Identity & Access Control for modern Applications

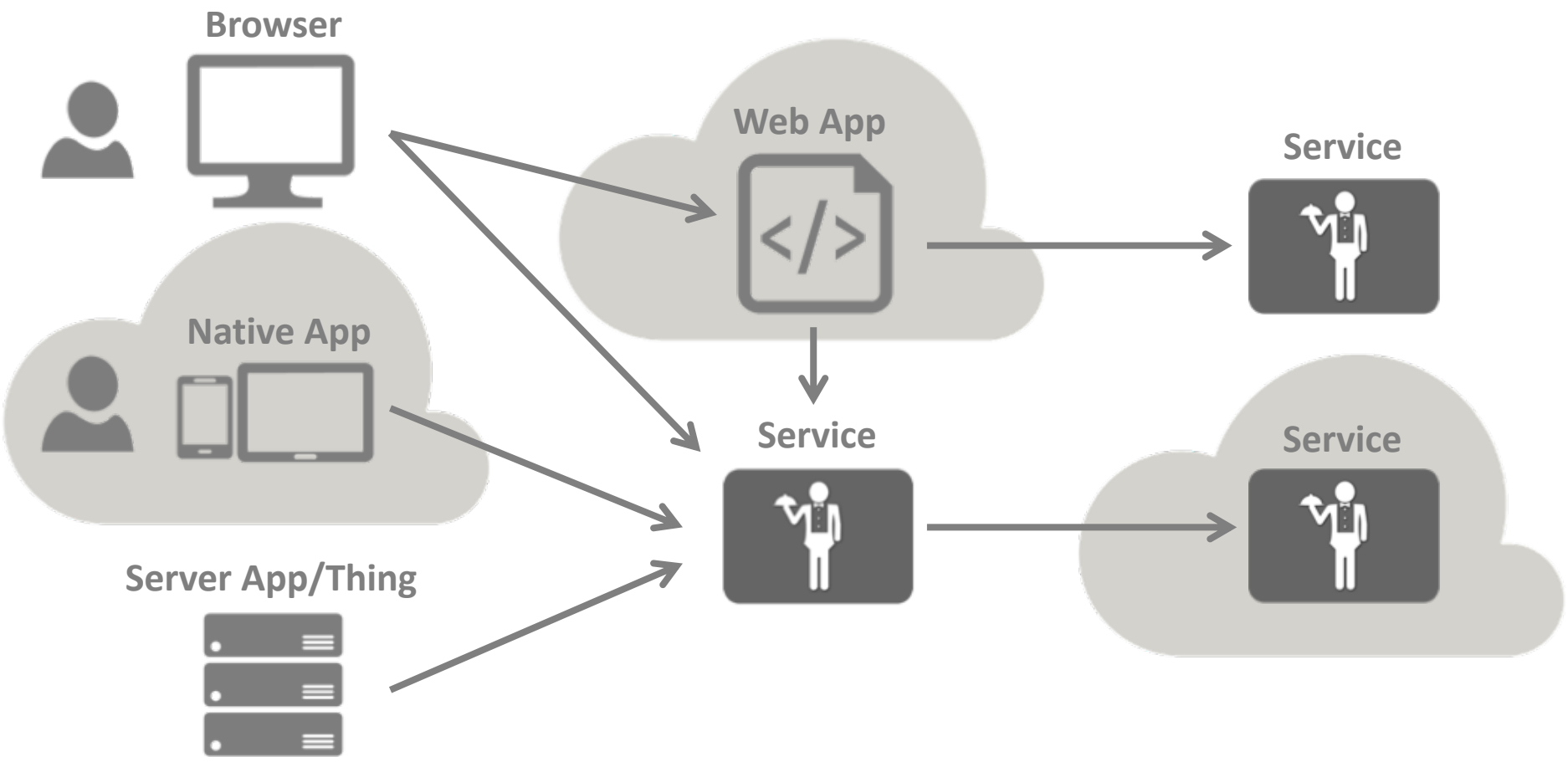
Dominick Baier
http://leastprivilege.com
@leastprivilege

Brock Allen
http://brockallen.com
@brocklallen

identity SERVER

PolicyServer

Modern Application Architecture



Agenda

- **Part 1: ASP.NET Core & Authentication**
  - ASP.NET Core security architecture
  - Authorization
  - Externalizing authentication
  - OpenID Connect
  - Patterns

Agenda

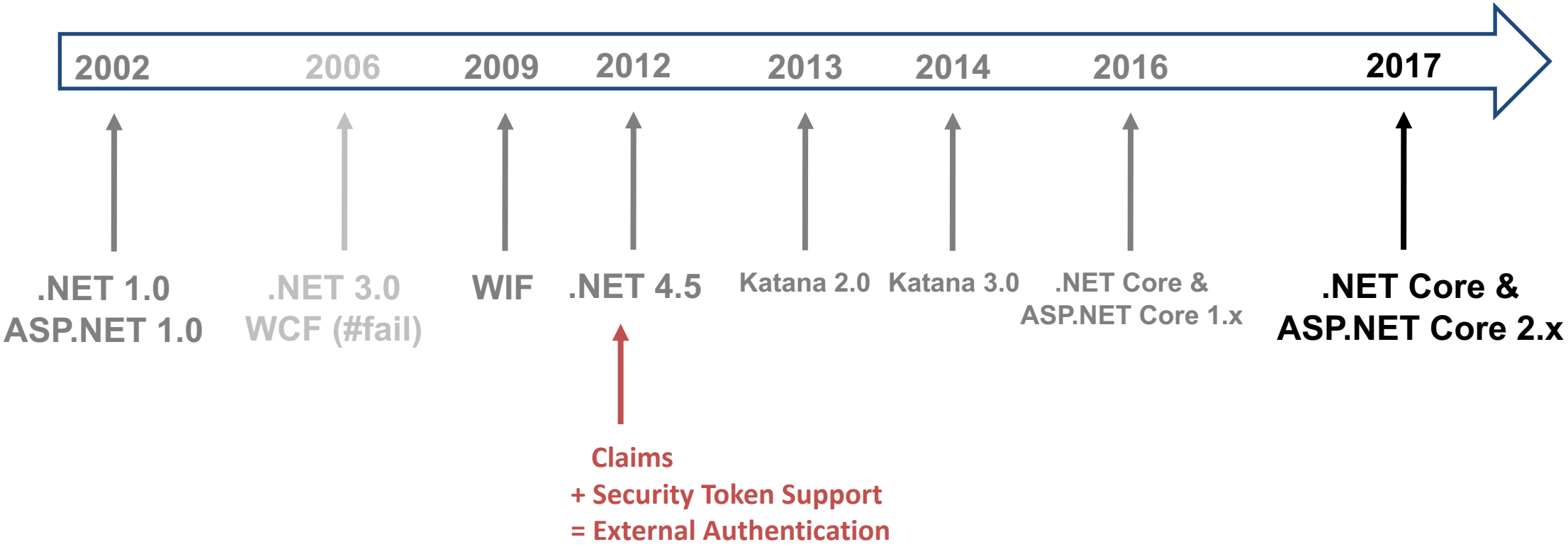
- **Part 2: Web APIs & Access Control**
  - OAuth 2.0 (& combining it with OpenID Connect)
  - Application Scenarios
    - Server to Server Communication
    - Native/Mobile Applications
    - JavaScript Applications

Initial Design

```
public interface IIdentity
{
  bool IsAuthenticated { get; }
  string AuthenticationType { get; }
  string Name { get; }
}
```

```
public interface IPrincipal
{
  IIdentity Identity { get; }
  bool IsInRole(string roleName);
}
```

Timeline


2002
2006
2009
2012
2013
2014
2016
2017
.NET 1.0
ASP.NET 1.0
.NET 3.0
WCF (#fail)
WIF
.NET 4.5
Katana 2.0
Katana 3.0
.NET Core &
ASP.NET Core 1.x
.NET Core &
ASP.NET Core 2.x
Claims
+ Security Token Support
= External Authentication

Claims

- **More flexible way to model identity data**
  - keys and values
  - concept of an issuer

- **Claim examples**
  - Bob is an administrator
  - Jim‘s email address is jim@foo.com
  - Alice's user id is #123
  - Alice works in the sales department

Claim

- **Statement about an entity made by someone else**

```
public class Claim
{
    public virtual string Type { get; }
    public virtual string Value { get; }
    public virtual string Issuer { get; }

    // rest omitted
}
```

ClaimsPrincipal & ClaimsIdentity

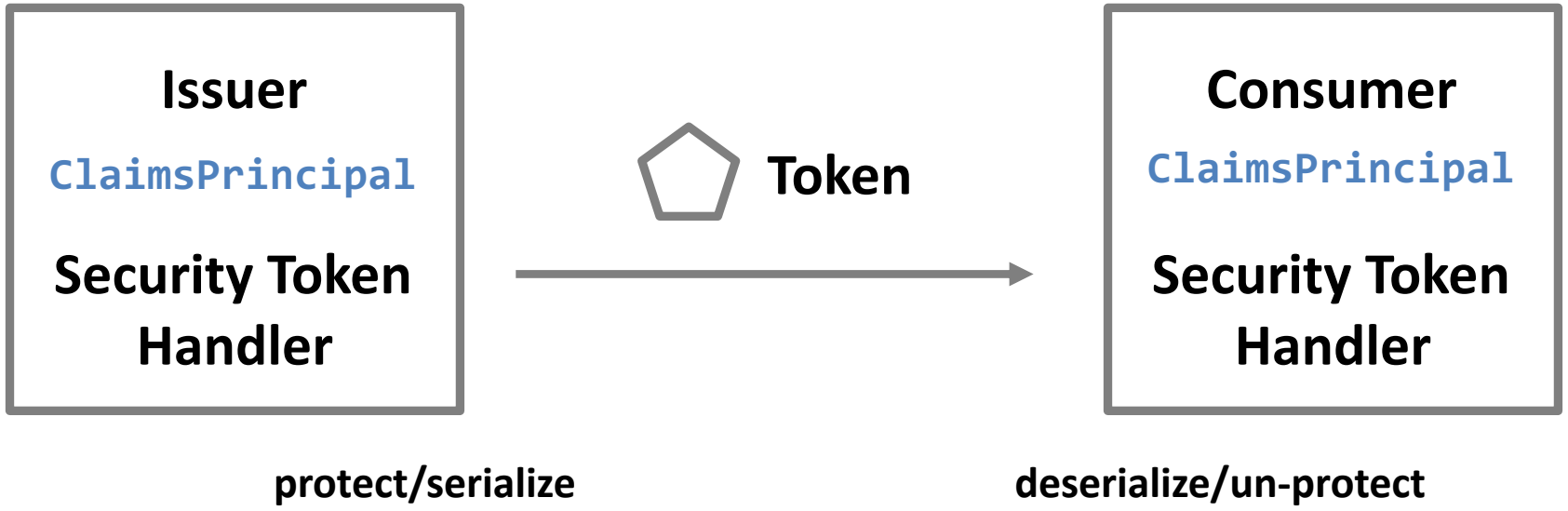
```
interface IIdentity
{
  bool IsAuthenticated      { get; }
  string AuthenticationType { get; }
  string Name               { get; }
}
```

```
interface IPrincipal
{
  IIdentity Identity { get; }
  bool IsInRole(string roleName);
}
```

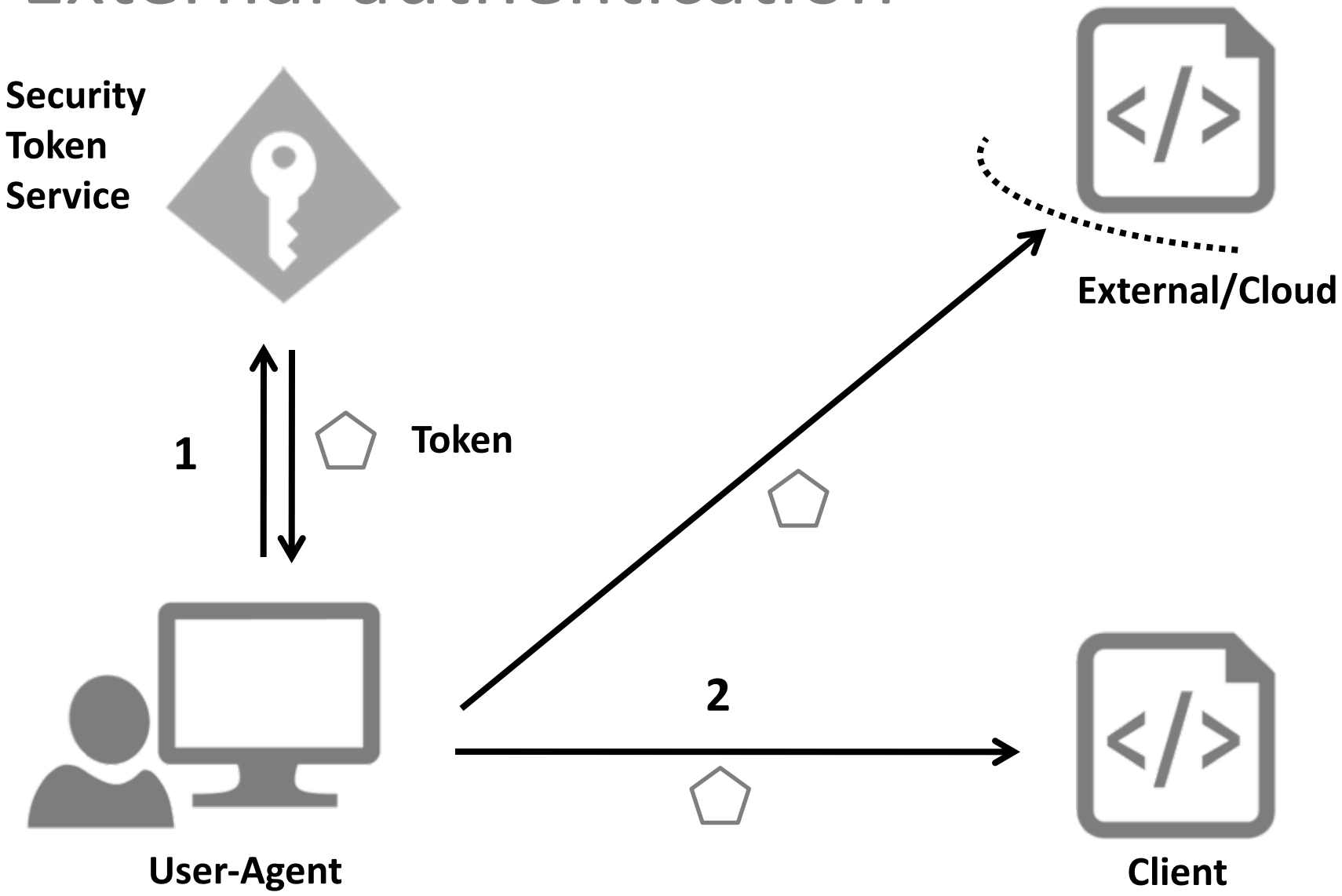
```
public class ClaimsIdentity : IIdentity
{
  IEnumerable<Claim> Claims { get; }
}
```

```
public class ClaimsPrincipal : IPrincipal
{
  ReadOnlyCollection<ClaimsIdentity> Identities { get; }
}
```

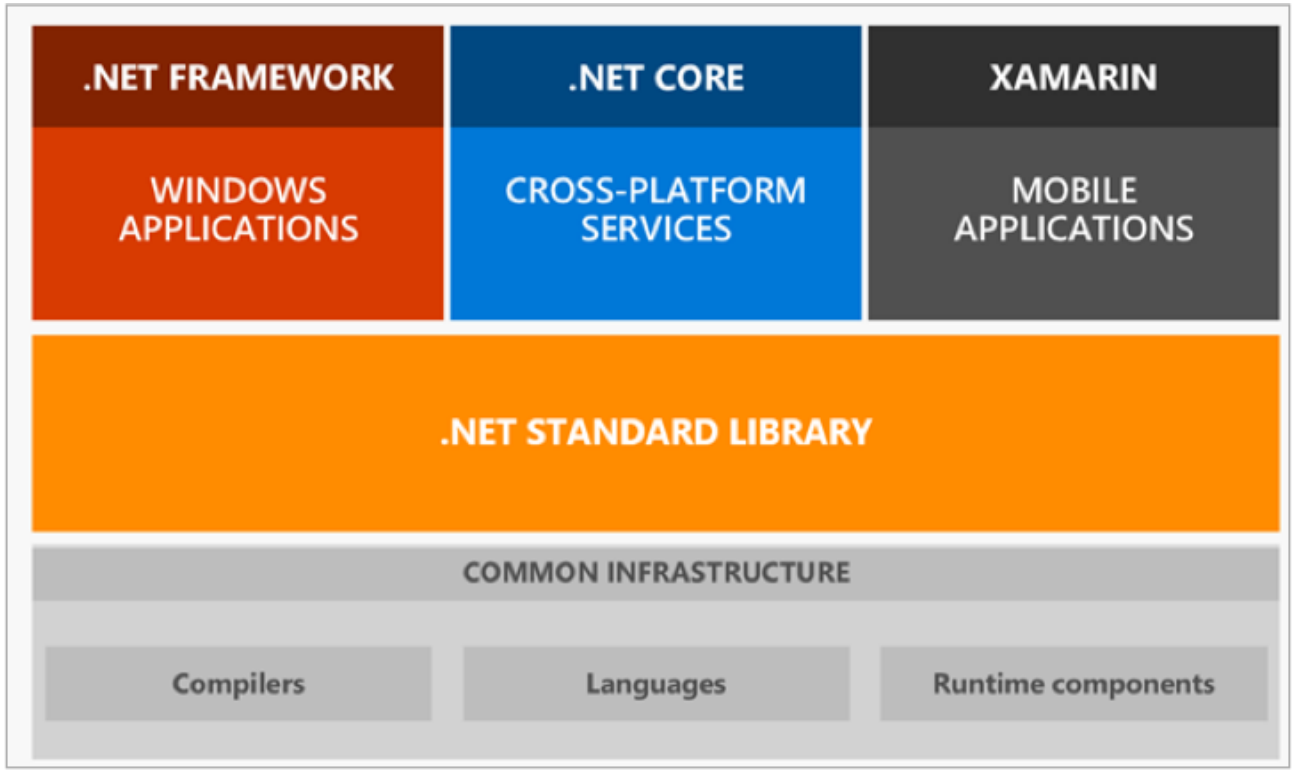
Security Tokens



External authentication



The new .NET

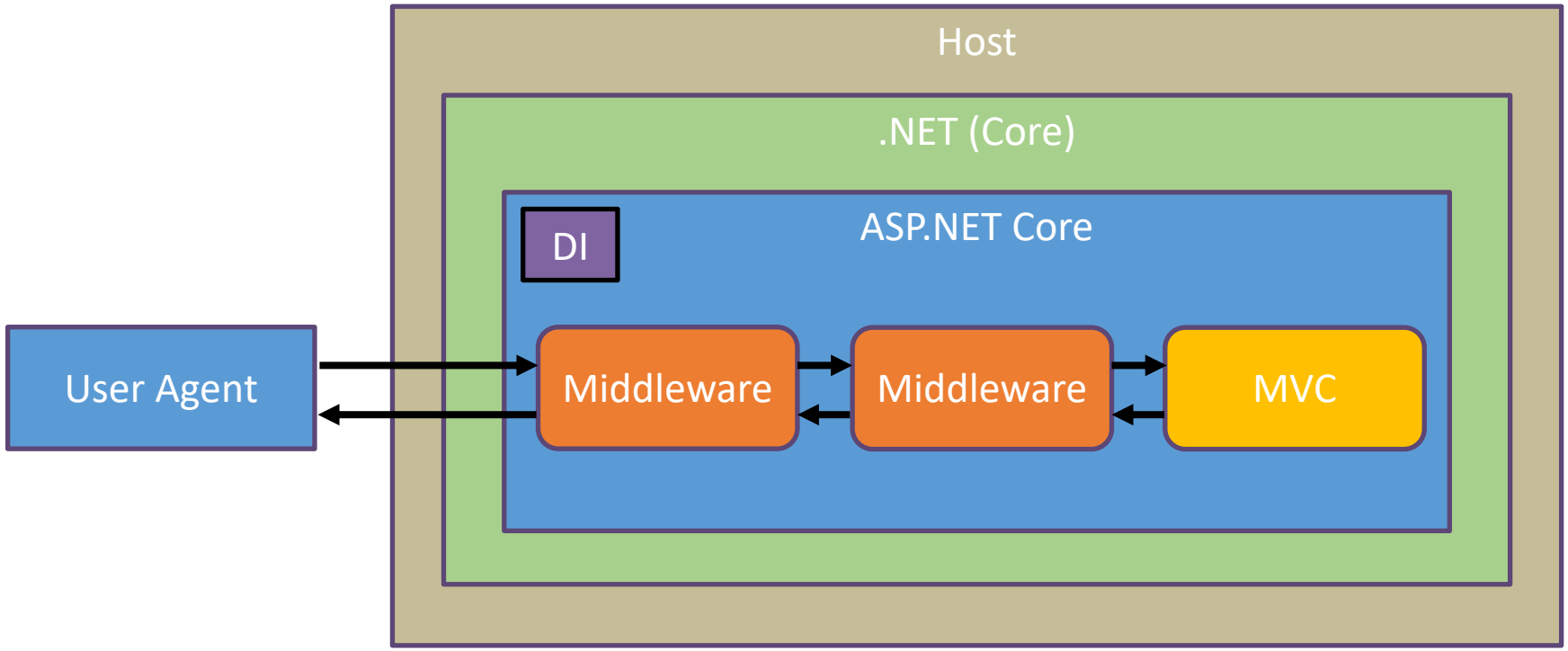

.NET FRAMEWORK
.NET CORE
XAMARIN
WINDOWS APPLICATIONS
CROSS-PLATFORM SERVICES
MOBILE APPLICATIONS
.NET STANDARD LIBRARY
COMMON INFRASTRUCTURE
Compilers
Languages
Runtime components

What is ASP.NET Core?

- **Microsoft's new web framework**
  - Runs on .NET Core and the full .NET Framework
- **Middleware-based pipeline architecture**
  - Components that provide services for web applications
  - Many features packaged as middleware
- **Familiar HttpContext programming model**
  - But all new
- **Hosting is provided by Kestrel (by default)**
  - HTTP.SYS as a Windows-specific alternative

ASP.NET Core Architecture

- **ASP.NET Core is the runtime (hosted by .NET Core)**
- **MVC is Microsoft's primary application framework**
  - combines web UI & API



Loading ASP.NET Core

```
public class Program
{
    public static void Main(string[] args)
    {
        BuildWebHost(args).Run();
    }

    public static IWebHost BuildWebHost(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>()
            .Build();
}
```

Default Web Host

- **Convenience method for setting up a default host**
  - Reads hosting environment and URLs from environment variables
  - Sets up Kestrel with IIS integration
  - Set up configuration infrastructure
    - appsettings.json / appsettings.{environment}.json / environment variables
  - Sets up default logging
    - debug and console
  - Sets up user secrets
  - Sets up a developer exception page when environment is set to 'Development'

- **Can be customized**

Authentication in ASP.NET Core

- **Combination of middleware and authentication handlers in DI**
  - middleware invokes handlers for request related processing
  - handlers can be also invoked manually

- **Handlers implement specific authentication methods**
  - Cookies for browser based authentication
  - Google, Facebook, and other social authentication
  - OpenId Connect for external authentication
  - JSON web token (JWT) for token-based authentication

Interacting with the authentication system

- **Extension methods on *HttpContext* call the *IAuthenticationService* in DI**

```
public static class AuthenticationHttpContextExtensions
{
    public static Task SignInAsync(this HttpContext context, ClaimsPrincipal principal) { }
    public static Task SignInAsync(this HttpContext context, string scheme, ClaimsPrincipal principal) { }

    public static Task SignOutAsync(this HttpContext context) { }
    public static Task SignOutAsync(this HttpContext context, string scheme) { }

    public static Task ChallengeAsync(this HttpContext context) { }
    public static Task ChallengeAsync(this HttpContext context, string scheme) { }

    public static Task ForbidAsync(this HttpContext context) { }
    public static Task ForbidAsync(this HttpContext context, string scheme) { }

    public static Task<AuthenticateResult> AuthenticateAsync(this HttpContext context) { }
    public static Task<AuthenticateResult> AuthenticateAsync(this HttpContext context, string scheme) { }
}
```

Setting up authentication

- **Global settings go into DI**
  - e.g. default schemes
- **Authentication middleware invokes handlers**

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(options =>
    {
        options.DefaultScheme = "Cookies";
    });
}

public void Configure(IApplicationBuilder app)
{
    app.UseAuthentication();
}
```

Setting up authentication (2)

- **Scheme settings can be more fine-grained**

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme = "...";

        options.DefaultSignInScheme = "...";
        options.DefaultSignOutScheme = "...";

        options.DefaultChallengeScheme = "...";
        options.DefaultForbidScheme = "...";

    });
}
```

Cookie Authentication

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(defaultScheme: "Cookies")
        .AddCookie("Cookies", options =>
        {
            options.LoginPath = "/account/login";
            options.AccessDeniedPath = "/account/denied";

            options.Cookie.Name = "myapp";
            options.Cookie.Expiration = TimeSpan.FromHours(8);
            options.SlidingExpiration = false;
        });
}
```

Cookies: Logging in

- **SignInAsync issues cookie**
  - either using a named scheme, or default

```
var claims = new Claim[]
{
    new Claim("sub", "37734"),
    new Claim("name", "Brock Allen")
};

var ci = new ClaimsIdentity(claims, "password", "name", "role");
var cp = new ClaimsPrincipal(ci);

await HttpContext.SignInAsync(cp);
```

Cookies: Logging out

- **SignOutAsync removes cookie**

```
await HttpContext.SignOutAsync();
```

Claims Transformation

- **Per-request manipulation of principal & claims**
  - register an instance of *IClaimsTransformation* in DI
  - gets called from the handler's *AuthenticateAsync* method

```
public class ClaimsTransformer : IClaimsTransformation
{
    public async Task<ClaimsPrincipal> TransformAsync(ClaimsPrincipal principal)
    {
        return await CreateApplicationPrincipalAsync(principal);
    }
}
```

```
services.AddTransient<IClaimsTransformation, ClaimsTransformer>();
```

Data Protection

- **Used to protect cookies and other secrets**
  - *IDataProtection*Provider in DI
- **Uses a key container file**
  - stored outside of application directory*
  - uses a key ring with automatic rotation
  - keys *should* be protected

- **Needs to be synchronized between nodes in a farm**

* https://docs.microsoft.com/en-us/aspnet/core/security/data-protection/configuration/default-settings

Authorization

- **Complete re-write**
  - better separation of business code and authorization logic
  - policy based authorization
  - resource/action based authorization
  - DI enabled

ASP.NET 4.x version: https://github.com/DavidParks8/Owin-Authorization

[Authorize]

- **Similar syntax**
  - roles still supported

```
[Authorize]
public class HomeController : Controller
{
    [AllowAnonymous]
    public IActionResult Index()
    {
        return View();
    }

    [Authorize(Roles = "Sales")]
    public IActionResult About()
    {
        return View(User);
    }
}
```

Authorization policies

Startup

```
services.AddAuthorization(options =>
{
    options.AddPolicy("ManageCustomers", policy =>
    {
        policy.RequireAuthenticatedUser();
        policy.RequireClaim("department", "sales");
        policy.RequireClaim("status", "senior");
    });
});
```

Controller

```
[Authorize("ManageCustomers")]
public IActionResult Manage()
{
    // stuff
}
```

Programmatically using policies

```
public class CustomerController : Controller
{
    private readonly IAuthorizationService _authz;

    public CustomerController(IAuthorizationService authz)
    {
        _authz = authz;
    }

    public async Task<IActionResult> Manage()
    {
        var result = await _authz.AuthorizeAsync(User, "ManageCustomers");
        if (result.Succeeded) return View();

        return Forbid();
    }
}
```

…or from a View

```
@using Microsoft.AspNetCore.Authorization
@inject IAuthorizationService _authz

@if ((await _authz.AuthorizeAsync(User, "ManageCustomers")).Succeeded)
{
    <div>
        <a href="/customers/test">Manage</a>
    </div>
}
```

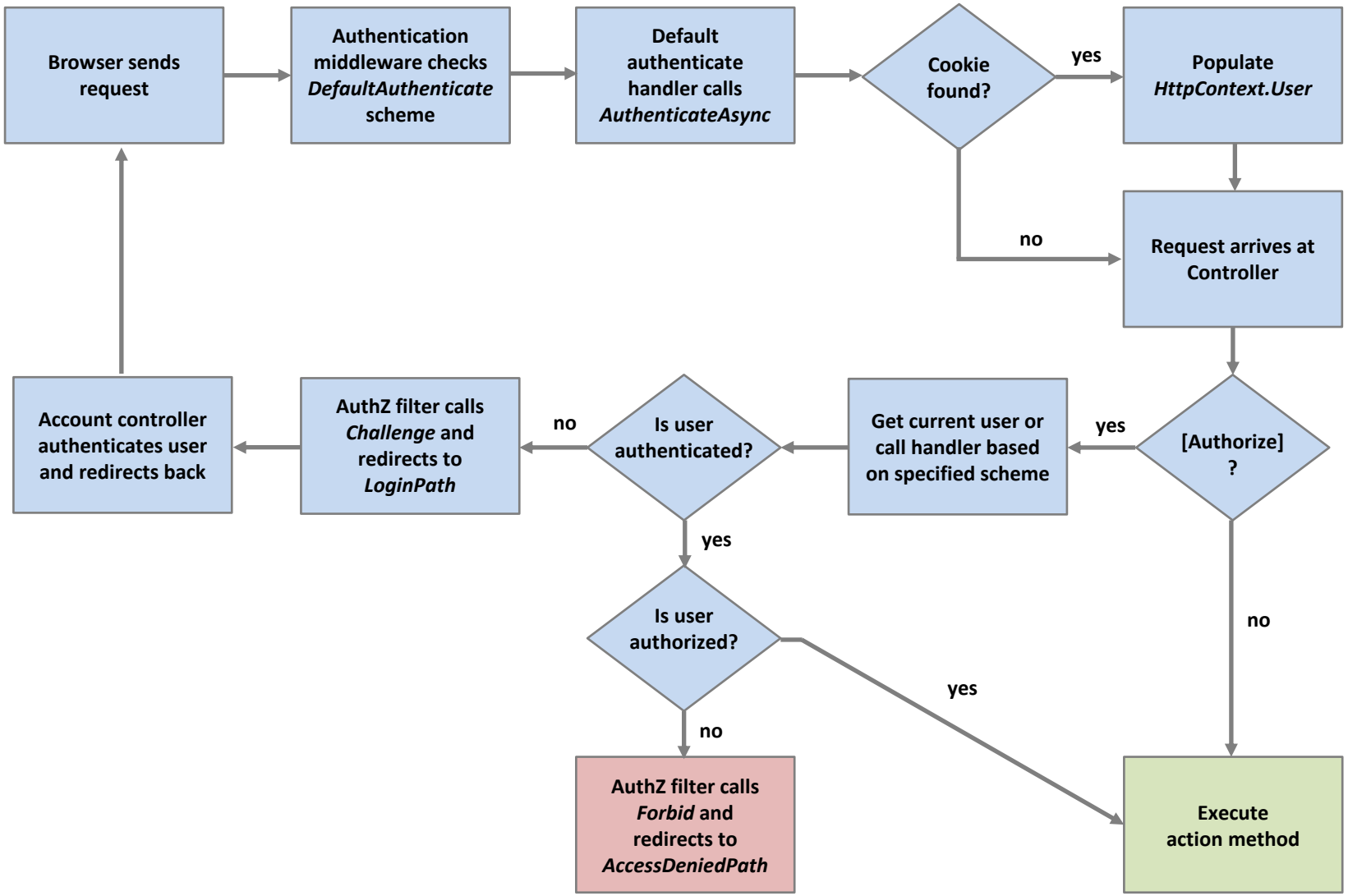
Custom Requirements

```
public class JobLevelRequirement : IAuthorizationRequirement
{
    public JobLevel Level { get; }

    public JobLevelRequirement(JobLevel level)
    {
        Level = level;
    }
}

public static class StatusPolicyBuilderExtensions
{
    public static AuthorizationPolicyBuilder RequireJobLevel(
      this AuthorizationPolicyBuilder builder, JobLevel level)
    {
        builder.AddRequirements(new JobLevelRequirement(level));
        return builder;
    }
}
```

Handling Requirements

```
public class JobLevelRequirementHandler : AuthorizationHandler<JobLevelRequirement>
{
    private readonly IOrganizationService _service;

    public JobLevelRequirementHandler(IOrganizationService service)
    {
        _service = service;
    }

    protected override void Handle(
        AuthorizationContext context, JobLevelRequirement requirement)
    {
        var currentLevel = _service.GetJobLevel(context.User);

        if (currentLevel == requirement.Level)
        {
            context.Succeed(requirement);
        }
    }
}
```

Summary: Cookies & Authorization



External Authentication

- **ASP.NET Core supports**
  - Google, Twitter, Facebook, Microsoft Account
  - OpenID Connect & JSON Web Tokens

- **New generic OAuth 2.0 handler makes integration with other proprietary providers easier**
  - LinkedIn, Slack, Spotify, WordPress, Yahoo, Github, Instragram, BattleNet, Dropbox, Paypal, Vimeo…

https://github.com/aspnet-contrib/AspNet.Security.OAuth.Providers

Social Identity Providers

- **Enabled with *AddGoogle*, et al.**
  - Rely upon cookie authentication handler for sign-in

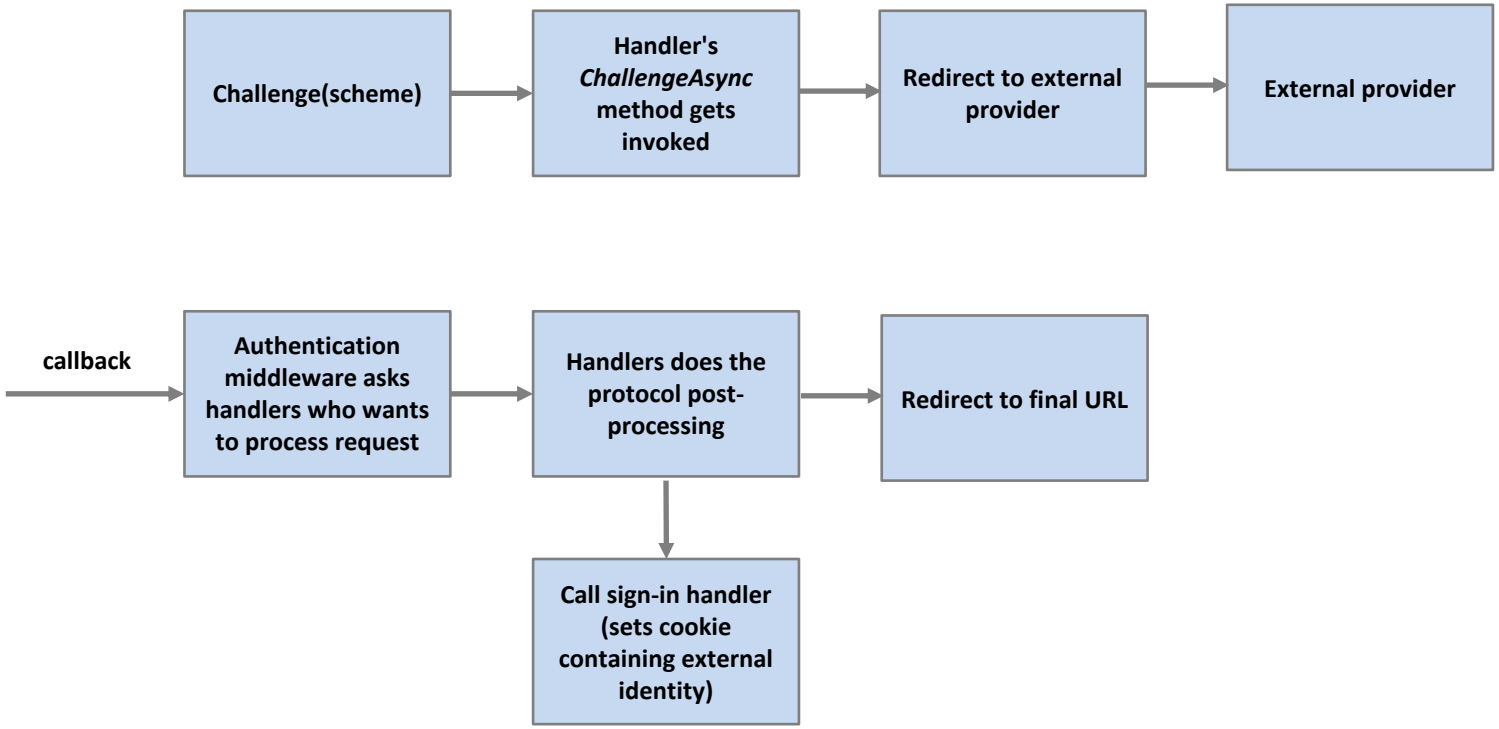
```
services.AddAuthentication("Cookies")
    .AddCookie("Cookies", options =>
    {
        options.LoginPath = "/account/login";
        options.AccessDeniedPath = "/account/denied";
    })
    .AddGoogle("Google", options =>
    {
        options.ClientId = "...";
        options.ClientSecret = "...";
    });
```

Social Identity Providers

- **Challenge triggers redirect for login**
  - Control URL user returns to and state with *AuthenticationProperties*
  - MVC *ChallengeResult* works with action result architecture

```
var props = new AuthenticationProperties
{
    RedirectUri = "/Home/Secure"
};
await HttpContext.ChallengeAsync("Google", props);

// or if using MVC:

return Challenge("Google", props);
```

Summary: External Authentication



External authentication with Callback

- **Add application level post-processing step**
  - provision logic, extra UI etc..
- **Second cookie handler to temporarily store external identity**

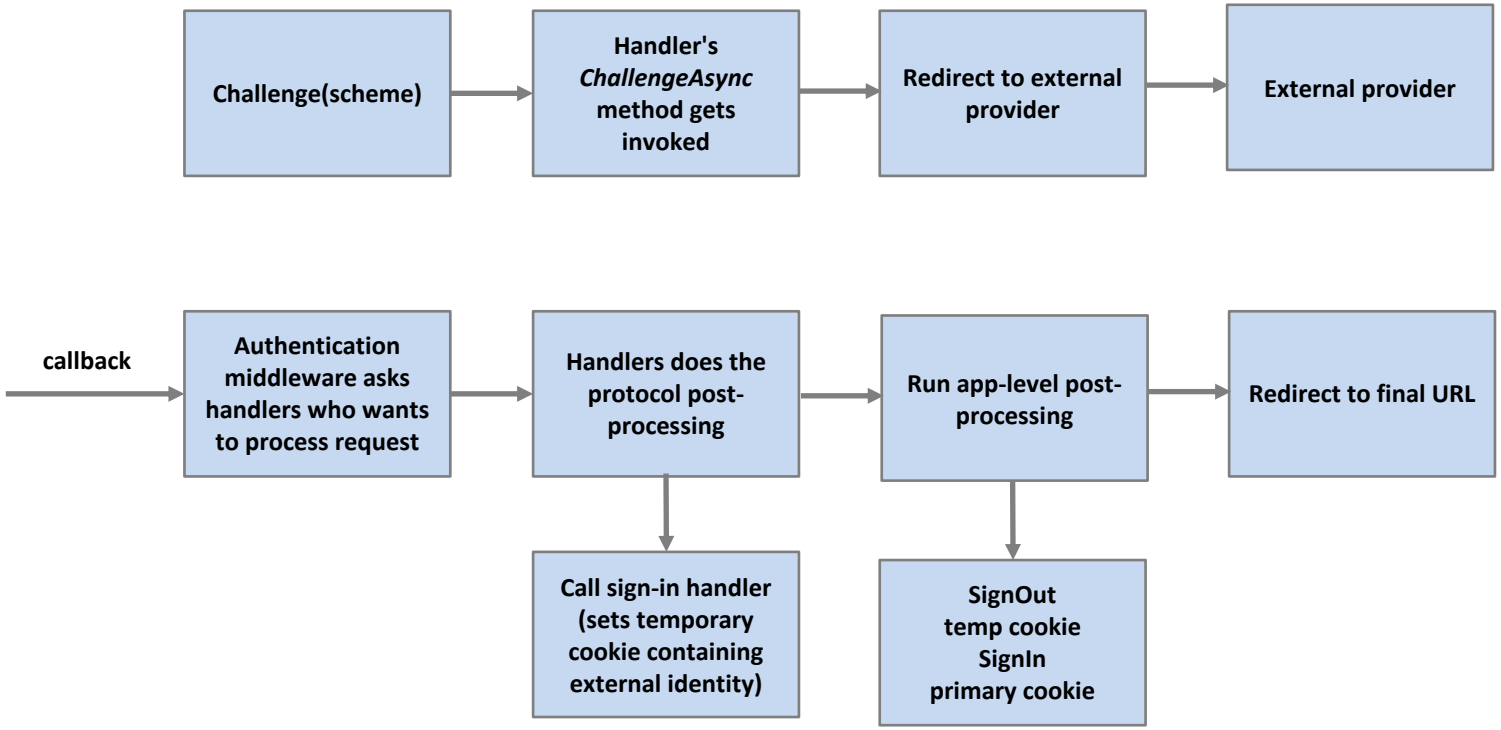
```
services.AddAuthentication("Cookies")
    .AddCookie("Cookies")
    .AddCookie("Temp")

    .AddGoogle("Google", options =>
    {
        options.SignInScheme = "Temp";

        options.ClientId = "...";
        options.ClientSecret = "...";
    });
```

Mixing local and external Authentication

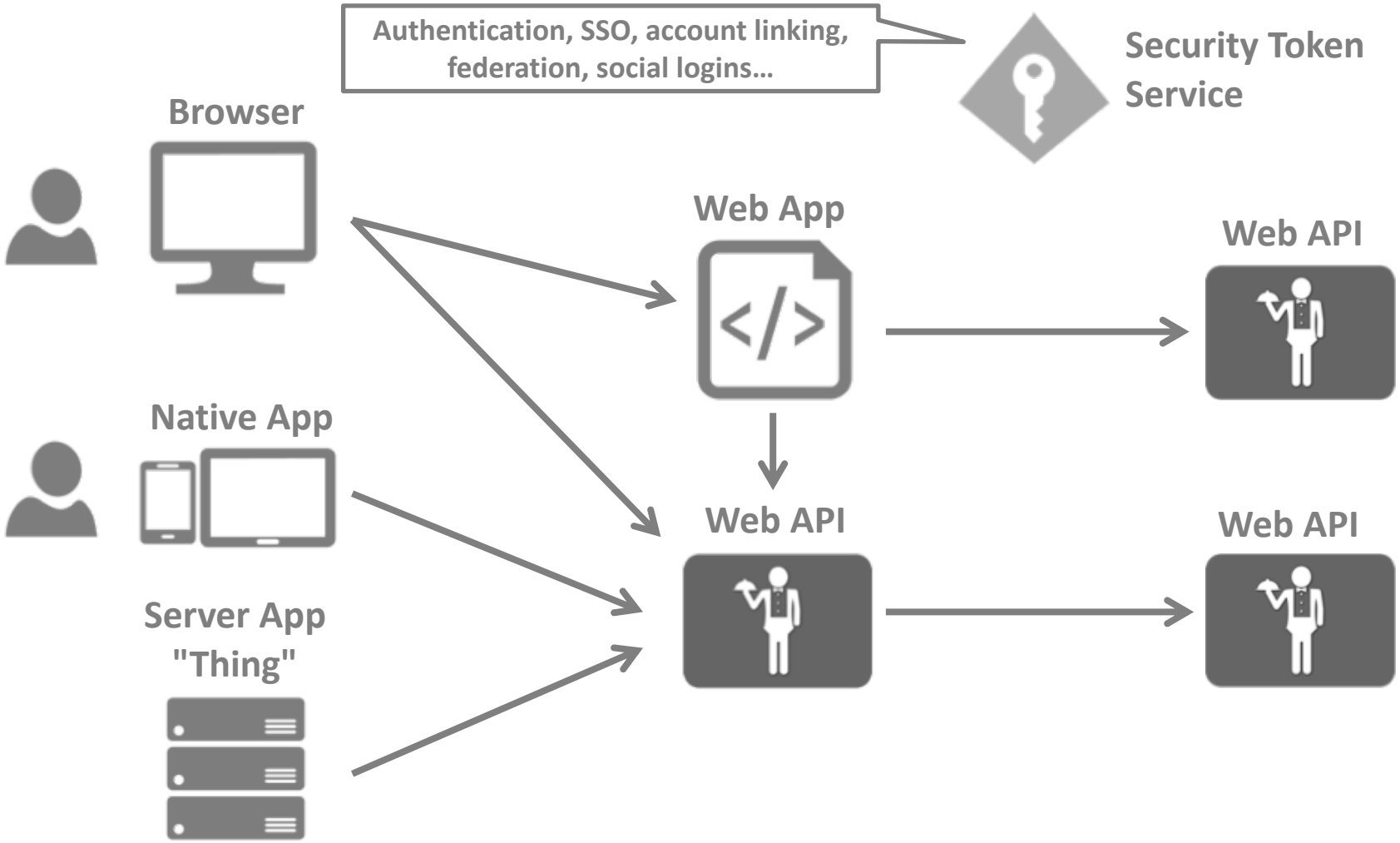
- **Redirect page performs post-processing logic**
  - *AuthenticateAsync* triggers temp cookie handler
  - Run post-processing logic / flow
  - Use primary cookie handler to log user in (and remove temp cookie)

```
var result = await HttpContext.AuthenticateAsync("Temp");

var userId = result.Principal.FindFirst(ClaimTypes.NameIdentifier);
var extProvider = userId.Issuer;

// post-processing workflow

var user = new ClaimsPrincipal(...);
await HttpContext.SignInAsync(user);
await HttpContext.SignOutAsync("Temp");
```

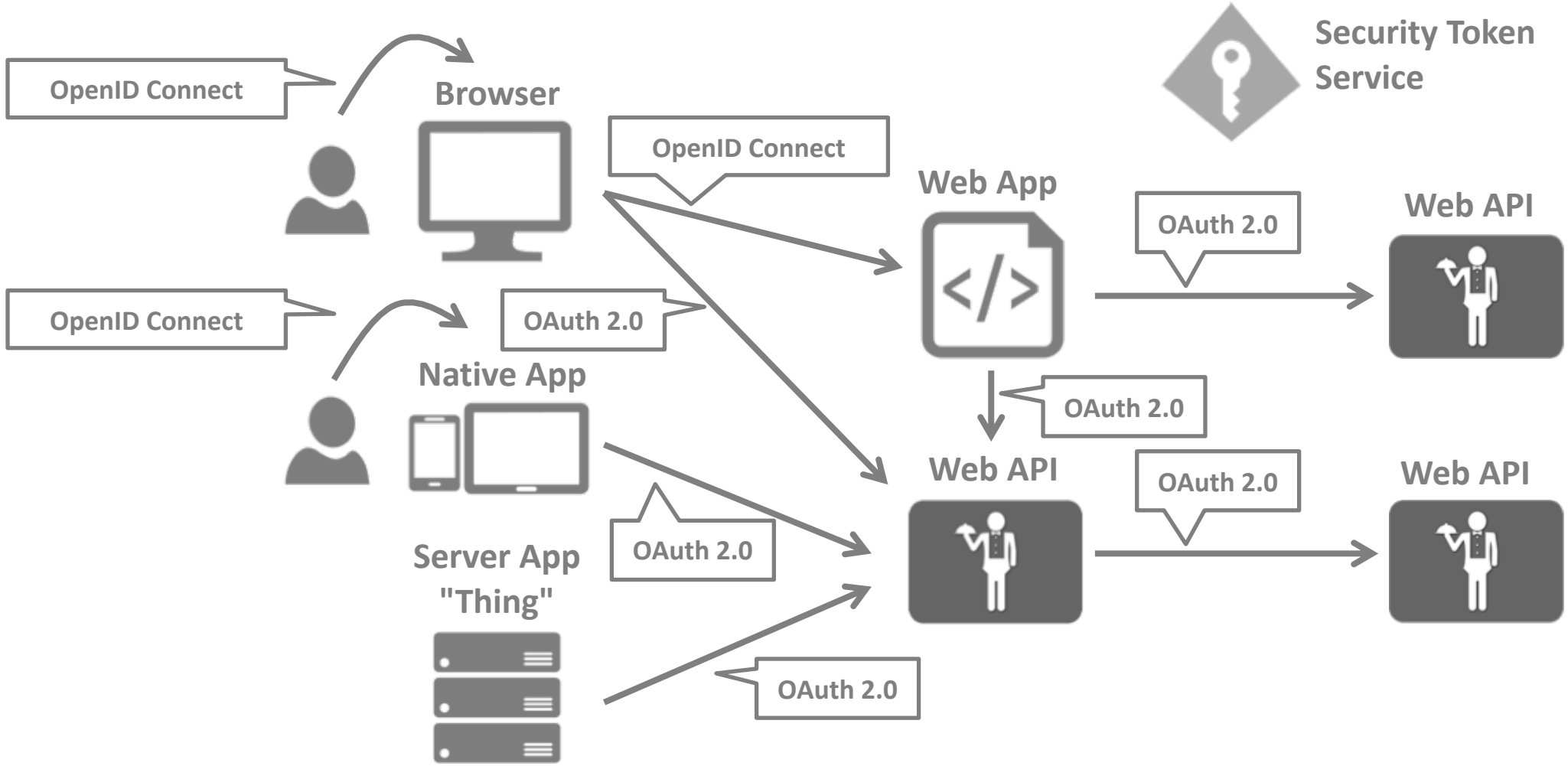
Summary: External Authentication with Callback



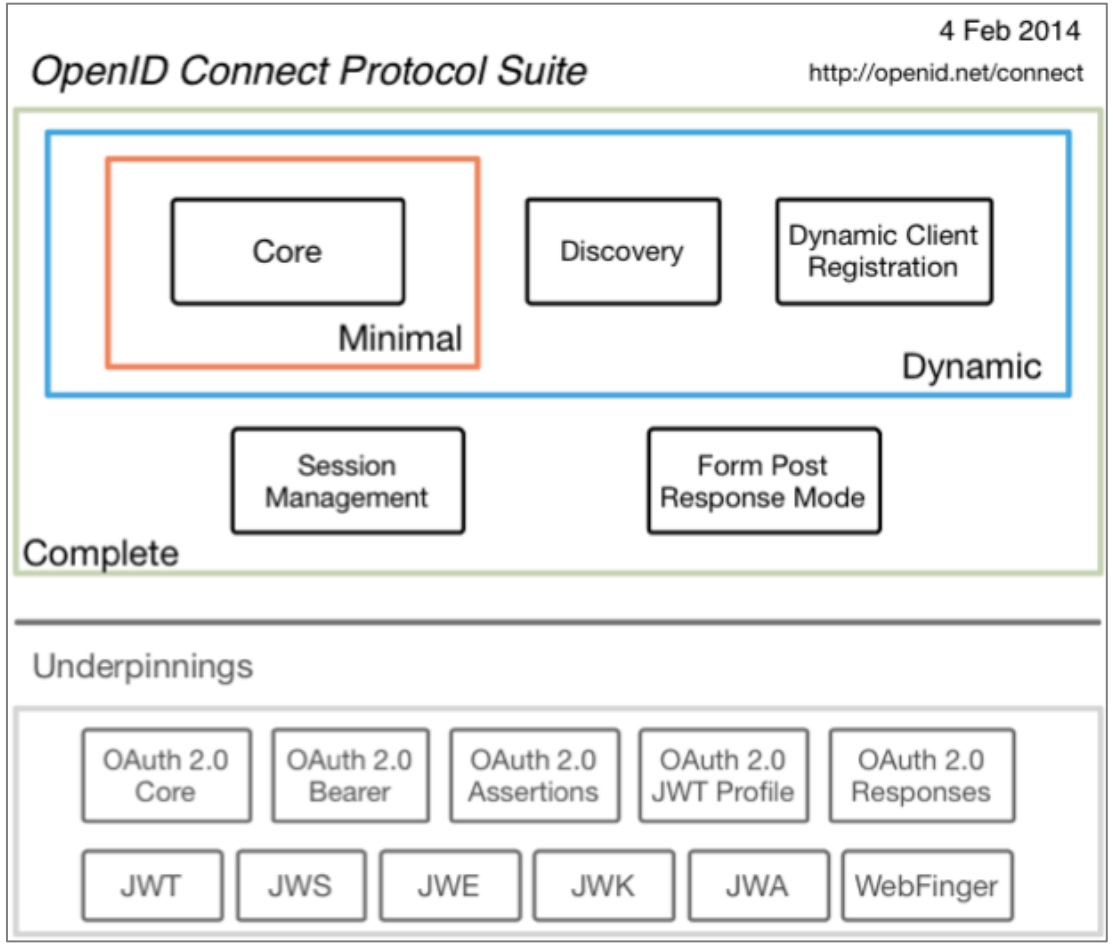
The way forward…



Security Protocols



http://openid.net/connect/



OpenID Connect Certification

for providers and
client libraries

OpenID Certification | OpenID

https://openid.net/certification/

OpenID Foundation · Current Working Groups · Specs & Dev Info · OpenID® Certification · OpenID Connect FAQ and Q&As

Certified OpenID Providers

These implementations have been granted certifications for these OpenID Provider conformance profiles:

Organization	Implementation	Basic OP	Implicit OP	Hybrid OP	Config OP	Dynamic OP
Auth0	Auth0	24-May-2016	15-Feb-2017	15-Feb-2017	24-May-2016	
Authlete	Authlete 1.1	12-Jul-2017	12-Jul-2017	12-Jul-2017	12-Jul-2017	
Dominick Baier & Brock Allen	IdentityServer3 v1.6	8-May-2015	8-May-2015	8-May-2015	8-May-2015	
Dominick Baier & Brock Allen	IdentityServer4	12-Dec-2016	12-Dec-2016	12-Dec-2016	12-Dec-2016	
CA	CA API Gateway/CA Mobile API Gateway	22-Jun-2017			22-Jun-2017	
CA	CA Single Sign-On 12.7	1-Mar-2017				
Clareity Security	Identity Provider v6.3.4	4-May-2016	23-Jun-2016	23-Jun-2016	23-Jun-2016	
ClassLink	ClassLink OneClick 2015	3-Nov-2015			3-Nov-2015	
Cloudentity	Cloudentity OIDC services 1.3	18-Aug-2017			18-Aug-2017	18-Aug-2017
Connect2id	Connect2id Server 6.1.2a	3-Jan-2017	3-Jan-2017	3-Jan-2017	3-Jan-2017	3-Jan-2017
CZ.NIC	mojeID	7-Jul-2016		31-Jul-2016	7-Jul-2016	7-Jul-2016
Deutsche Telekom	Telekom Login	29-Sep-2015			22-Sep-2015	
ForgeRock	OpenAM 13	13-Apr-2015	13-Apr-2015	13-Apr-2015	13-Apr-2015	
Google	Google Federated Identity	20-Apr-2015	21-Apr-2015	23-Apr-2015	15-Apr-2015	
Thierry Habart	SimpleIdentityServer V1.0.0	9-Dec-2015			11-Dec-2015	
Thierry Habart	SimpleIdentityServer V2.0.0	19-Jan-2016	19-Jan-2016	19-Jan-2016	19-Jan-2016	19-Jan-2016

IdentityServer

Dominick (Main)

GitHub, Inc. [US] https://github.com/IdentityServer

This organization Search Pull requests Issues Gist

IdentityServer

https://identityserver.io identity@leastprivilege.com

Repositories | People 6 | Teams 6 | Projects 0 | Settings

Pinned repositories

Customize pinned repositories

IdentityServer4
OpenID Connect and OAuth 2.0 Framework for ASP.NET Core
C# 995 303

IdentityServer4.AccessTokenValidation
IdentityServer Access Token Validation for ASP.NET Core
C# 50 38

IdentityServer4.Samples
Samples for IdentityServer4
JavaScript 213 200

IdentityServer3
OpenID Connect Provider and OAuth 2.0 Authorization Server Framework for ASP.NET 4.x/Katana
C# 1.8k 738

IdentityServer3.AccessTokenValidation
OWIN Middleware to validate access tokens from IdentityServer3
C# 57 70

IdentityServer3.Samples
Samples for IdentityServer v3
JavaScript 432 946

Search repositories... Type: All Language: All

IdentityServer4
OpenID Connect and OAuth 2.0 Framework for ASP.NET Core
security identity oauth2 dotnet aspnet-core openid-connect identityserver4
C# 995 303 Updated 14 hours ago

IdentityServer4.Quickstart.UI
Starter UI for in-memory IdentityServer4

Top languages
C# JavaScript CSS HTML

Most used topics Manage
aspnet-core dotnet identityserver4 oauth2 openid-connect

.NET foundation

OpenID CERTIFIED

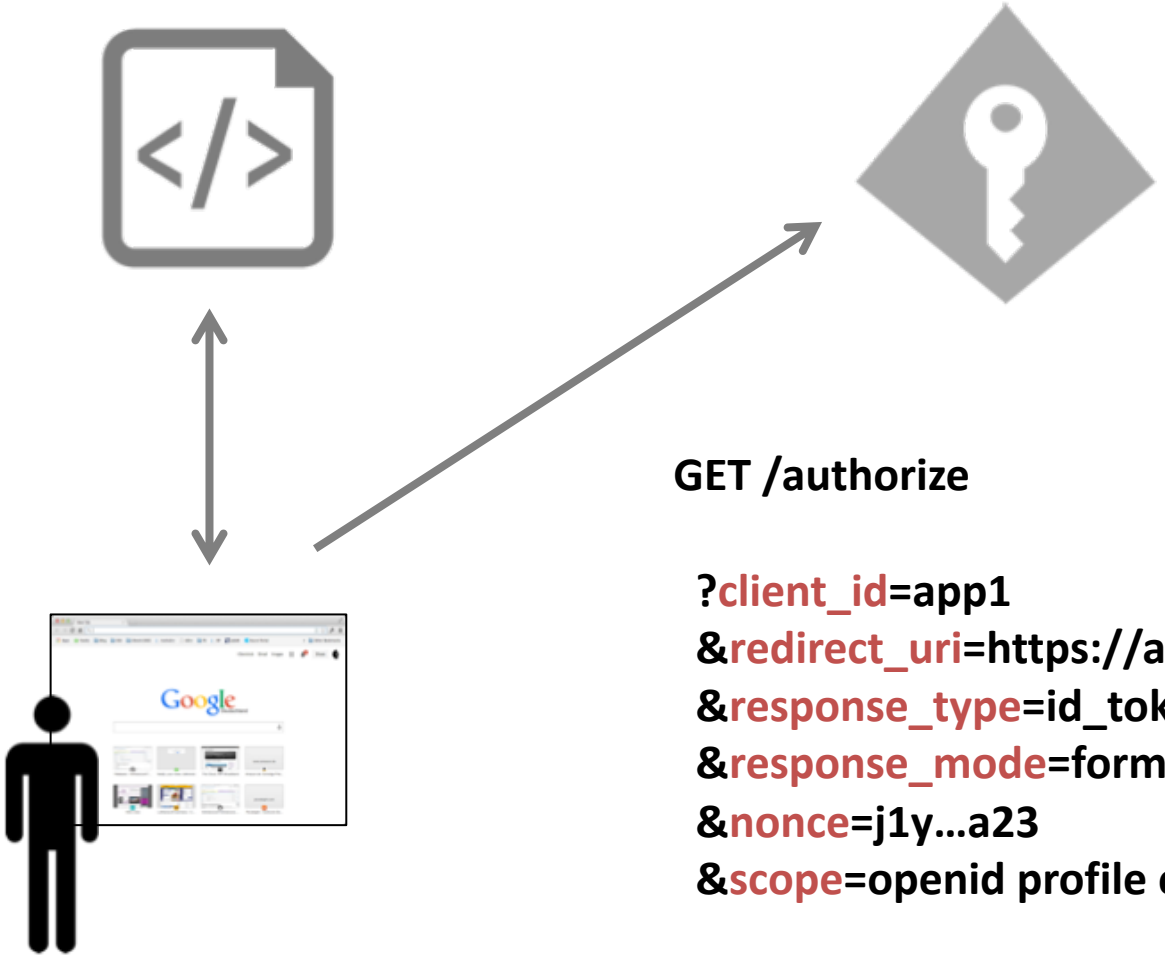
Endpoints

Discovery Endpoint

Authorize Endpoint

Token Endpoint

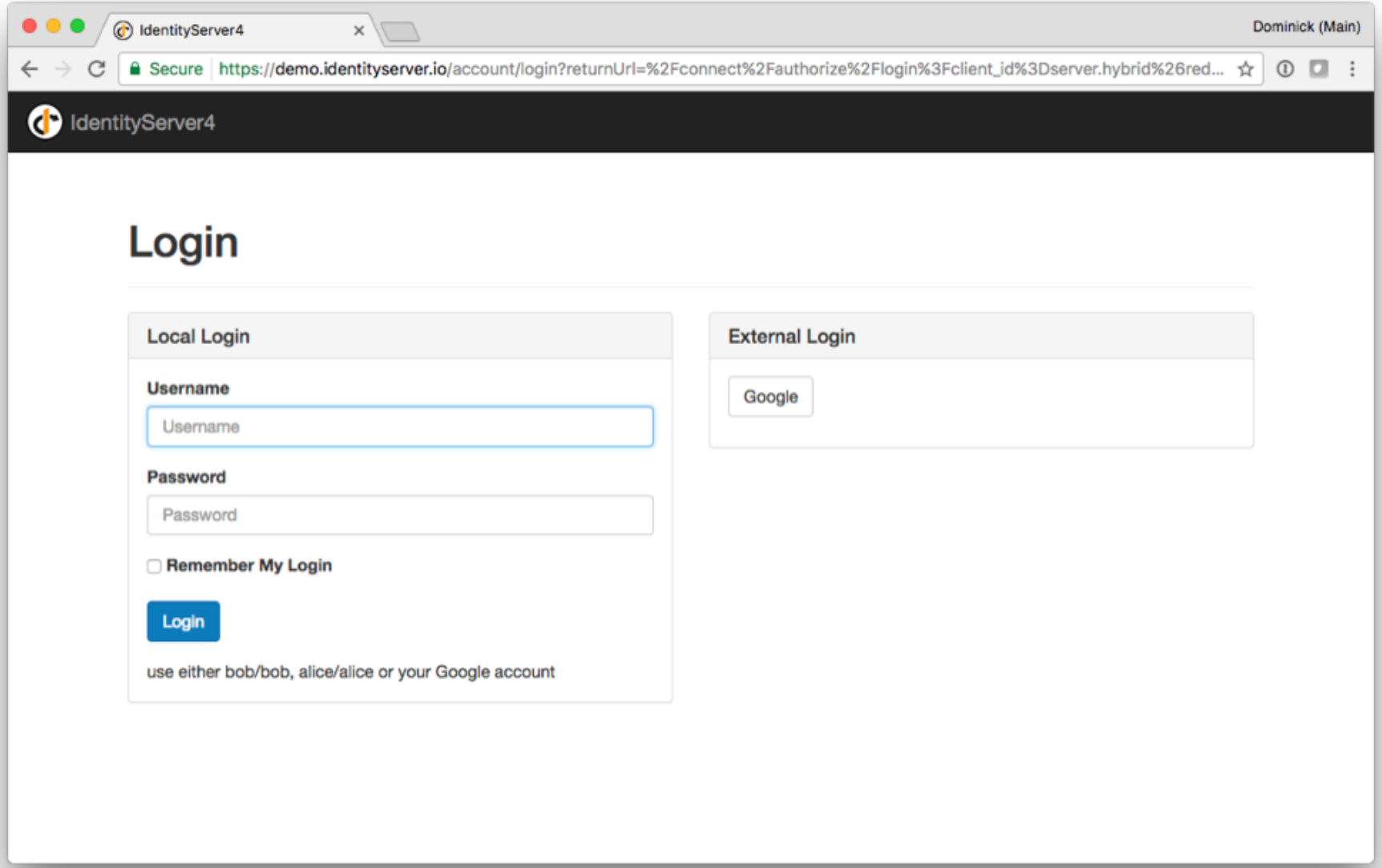
Authentication for Web Applications



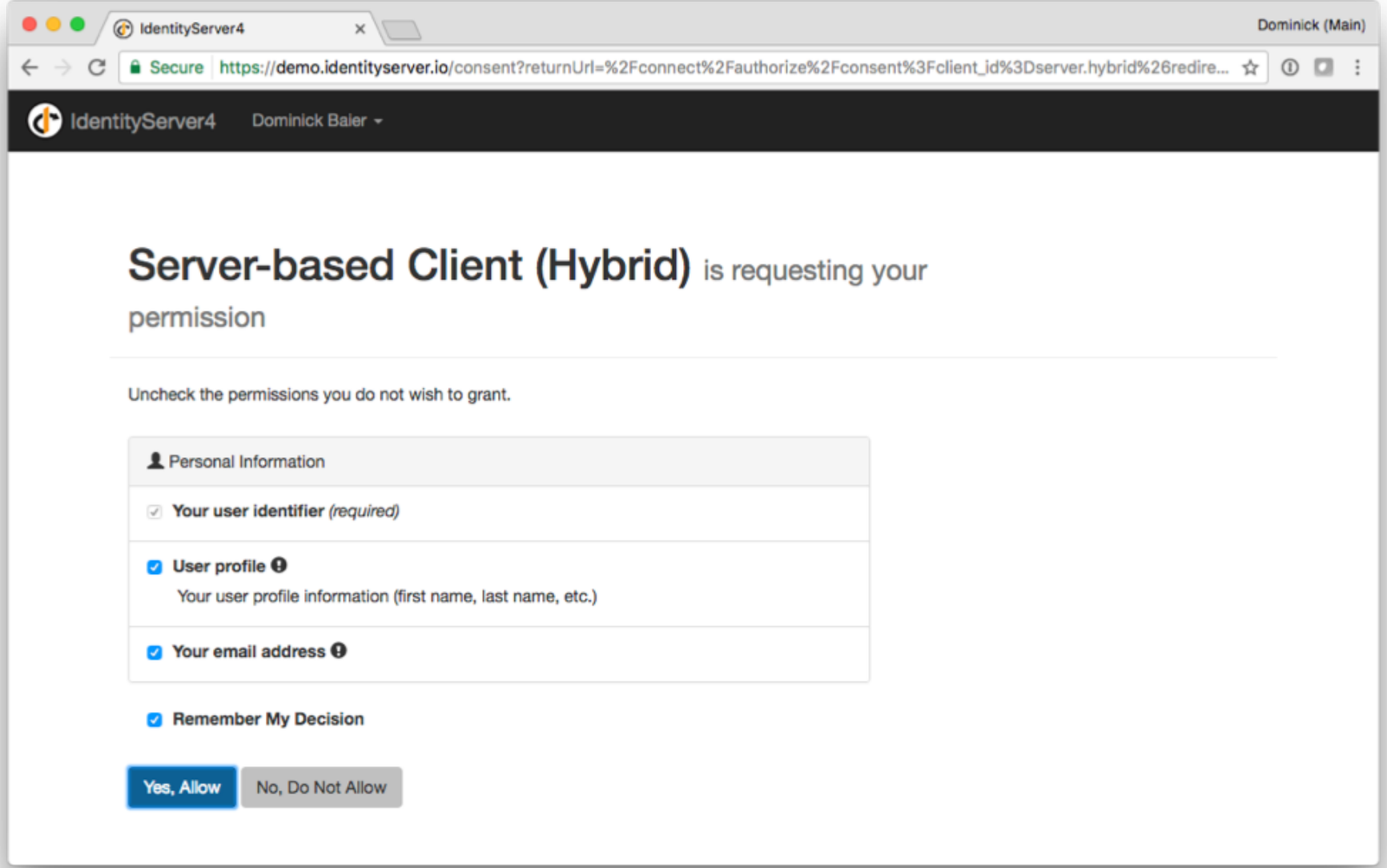
GET /authorize

?client_id=app1
&redirect_uri=https://app.com/cb
&response_type=id_token
&response_mode=form_post
&nonce=j1y…a23
&scope=openid profile email

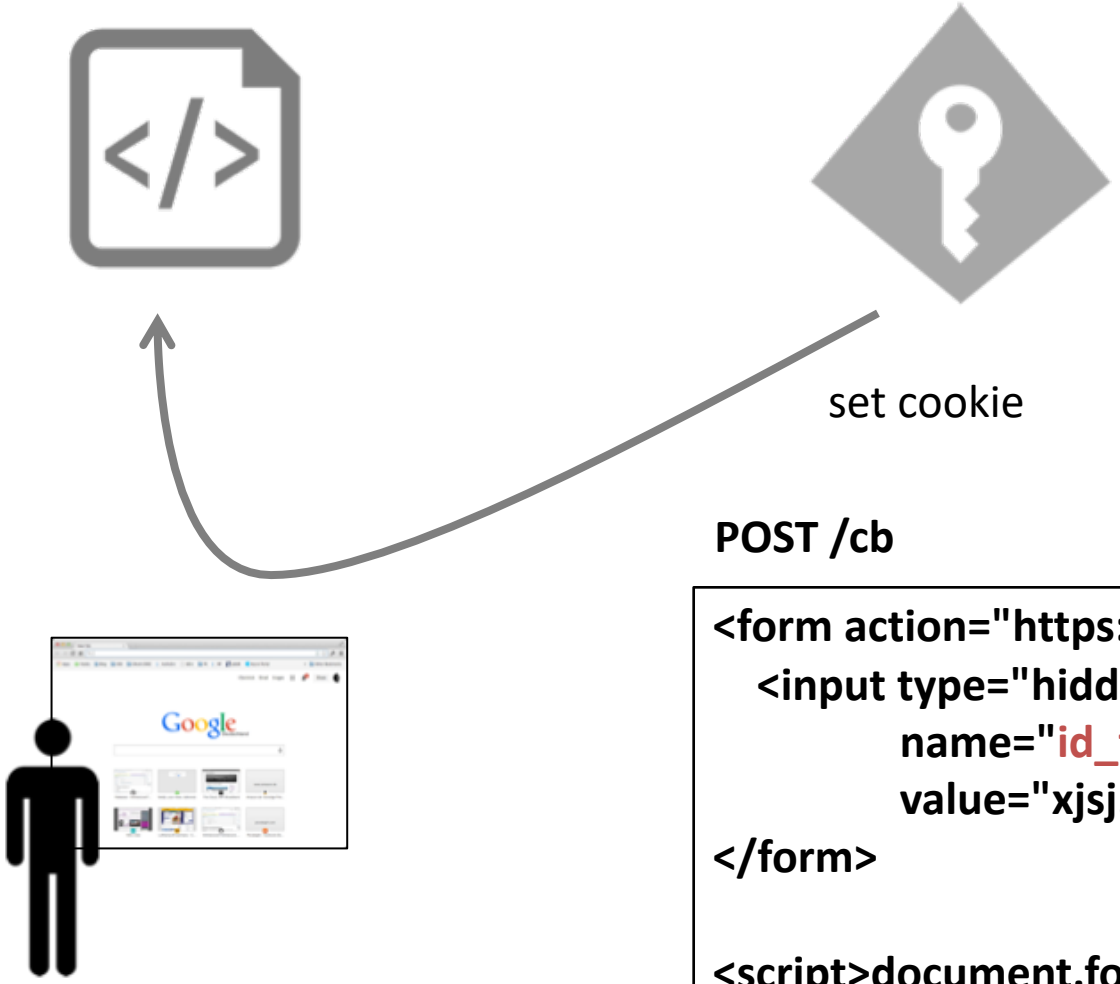
Authentication



Consent


IdentityServer4
Dominick (Main)
Secure
https://demo.identityserver.io/consent?returnUrl=%2Fconnect%2Fauthorize%2Fconsent%3Fclient_id%3Dserver.hybrid%26redire...
IdentityServer4
Dominick Baier
Server-based Client (Hybrid) is requesting your permission
Uncheck the permissions you do not wish to grant.
Personal Information
Your user identifier (required)
User profile
Your user profile information (first name, last name, etc.)
Your email address
Remember My Decision
Yes, Allow
No, Do Not Allow

Response



POST /cb

```
<form action="https://app.com/cb">
   <input type="hidden"
          name="id_token"
          value="xjsj…aas" />
</form>

<script>document.forms[0].submit()</script>
```

Identity Token

Header

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "mj399j…"
}
```

Payload

```
{
  "iss": "https://issuer",
  "exp": 1340819380,
  "iat": 1340818761,
  "aud": "app1",
  "nonce": "j1y…a23",
  "amr": [ "pwd" ],
  "auth_time": 12340819300

  "sub": "182jmm199",
  "name": "Alice",
}
```

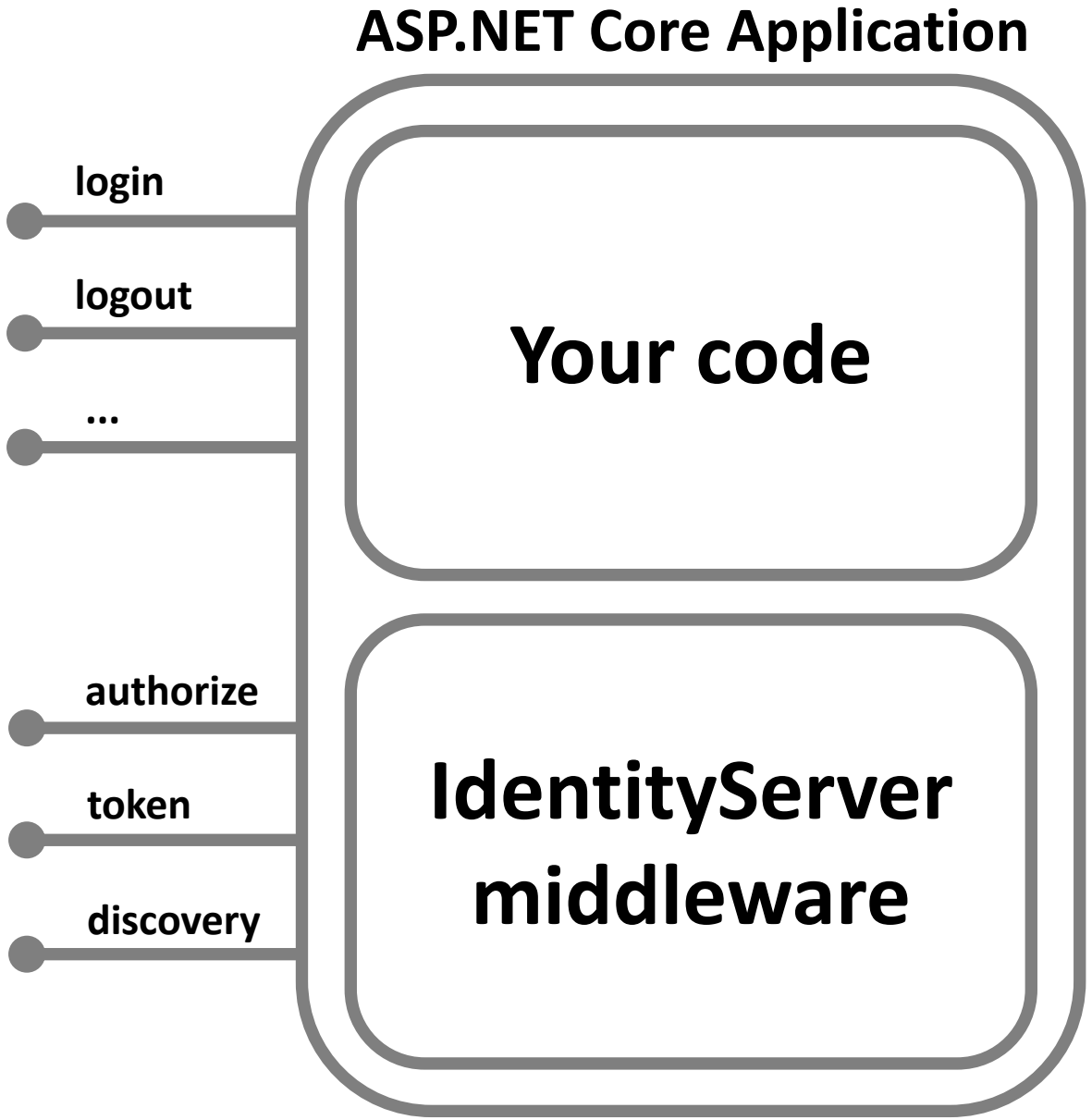
eyJhbGciOiJub25lIn0.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD.4MTkzODAsDQogImh0dHA6Ly9leGFt

Header **Payload** **Signature**

Identity Token Validation

- **According to 3.1.3.7 of the OpenID Connect specification**
  - The issuer name in the discovery document MUST exactly match the value of the **iss** claim.
  - The client MUST validate that the **aud** (audience) claim contains its **client_id** value registered at the issuer.
  - The **alg** value SHOULD be the default of RS256 or some other expected value.
  - The current time MUST be before the time represented by the **exp** Claim.
  - The **iat** claim can be used to reject tokens that were issued too far away from the current time.
  - The **nonce** value must match the nonce that was sent in the authentication request. A nonce claim MUST be present.
  - (some more checks for specific scenarios)

Setting up


ASP.NET Core Application
login
logout
...
Your code
authorize
token
discovery
IdentityServer
middleware

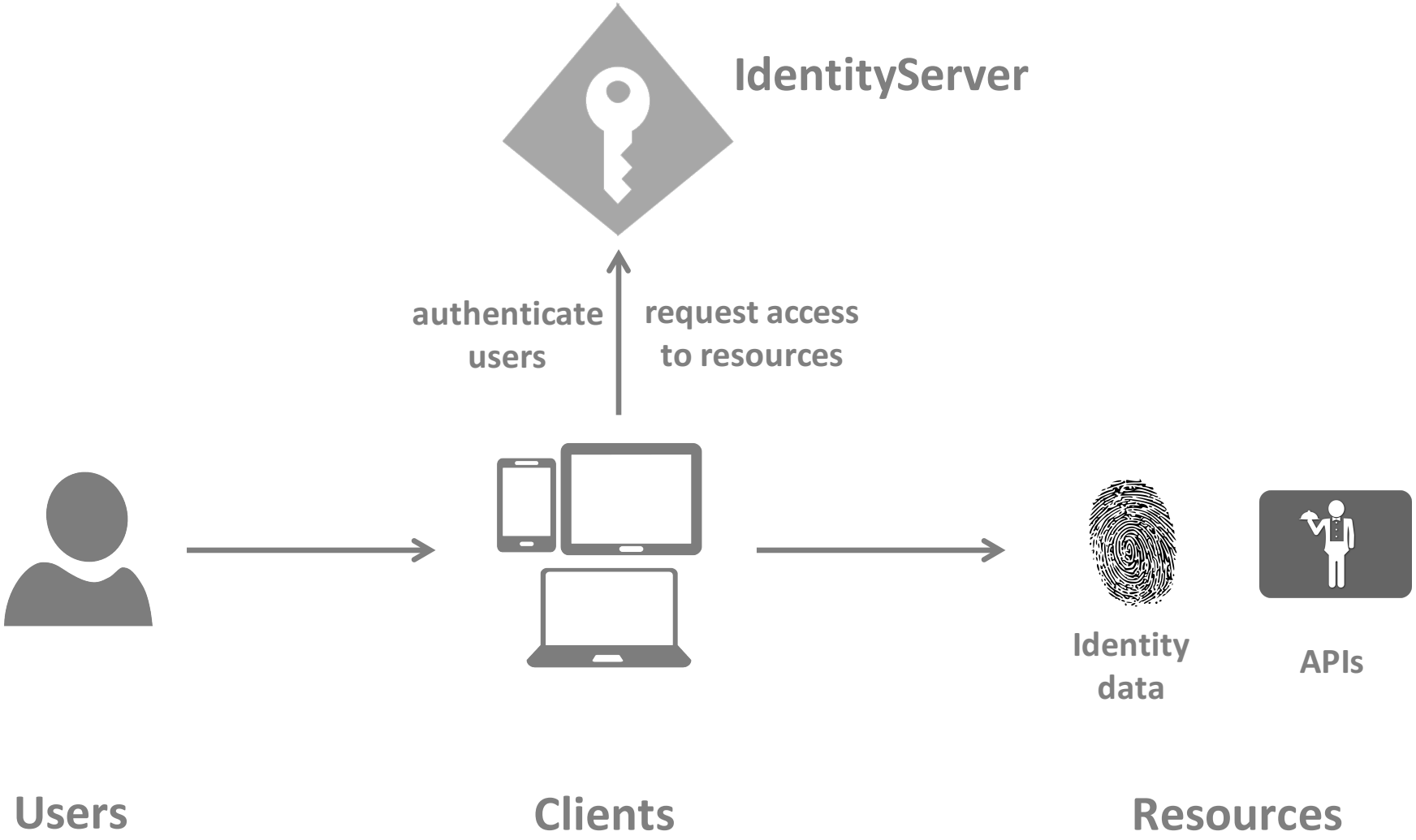
Basic Setup

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();

        services.AddIdentityServer(options =>
            {
                options.UserInteraction.LoginUrl = "/home/account";
            })
            .AddSigningCredential("CN=sts");
    }

    public void Configure(IApplicationBuilder app)
    {
        app.UseIdentityServer();

        app.UseStaticFiles();
        app.UseMvcWithDefaultRoute();
    }
}
```

adds authentication services, primary and temp cookie

adds authentication middleware

Modeling your Architecture



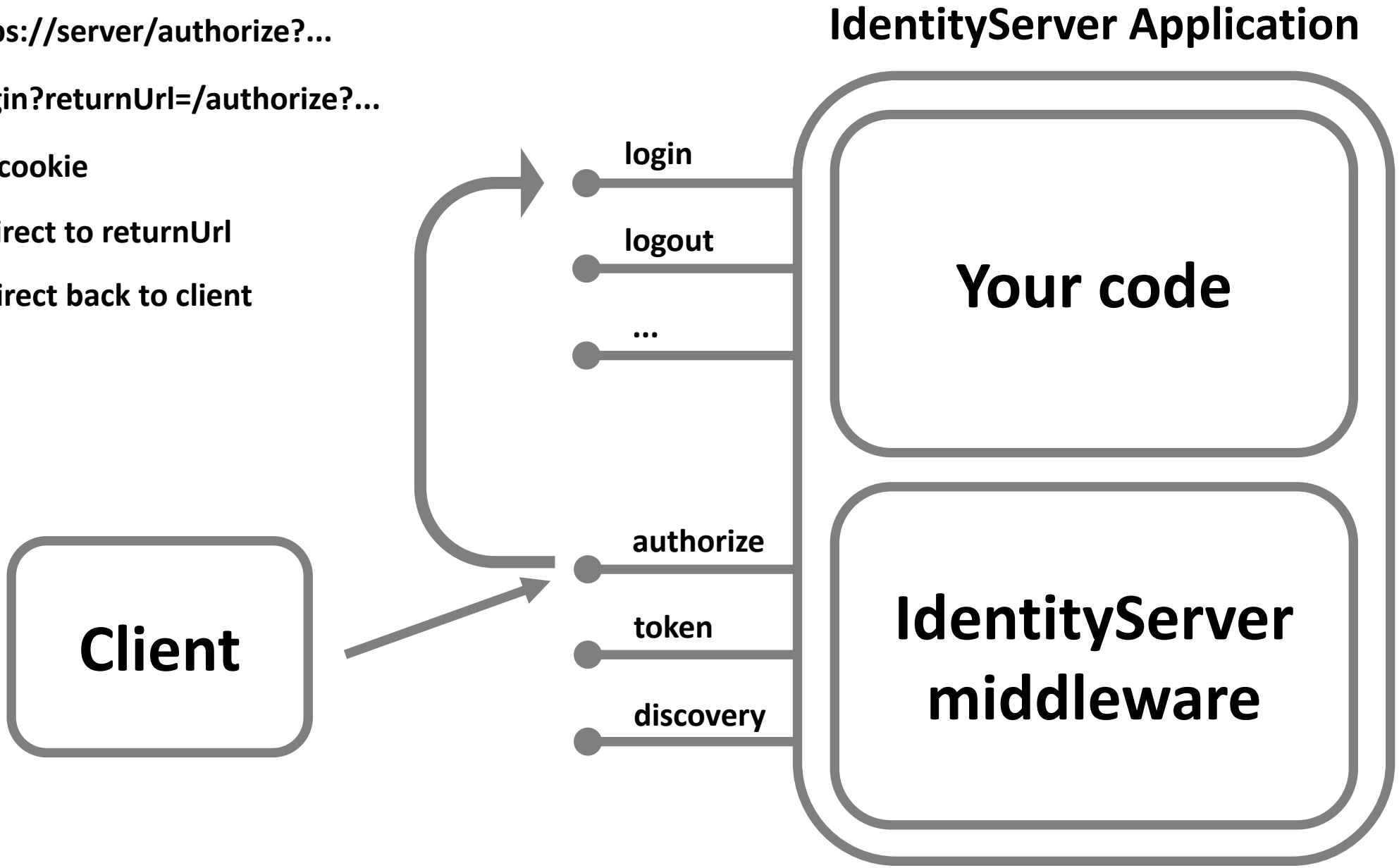
1) https://server/authorize?...

2) /login?returnUrl=/authorize?...

3) set cookie

4) redirect to returnUrl

5) redirect back to client



Connecting an MVC Client

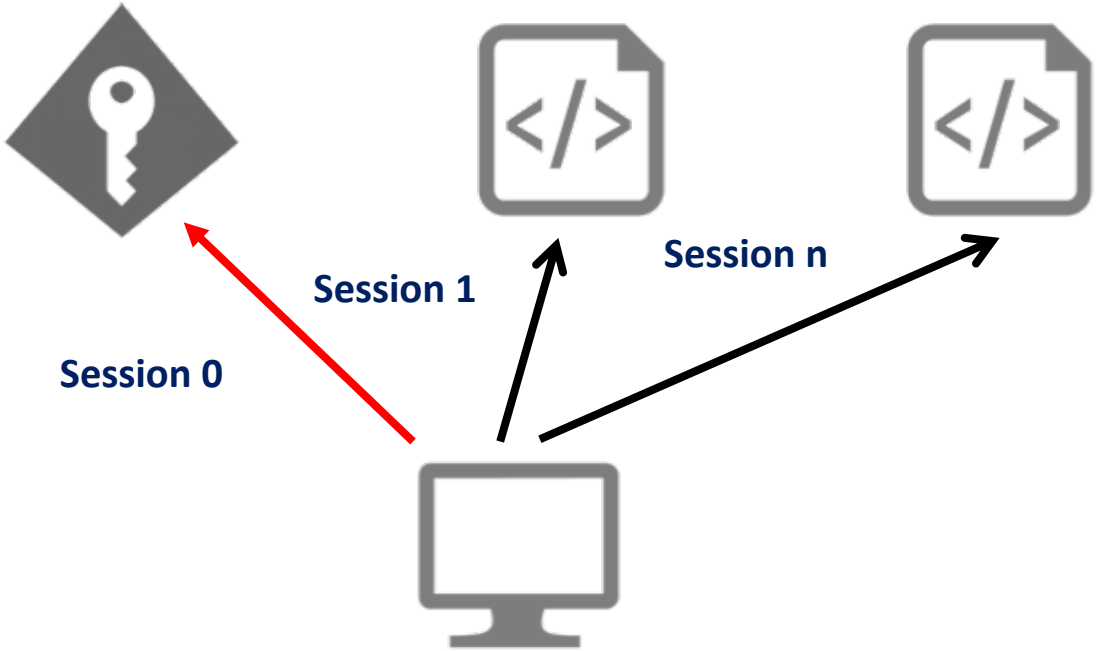
```
services.AddAuthentication("Cookies")
    .AddCookie("Cookies", options =>
    {
        options.LoginPath = "/account/login";
        options.AccessDeniedPath = "/account/denied";
    })
    .AddOpenIdConnect("oidc", options =>
    {
        options.Authority = "https://demo.identityserver.io";
        options.ClientId = "mvc";

        options.TokenValidationParameters = new TokenValidationParameters
        {
            NameClaimType = "name",
            RoleClaimType = "role"
        };
    });
```

Patterns

- **Single Sign On**
- **Single Sign Off**
- **Federation**
- **Home Realm Discovery**

Single Sign-On

- **OpenID Connect provider establishes a logon session with browser**
  - multiple clients in same browser use provided
  - for the duration of session, clients can request authentication user interaction
  - after successful authentication request, each client establishes its own session

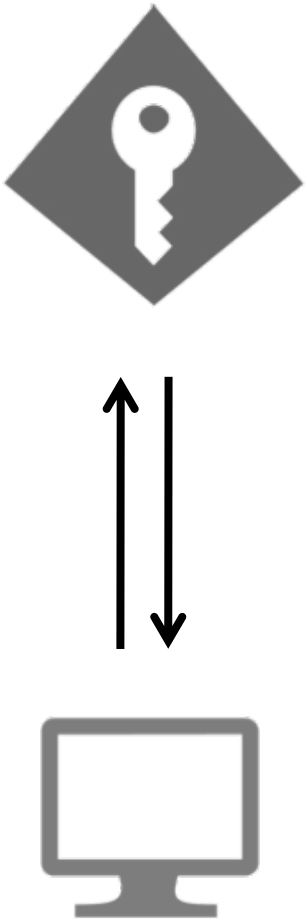


Single Sign-Out

- **Complete sign-out process consists of**
  1. clean up session at local RP
  2. clean up session at the STS
  3. clean up resources at all other RPs in the same session
  4. (clean up session at potential upstream STS)

- **Cleanup is complicated - thus three specs**
  - JS-based notifications
  - front-channel notification
  - back-channel notifications

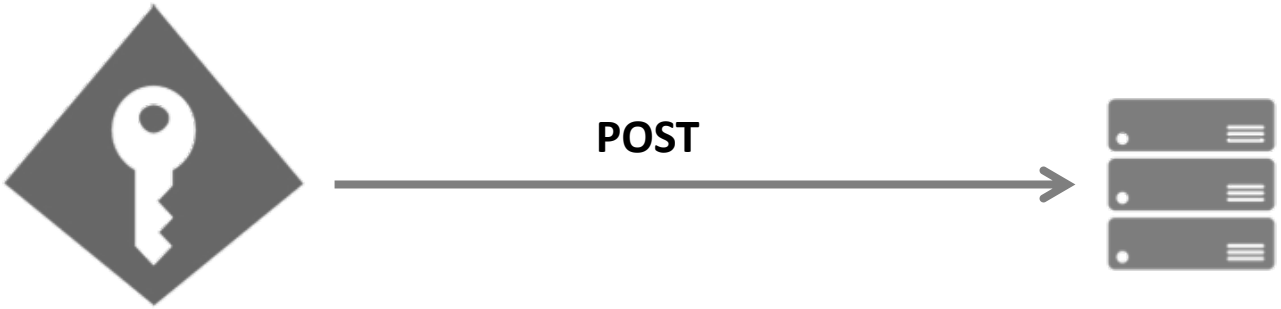
Front-Channel Notifications



GET /end_session

```
<iframe style="visibility:hidden"
     src="https://client1/signout?sid=123">
</iframe>
<iframe class="visibility:hidden"
     src="https://client2/signout?sid=123">
</iframe>
<iframe class="visibility:hidden"
     src="https://client3/signout?sid=123">
</iframe>

<a href="https://client1">return</a>
```

Back-Channel Notifications

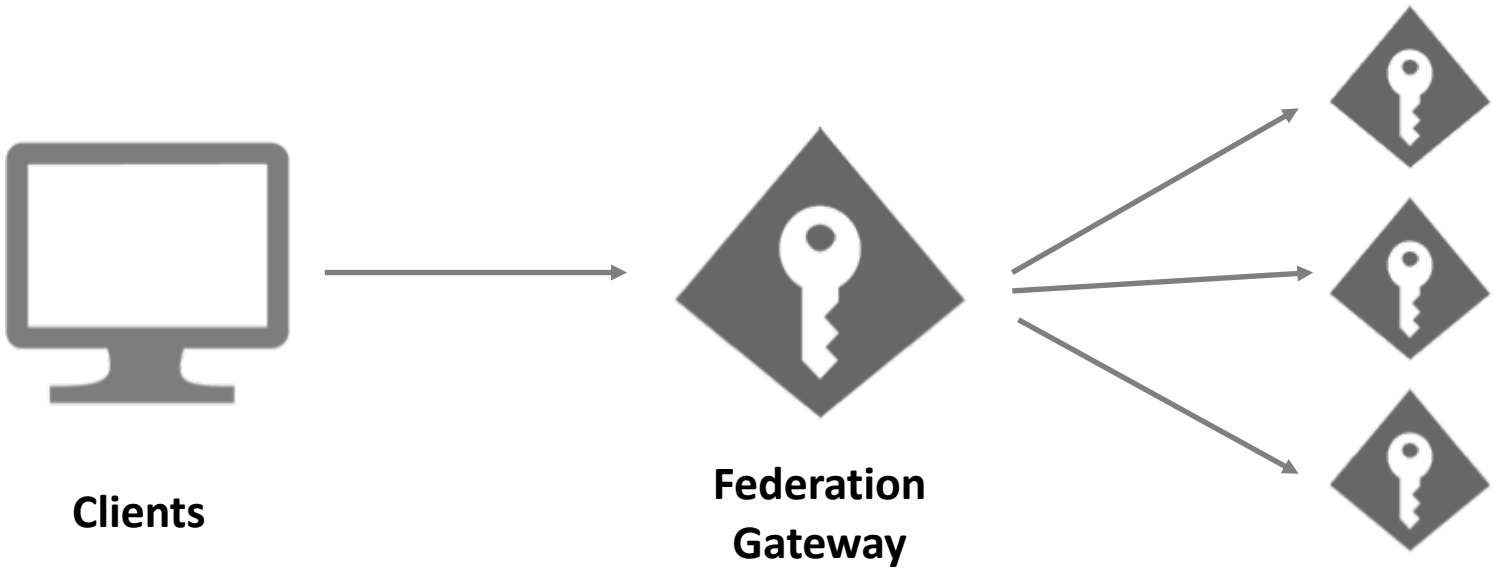


GET /end_session

Client


```
{
  "iss": "https://demo.identityserver.io",
  "sub": "248289761001",
  "aud": "client1",
  "iat": 1471566154,
  "jti": "bWJq",
  "sid": "8u09jejd099",
  "events": {
    "http://schemas.openid.net/event/backchannel-logout": {}
  }
}
```

Federation Gateway Pattern



- Active Directory
- Azure AD/B2C
- Business Partner
- Social Identities..

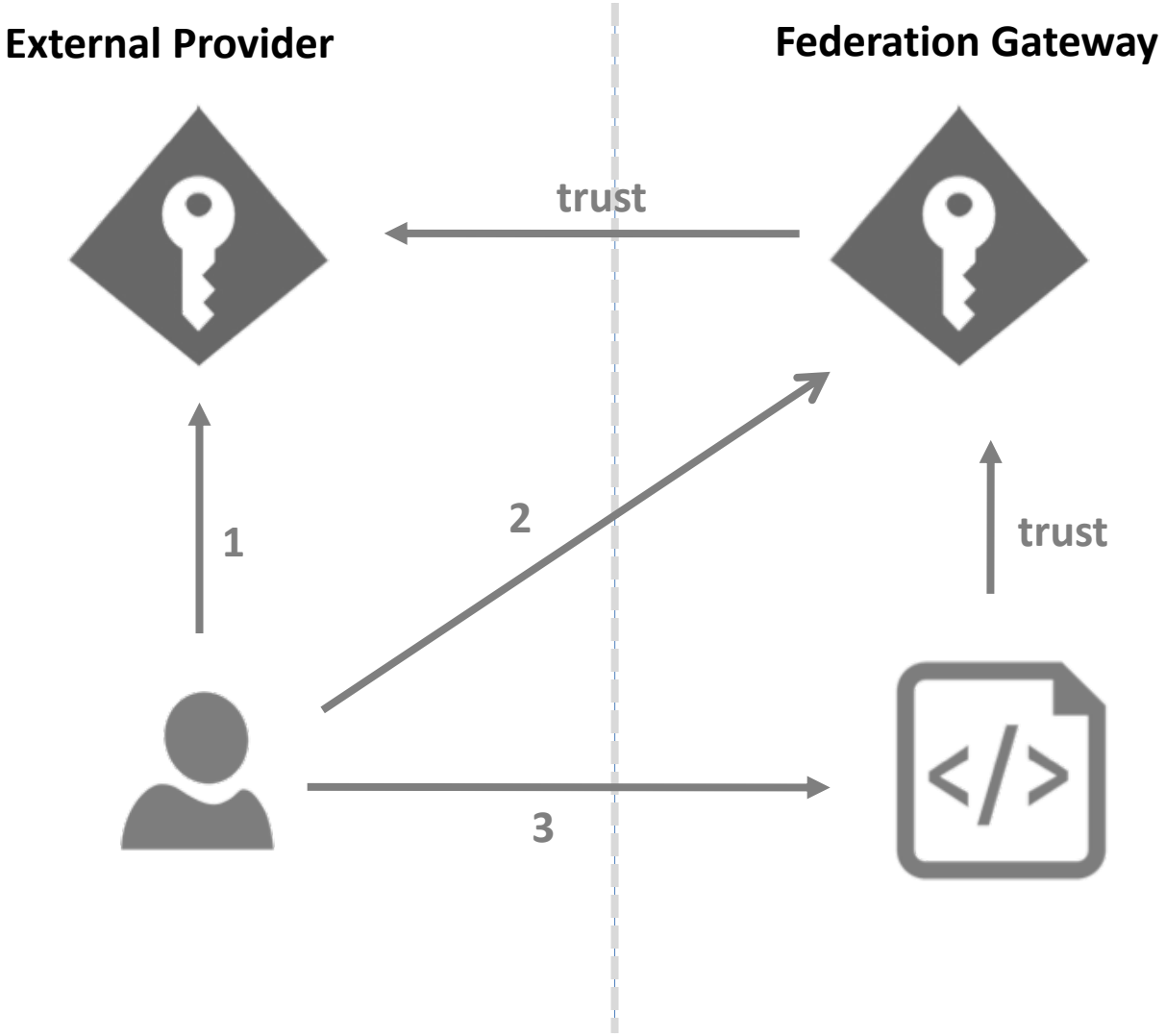
Benefits

- **Clients only "knows" about single provider (the gateway)**
- **Client shielded from all technical details (and changes over time)**

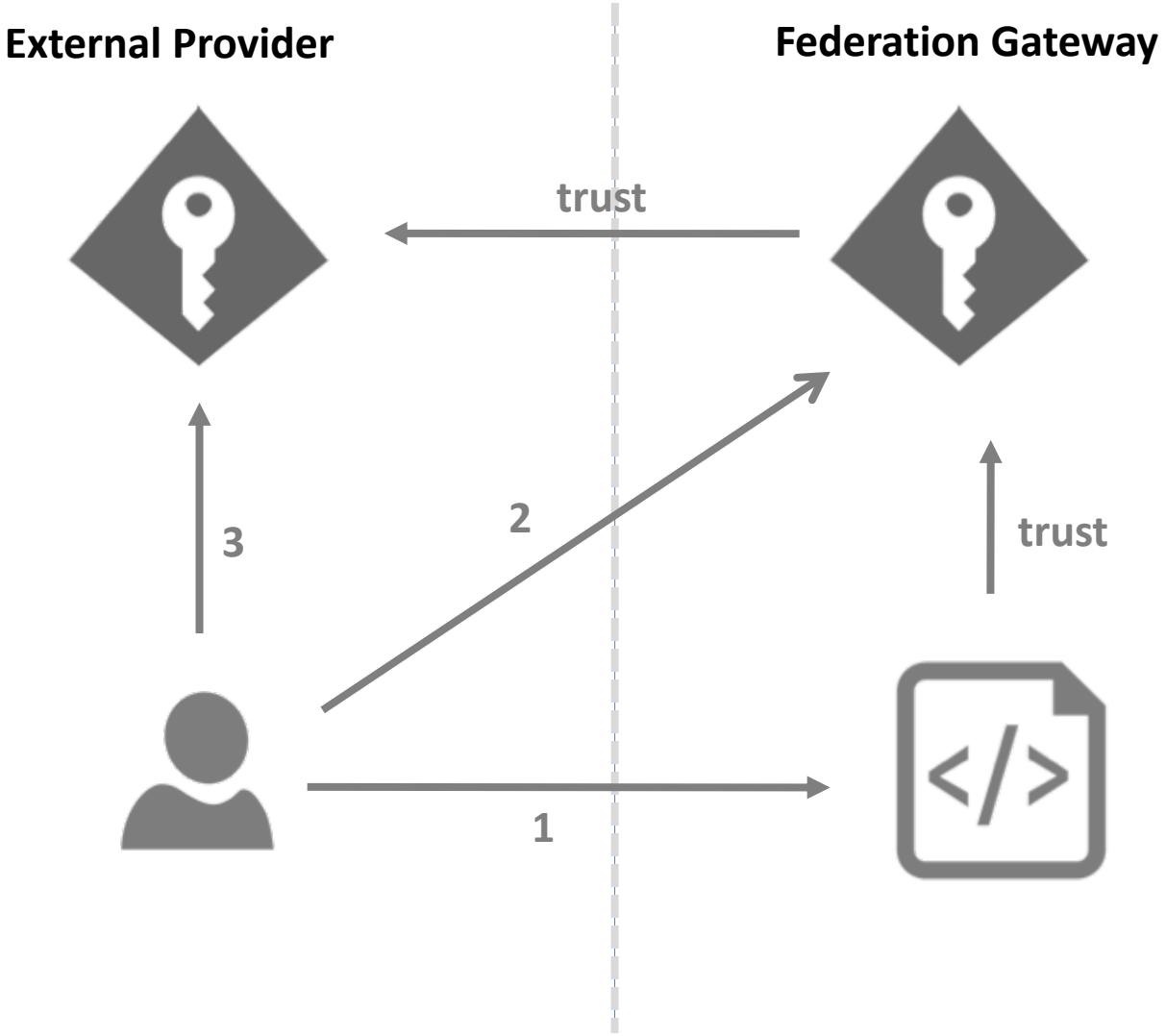
- **Gateway deals with all complexity**
  - protocols
  - token types
  - claim types and transformation
  - provisioning of external users

- **Gateway acts as single client to external provider**
  - can save money in IdaaS scenarios

Federation – logical model

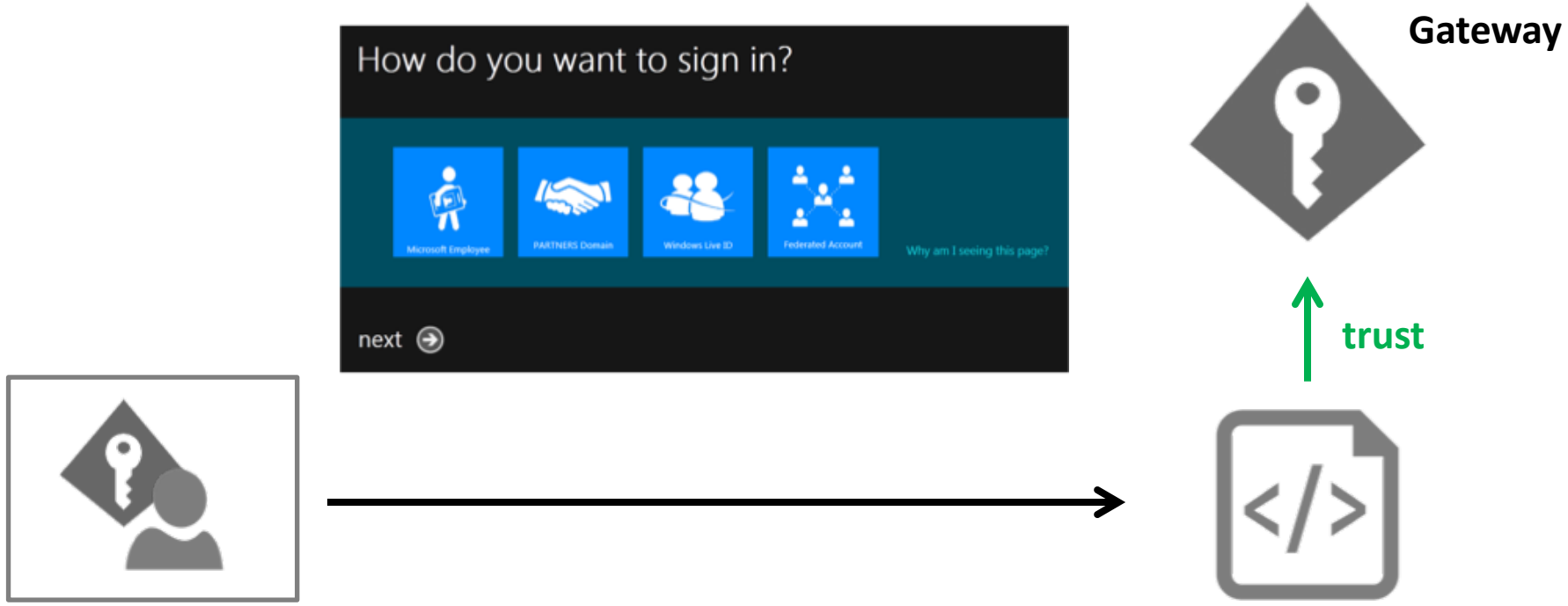


Federation – physical model



Home Realm Discovery (HRD)

- **How can we know which external provider to use when user is anonymous?**
  - some sort of hint required



Sending a home realm hint from client to provider

- **Unfortunately, no dedicated parameter in OpenID Connect**
  - Azure AD uses *domain_hint*
  - IdentityServer uses *acr_values*

Example: IdentityServer

```
https://idsrv/connect/authorize/?
  client_id=myapp&
  redirect_uri=https://www.myapp.com
  &acr_values=idp:ext_idp
```

Summary

- **ASP.NET Core is the latest and greatest web platform from Microsoft**
  - driven by a combination of middleware and DI services
  - claims-based identity
- **IAuthenticationService coordinates authentication handlers**
- **Policy- and resource-based authorization**
- **Authentication logic should be separated into an OpenID Connect provider**
  - enables patterns like single sign-on/out & federation
- **Home realm discovery needs a strategy**
  - hints from user, environment or context