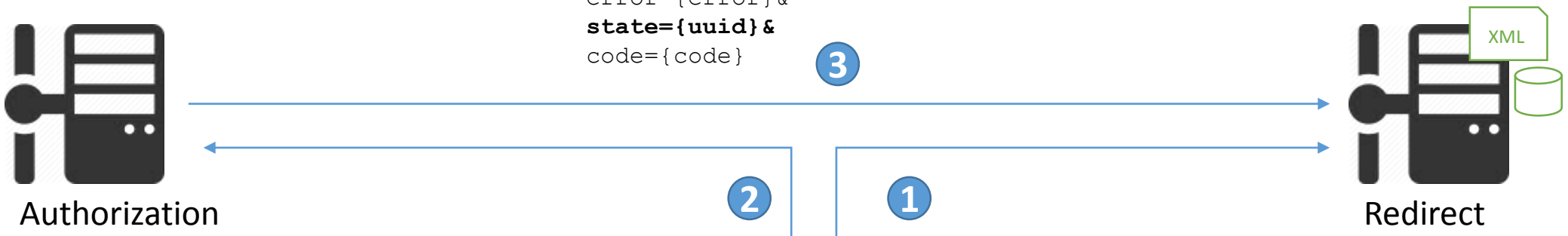
1. The HTTP call to the secure service is intercepted based on the operation's host name. Unity builds an authorization server URI and opens the device's native web browser.
2. The user is prompted to enter their credentials for the secure service.
3. The user then approves (or denies) the application access to the secure service.
4. Authorization code and access token exchange occurs on our redirect server. The user is presented with a success or failure message and is prompted to return to the Unity application.
5. Unity obtains the access token from our redirect server and uses it to access the secure resource.

Web API Kit EXTENSION OAuth 2



Web API Kit EXTENSION OAuth 2

*Google example




```
GET https://accounts.google.com/o/oauth2/auth?
        client_id={client_id}&
        redirect_uri={redirect_uri(1)}&
        state={uuid}&
        scope={list of scopes}&
        response_type=code&
        access_type=offline&
        approval_prompt=force&
        include_granted_scopes=true
```

```
POST {redirect_uri(1)}/register-state
Content-Type: application/hg+json
Accept: application/json
```

```
{ "s" : "uuid",
  "i" : "client_id",
  "k" : "client_secret",
  "t_u" : "token_server_uri",
  "t_p" : "token_uri_params",
  "r_u" : "redirect_uri(1)",
  "o" : [ /* user options */ ] }
```

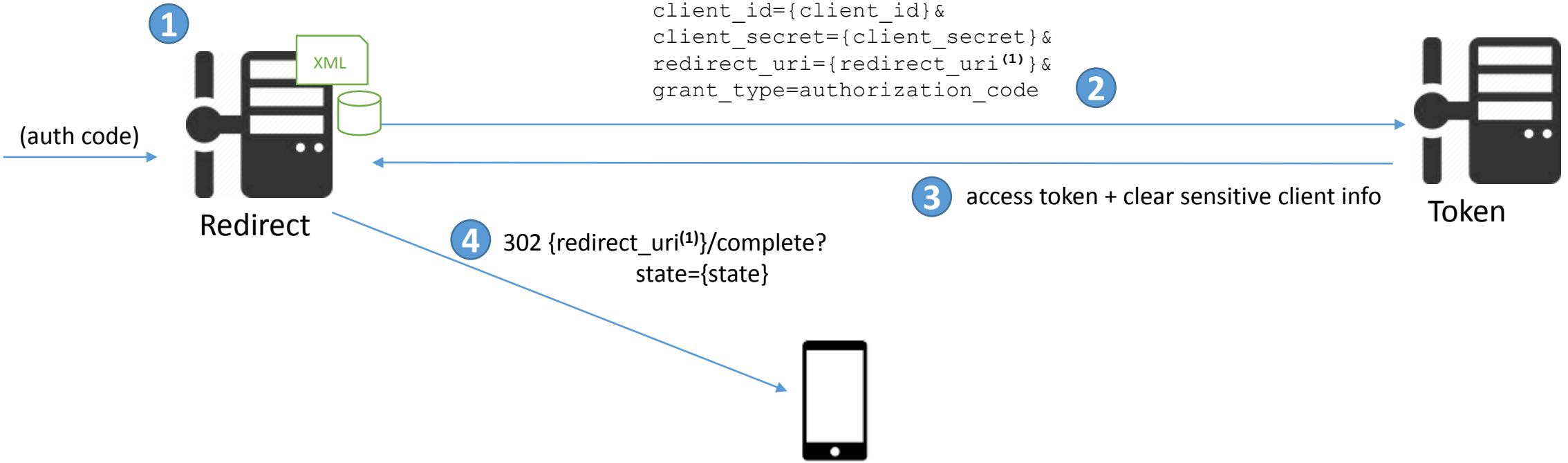
1. The Unity application generates a random GUID which is stored on our redirect server.
2. The same GUID is then sent as the 'state' parameter in our request for an authorization code using the native browser.
3. The authorization server then redirects our user agent and includes the authorization code as well as our 'state' GUID.

(1) eXist-db redirect URI example: `http://redirect.mydomain.com:8080/exist/restxq/oauth-interceptor`

*Google example

```
POST https://www.googleapis.com/oauth2/v3/token
Content-Type: application/x-www-form-urlencoded

code={auth_code}&
client_id={client_id}&
client_secret={client_secret}&
redirect_uri={redirect_uri(1)}&
grant_type=authorization_code
```



1. The redirect server locates a Unity registration by the ‘state’ GUID parameter. It then builds the request for step 2.
2. The redirect server requests an access token from the token server by passing the authorization code and client info.
3. The access token is stored and sensitive client info is cleared from the registration record.
4. Once again, the user agent is redirected to a page notifying them to return to the Unity application.

(1) eXist-db redirect URI example: `http://redirect.mydomain.com:8080/exist/restxq/oauth-interceptor`

*Google example

1. When Unity regains application focus, it queries our redirect server using the previously assigned ‘state’ GUID.
   Unity has one chance to retrieve the stored access token from our redirect server.
   Any consecutive requests will result in a 410 GONE response.
   If the authorization or access token exchange failed for any reason the request will result in a 403 FORBIDDEN response.
2. If the token server did not respond to the authorization code exchange the request will result in a 404 NOT FOUND response.
   The redirect server returns the token information, removes the token info and marks the registration record as redeemed.

(1) eXist-db redirect URI example: `http://redirect.mydomain.com:8080/exist/restxq/oauth-interceptor`

- Client Id and Client Secret are stored in the application binary.
- Client Id and Client Secret are temporarily stored on the redirect server.
  (see Obtain Authorization Server slide, step 1 and
  Exchange Authorization Code For Access Token, step 3)