Identidade e Controlo de Acesso baseado em Claims na plataforma Windows Azure

Pedro Félix e David Júlio

Microsoft, Tagus Park, Oeiras

27 e 28 de Janeiro de 2011

Outline

- Day 1
  - The claims-based model and specifications
  - Windows Identity Foundation (WIF)
  - The Azure AppFabric Access Control Service
  - Building ASP.NET based identity consumers (web apps) using WIF
  - WIF and ASP.NET on Windows Azure
- Day 2
  - Building WCF based identity consumers (web services) using WIF
  - WIF and WCF on Windows Azure
  - Building identity providers (web apps and services) using WIF
  - Active Directory Federation Services 2 (ADFS 2)

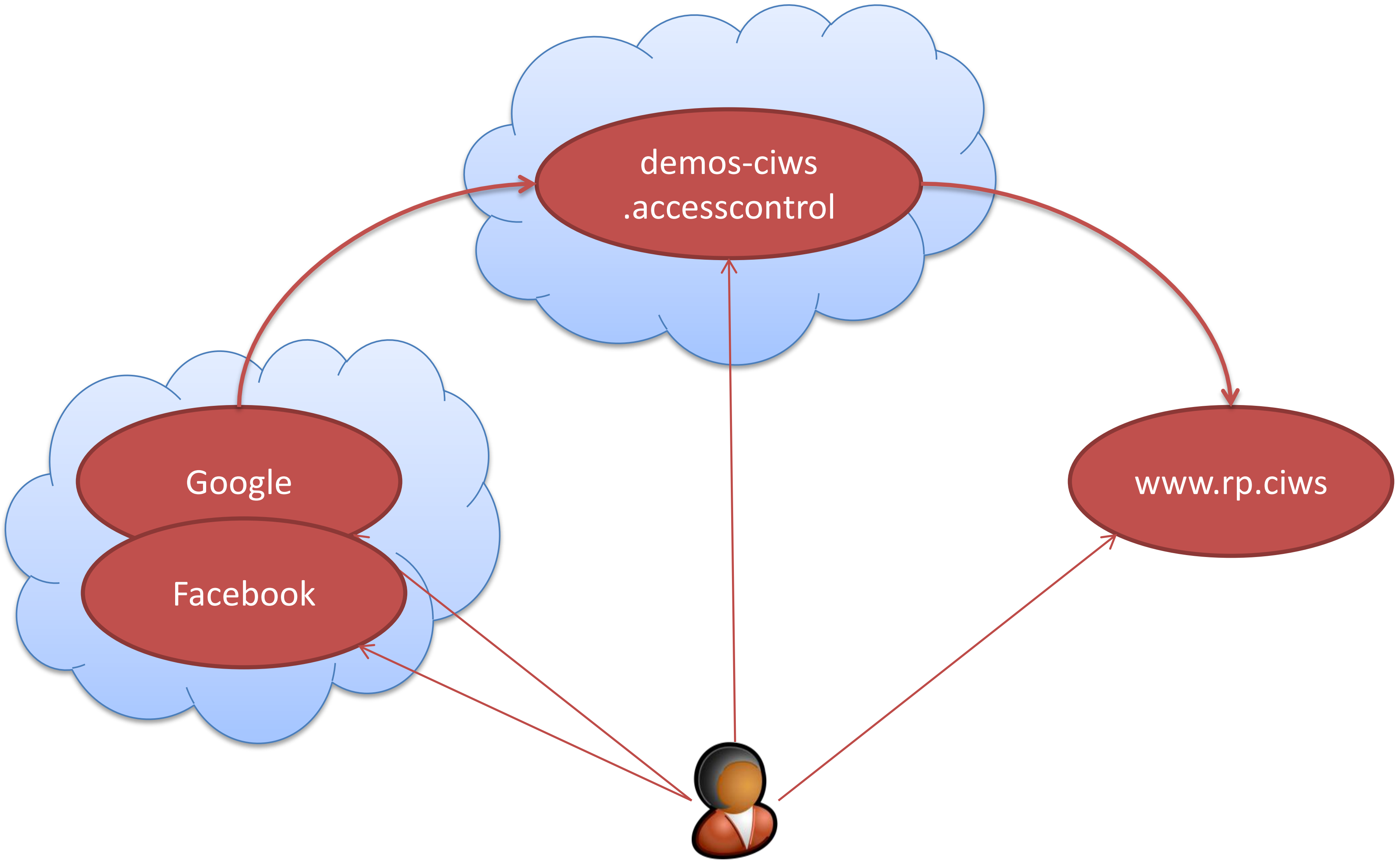
Day 1

- **HOL 1 – ASP.NET web app with Google and Facebook based authentication, using Access Control Service**
- The claims based model, use cases and specifications
- The Windows Identity Foundation
- The Azure AppFabric Access Control Service (ACS)
- **HOL 2 – creating and configuring an ACS tenant**
- WIF and ASP.NET deep dive
- **HOL 3 – exploring WIF and ASP.NET**
- Brief intro to Windows Azure
- WIF and Windows Azure
- **HOL 4 – moving a claims based ASP.NET web app to Windows Azure**

Hands-On-Lab 1

- Create default ASP.NET project
- Create and configure IIS site with HTTPS support
- Check behavior
- “Add STS reference”
  - https://demos-ciws.accesscontrol.appfabriclabs.com/FederationMetadata/2007-06/FederationMetadata.xml
- Check behavior
- Minor “hacks”
- Check behavior
- Add code to show claims

HOL 1



CLAIMS BASED MODEL

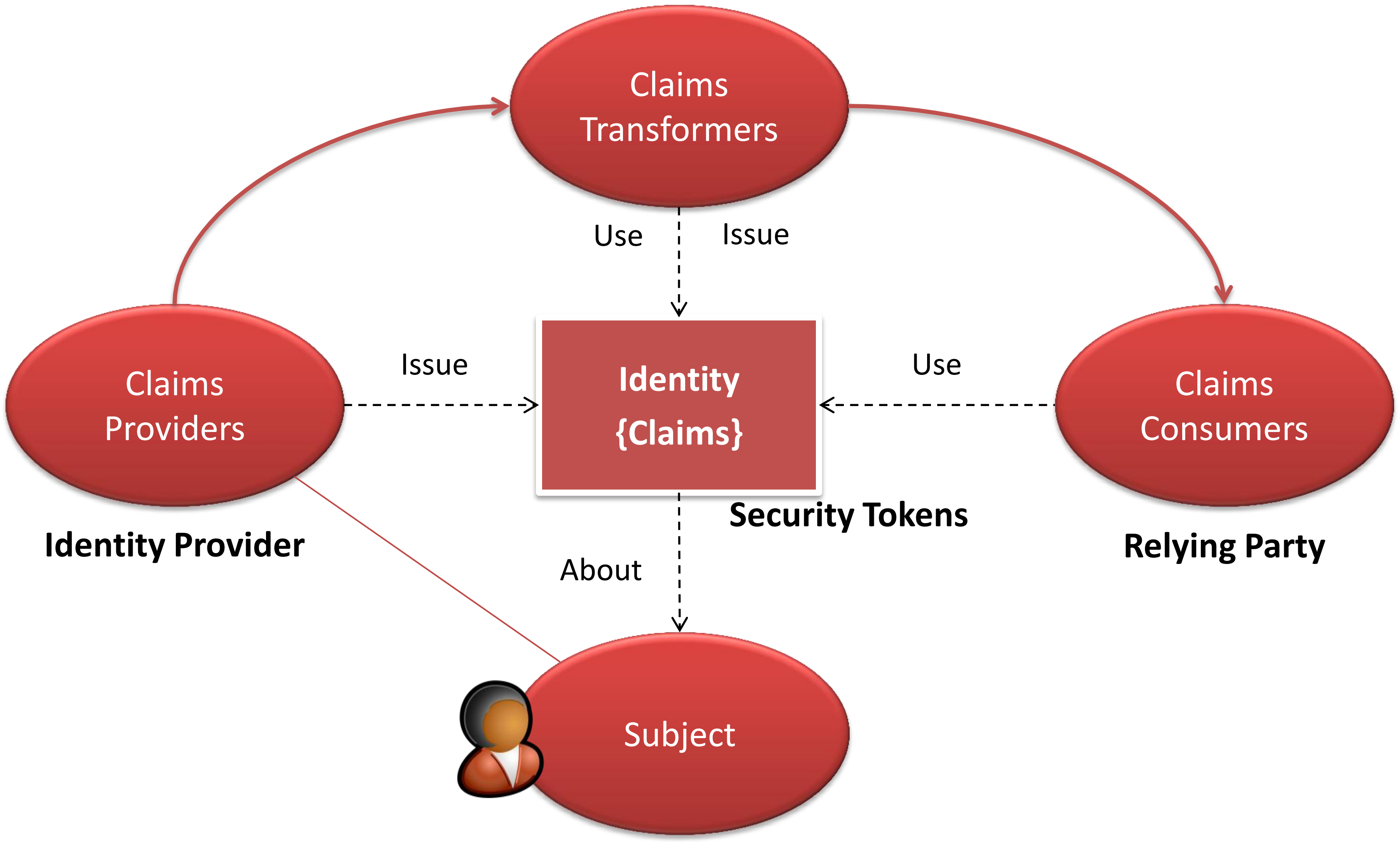
The claims model - claims

- *Identity* as a set of claims
  - Google::alice4demos@gmail.com
  - Facebook::”Alice Demos”
  - ISEL::**student**
  - OrganizationA::**Employee**
  - ADSE::**Beneficiary**
  - PurchaseApp::**PurchaseManager**

- A *claim* is a declaration made by an entity
  - Entities: ISEL, OrganizationA, Google, Facebook
  - Claims: name, role, email, authorization

- Opposite to *Identity* as a *unique identifier*

The Claims Model - participants



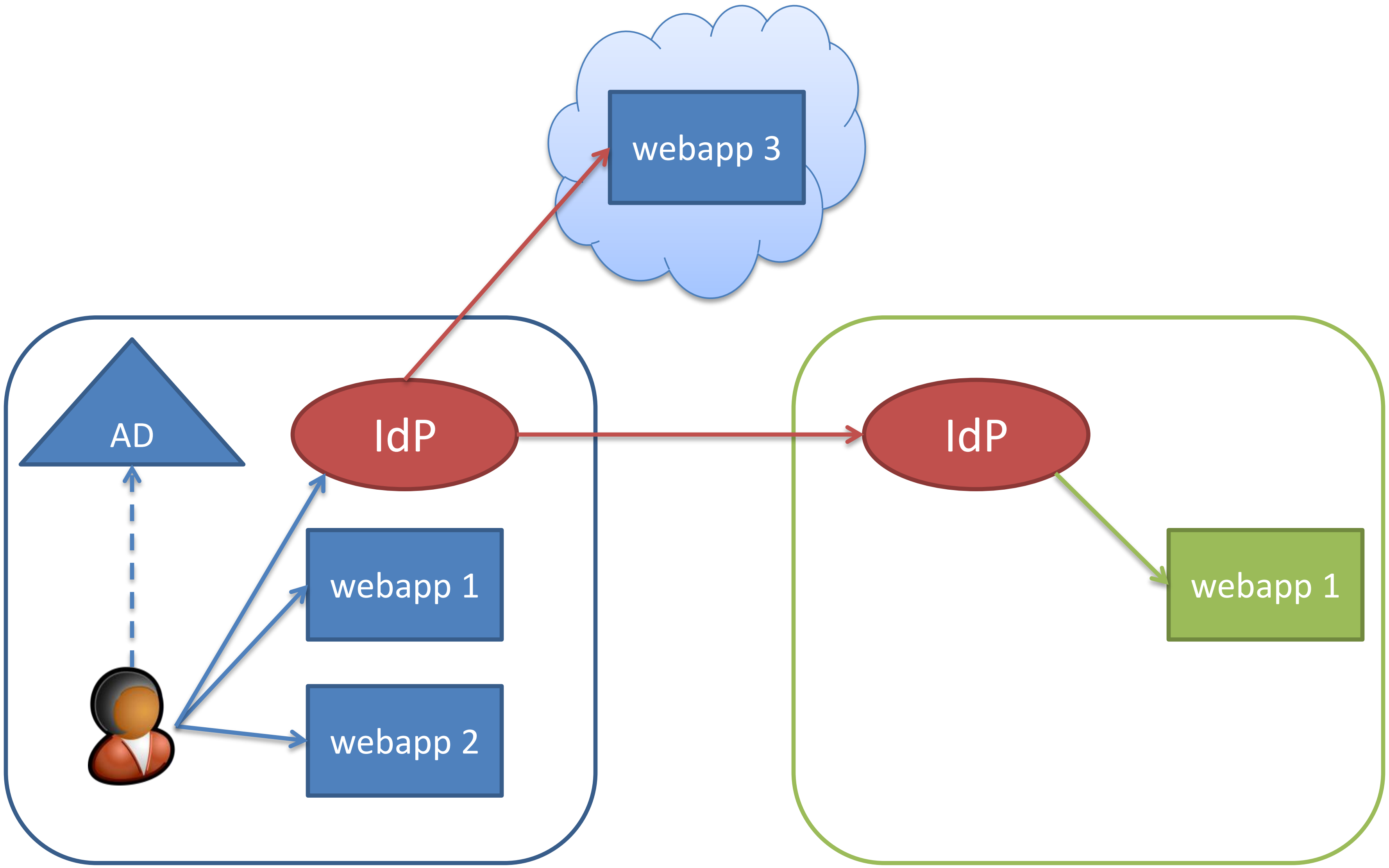
Concrete components

- Claims
- Tokens
  - Claims packaging
    - Origin authentication
    - Confidentiality
    - Proof-of-possession
- Protocols
  - Token request and communication
- Policy and metadata
  - Requirements and capabilities description

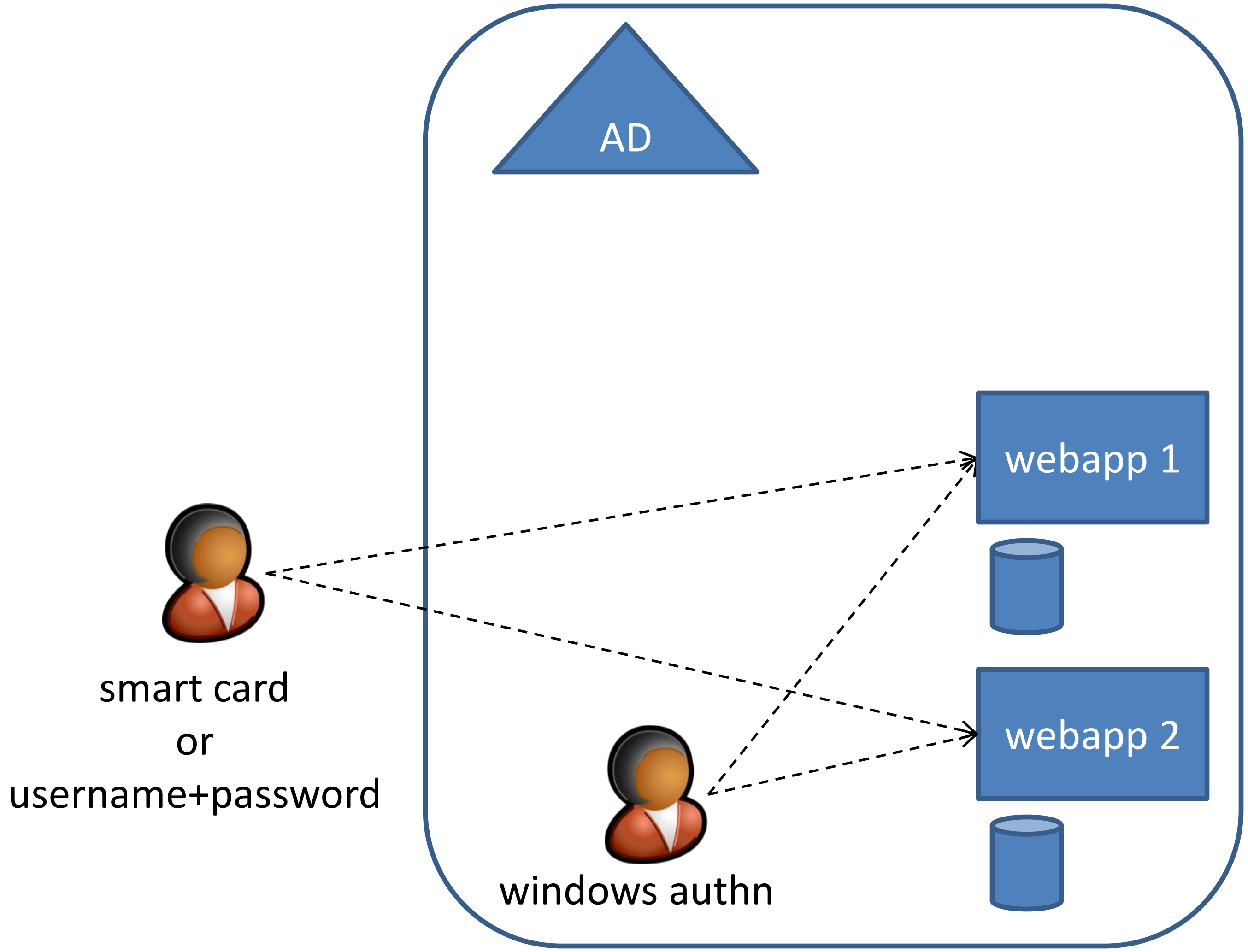
Claims transformers

- Example: https://demos-ciws.accesscontrol.appfabriclabs.com/
- Bridge between
  - Technological boundaries
    - Protocols: E.g. OpenID protocol $\leftrightarrow$ WS-Fed protocol
    - Token formats: SAML $\leftrightarrow$ SWT
  - Organizational boundaries
    - E.g. Domain group $\rightarrow$ Public Role
    - E.g. Public Role $\rightarrow$ Authorization
- Claim inference – policy evaluation

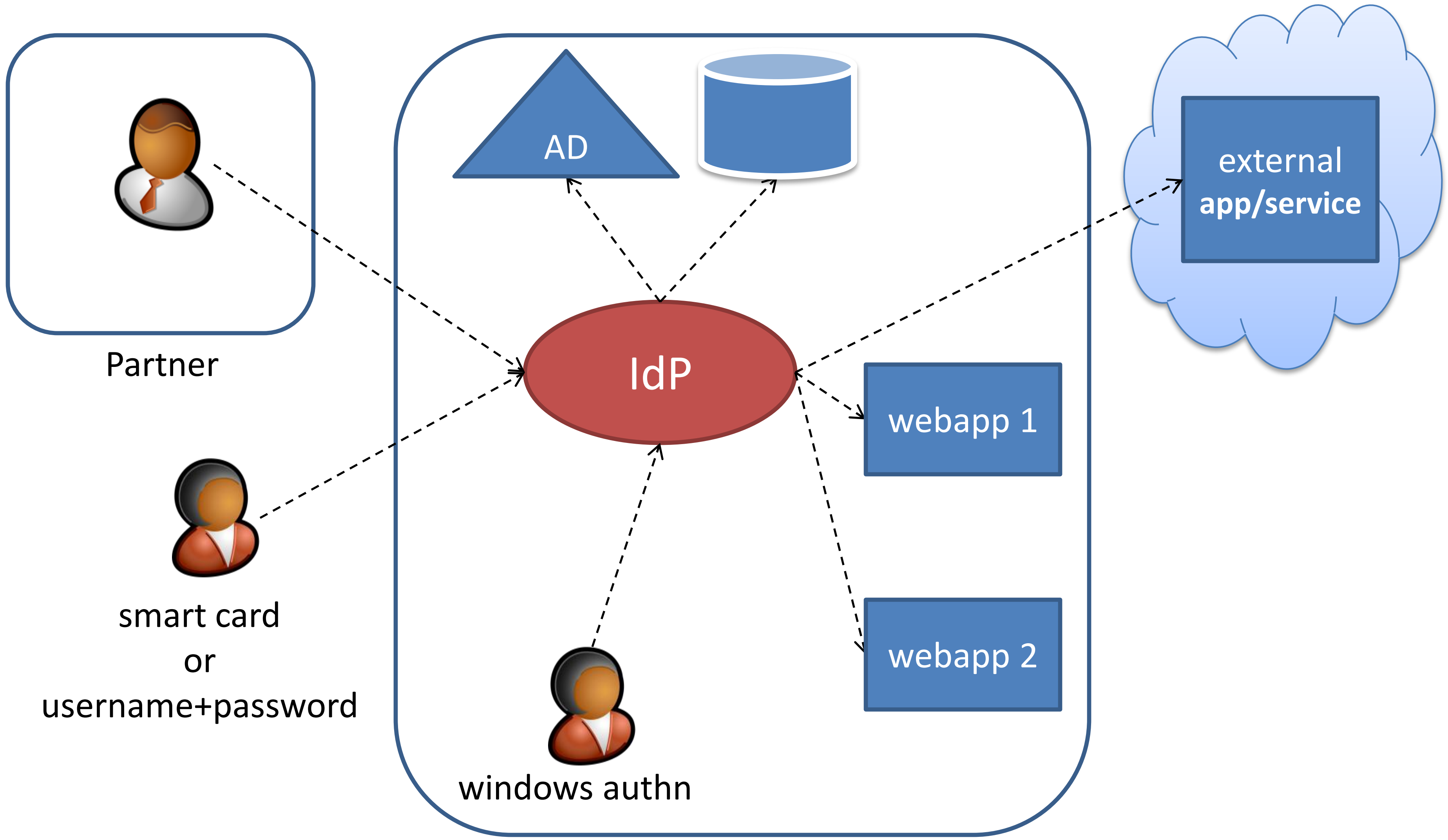
Federation



Not only for Federation



Not only for Federation



SPECIFICATIONS

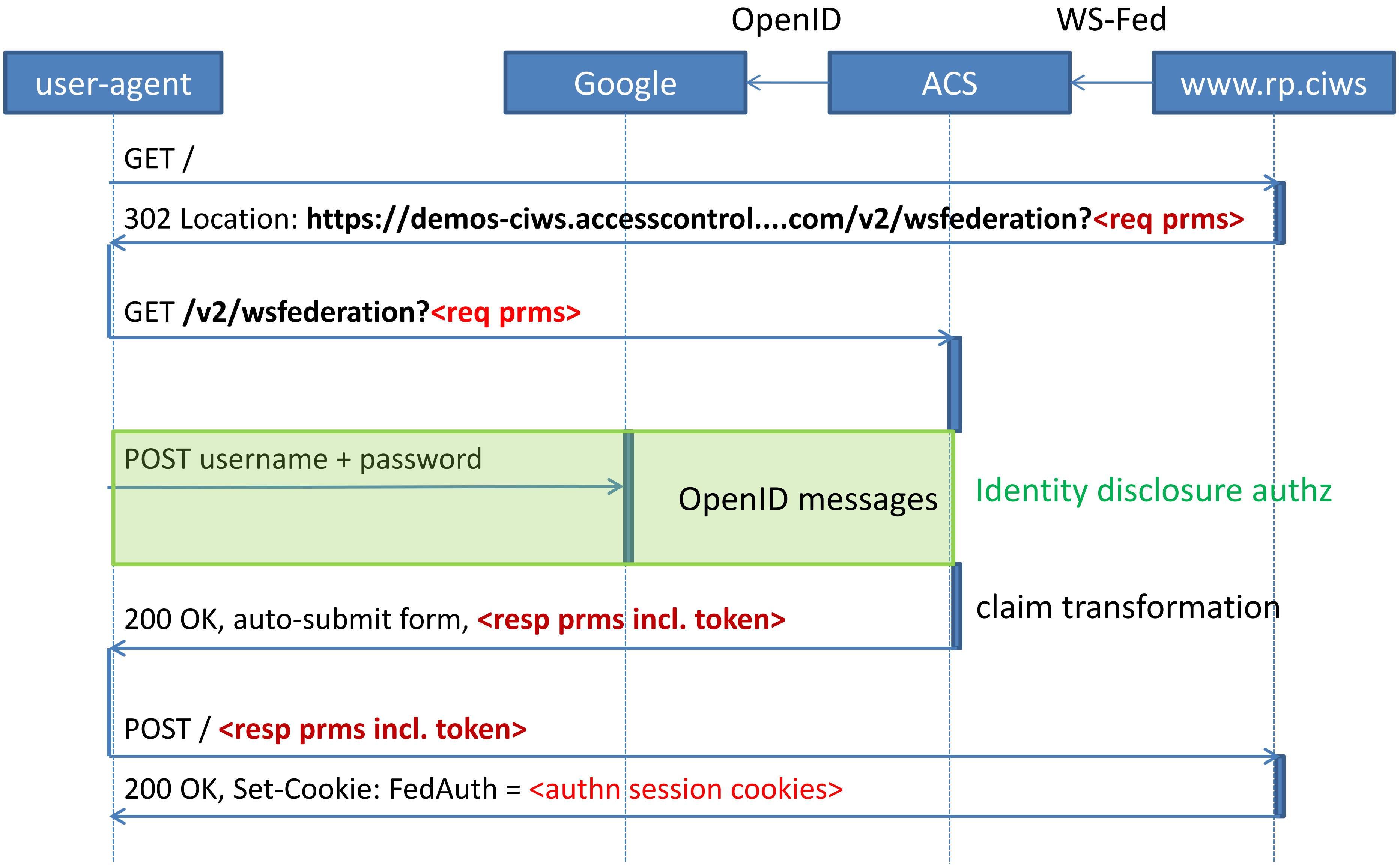
Specifications

- Protocols
  - WS-Federation
- Token formats
  - SAML (Security Assertion Markup Language)
- Metadata
  - SAML and WS-Federation metadata

WS-Federation

- OASIS specification
  - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsfed
- Defines
  - Metadata format
  - Protocol for passive scenarios (browser)
    - HTTP protocol usage
    - Message parameters
    - Active protocol is defined by WS-Trust
- Uses
  - Metadata format from SAML
  - Messages formats from WS-Trust

WS-Federation


OpenID
WS-Fed
user-agent
Google
ACS
www.rp.ciws
GET /
302 Location: **https://demos-ciws.accesscontrol....com/v2/wsfederation?<req prms>**
GET **/v2/wsfederation?<req prms>**
POST username + password
OpenID messages
Identity disclosure authz
claim transformation
200 OK, auto-submit form, **<resp prms incl. token>**
POST / **<resp prms incl. token>**
200 OK, Set-Cookie: FedAuth = <authn session cookies>

WS-Federation

- Request parameters
  - **wa** = wsignin1.0 (action)
  - **wtrealm** = https://www.rp.ciws:8443 (req. realm)
  - **wctx** = <opaque> (req. context)
  - **wct** = <date-time>
  - wreply = <return url>
  - whr = <home realm>
- Response parameters
  - **wa** = wsignin1.0
  - **wctx** = <opaque> (req. context)
  - **wresult** = < **RequestSecurityTokenResponse** XML elem.>

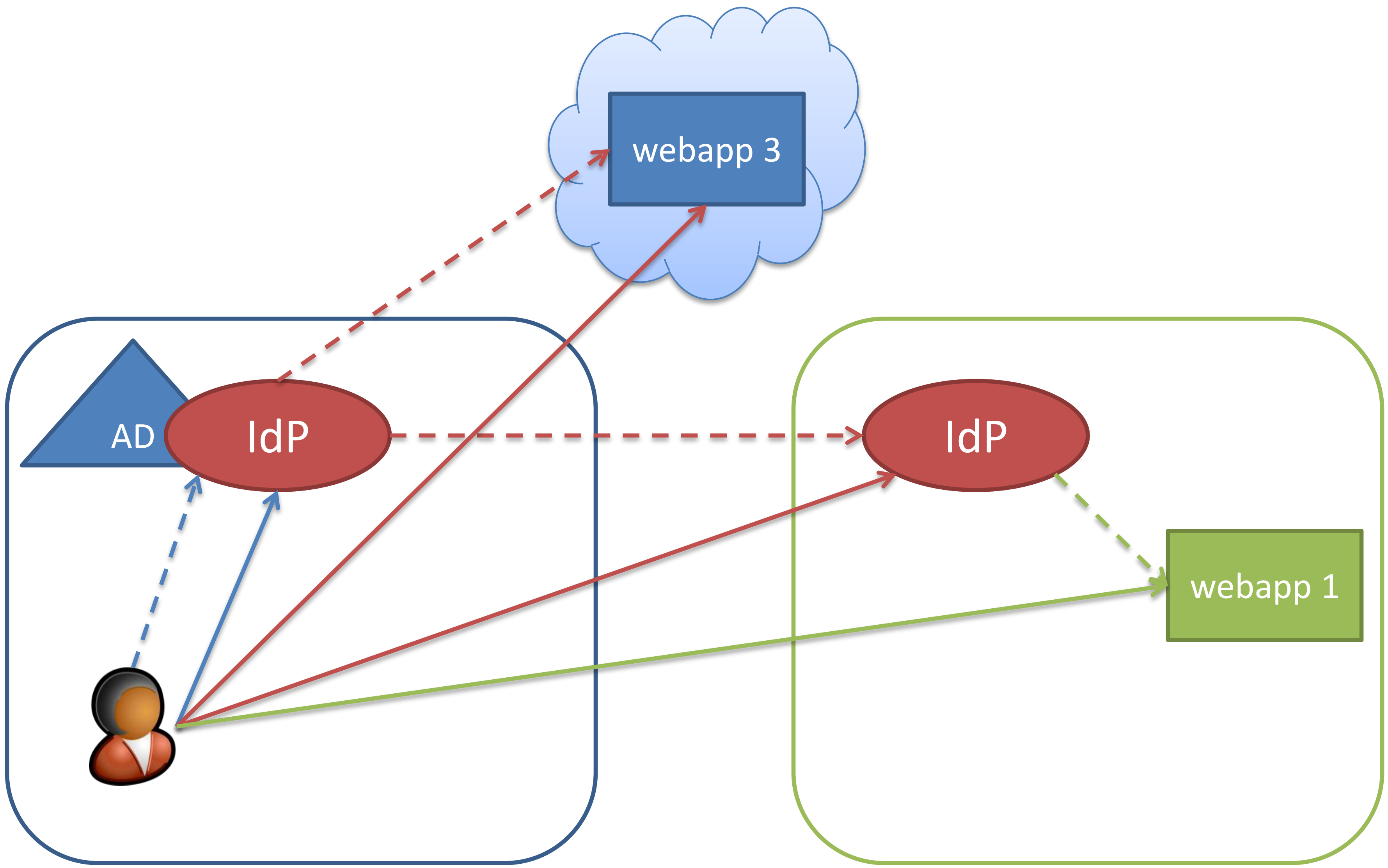
RequestSecurityTokenResponse

```
<RequestSecurityTokenResponse Context = “req. ctx”>
   <Lifetime>
      <Created> …
      <Expires> …
   <AppliesTo>
      <EndpointReference>
         <Address> https://www.rp.ciws:8443
   <RequestedSecurityToken>
         [SAML assertion, incl. signature]
   <TokenType> urn:oasis:names:tc:SAML:2.0:assertion
   <RequestType>  Issue
   <KeyType> NoProofKey
```

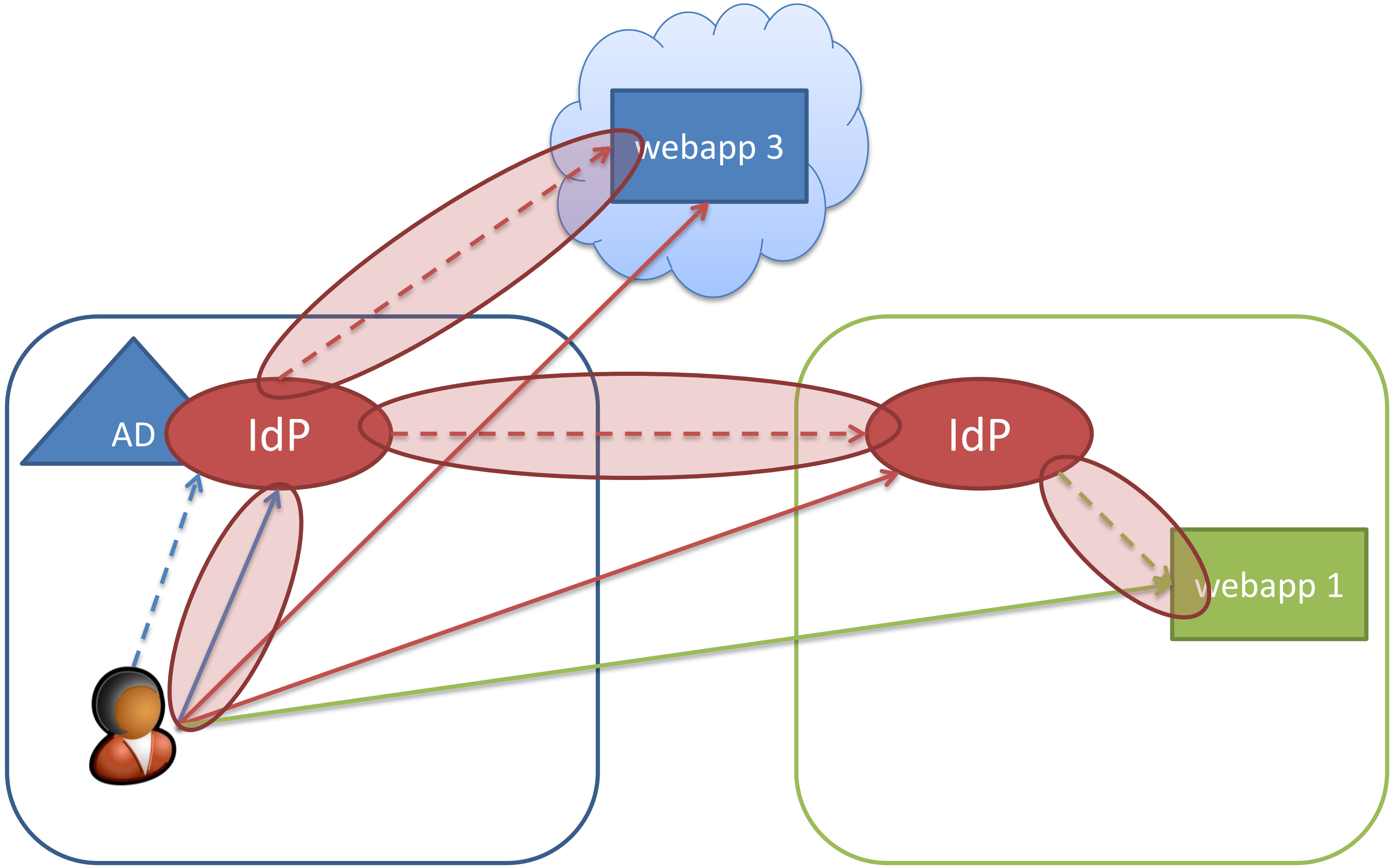
SAML Assertion

```
<Assertion ID = “…” IssueInstant = “…” Version = “2.0”>
  <Issuer> https://demos-ciws.accesscontrol.appfabriclabs.com/
  <Signature> [XML DSIG signature, incl. certificate(s)]
  <Subject>
    <NameID>https://www.google.com/accounts/08/id? id = xxxxx
    <SubjectConfirmation Method=“Bearer”>
  <Conditions NotBefore=“…” NotAfter=“…” >
    <AudienceRestriction><Audience>https://www.rp.ciws:8443
  <AttributeStatement>
    <Attribute Name = “emailaddress”>
      <AttributeValue>alice4demos@gmail.com
   <Attribute Name = “name”>
      <AttributeValue>Alice Demos
   <Attribute Name = “identityprovider”>
      <AttributeValue>Google
```

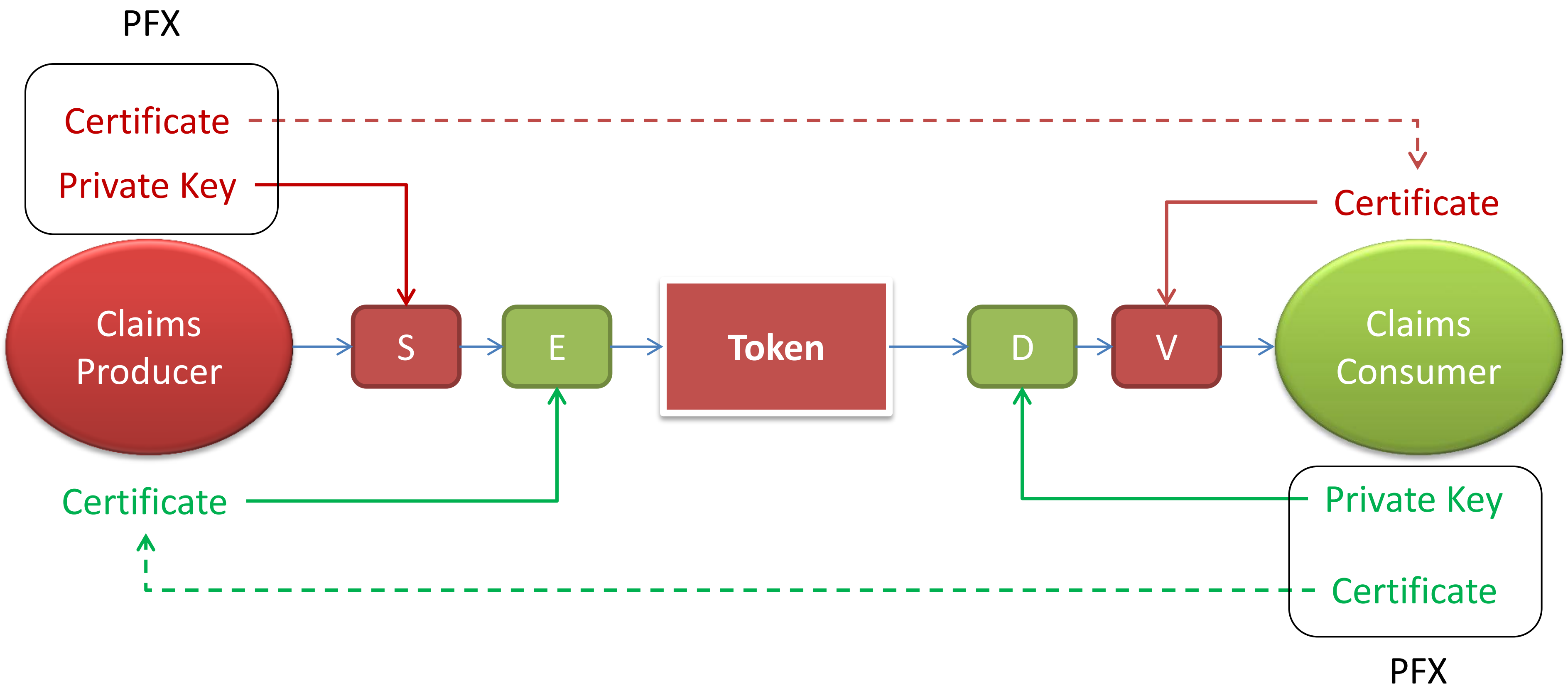
Federation



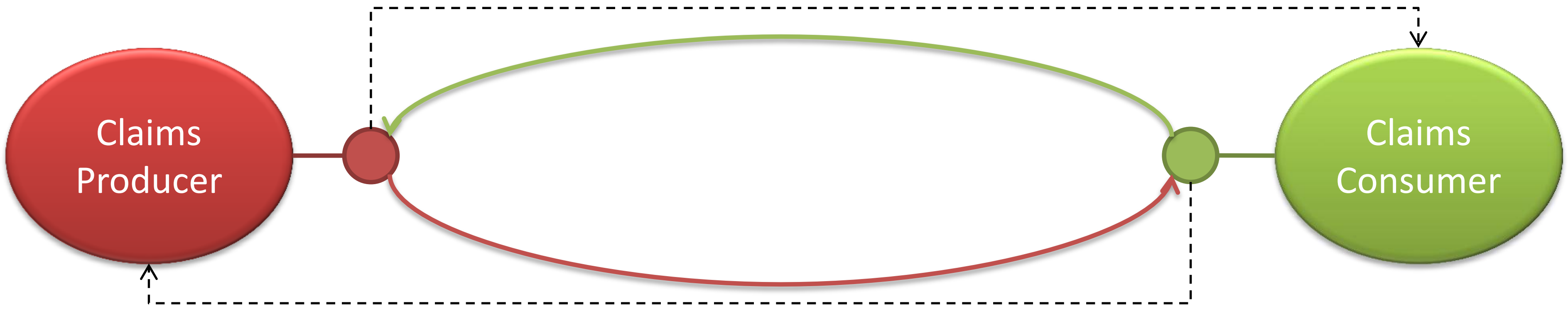
Relations



Relations - keys



Relations - endpoints



Metadata – Identity Provider

[https://demos-ciws.accesscontrol.appfabriclabs.com/FederationMetadata/2007-06/FederationMetadata.xml]

```
<EntityDescriptor>
  <Signature> [XML DSIG signature]
  <RoleDescriptor type="SecurityTokenService">
    <KeyDescriptor use="signing"><KeyInfo> [X509Certificate] [copy to file]
    <ClaimTypesOffered>…
    <SecurityTokenServiceEndpoint>
      <EndpointReference> [WS-Addressing endpoint reference]
    <SecurityTokenServiceEndpoint>
      <EndpointReference> [WS-Addressing endpoint reference]
    <PassiveRequestorEndpoint>
      <EndpointReference><Address>
        https://demos-ciws.accesscontrol.appfabriclabs.com/v2/wsfederation
  <RoleDescriptor type="ApplicationServiceType">
    (…)
```

Metadata – www.rp.ciws

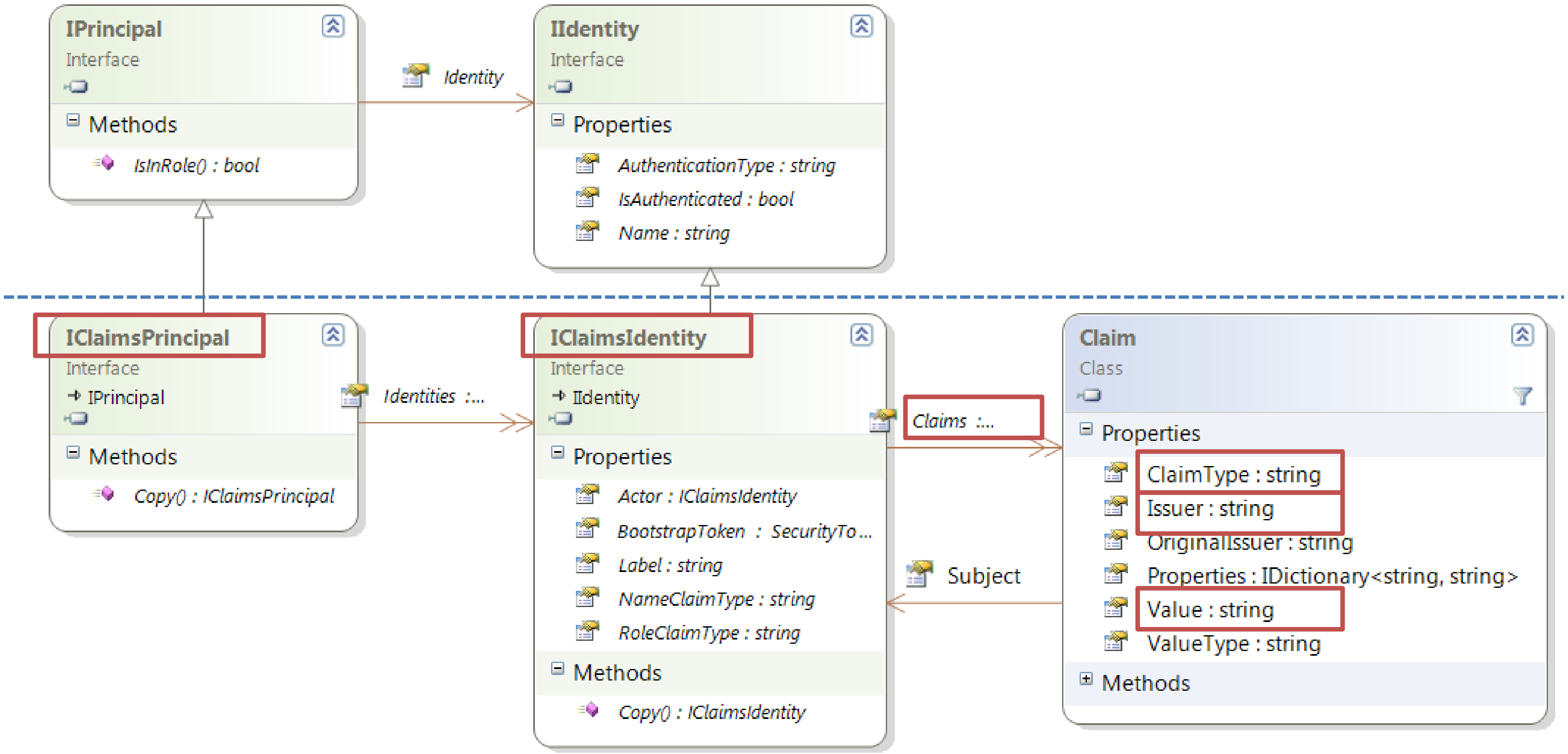
```
<EntityDescriptor>
  <RoleDescriptor type=“ApplicationServiceType”>
    <ClaimTypesRequested>
      <ClaimType Uri=“.../name” Optional=“true” />
      <ClaimType Uri=“.../role” Optional=“true” />
    <TargetScopes>
      <EndpointReference><Address>https://www.rp.ciws:8443/
   <PassiveRequestorEndpoint>
      <EndpointReference><Address>https://www.rp.ciws:8443/
```

WINDOWS IDENTITY FOUNDATION

WIF

- Class model for identity representation
- Protocol and token handling
- Claims consumption pipeline
  - Token validation
  - Identity transformation
  - Authorization decision
- Claims issuance pipeline

Claims object model



Claim types

```
public static class ClaimTypes {
   public const string AuthenticationInstant = "...";
   public const string AuthenticationMethod = "...";
   public const string DateOfBirth = "...";
   public const string Dns = "...";
   public const string Email = "...";
   public const string Gender = "...";
   public const string GivenName = "...";
   public const string GroupSid = "...";
   public const string HomePhone = "...";
   public const string Name = "...";
   public const string PostalCode = "...";
   public const string PPID = "...";
   public const string Role = "...";
   public const string Rsa = "...";
   public const string StreetAddress = "...";
   public const string Surname = "...";
   public const string WindowsAccountName = "...";
   public const string X500DistinguishedName = "...";
   (…)
}
```

Value types

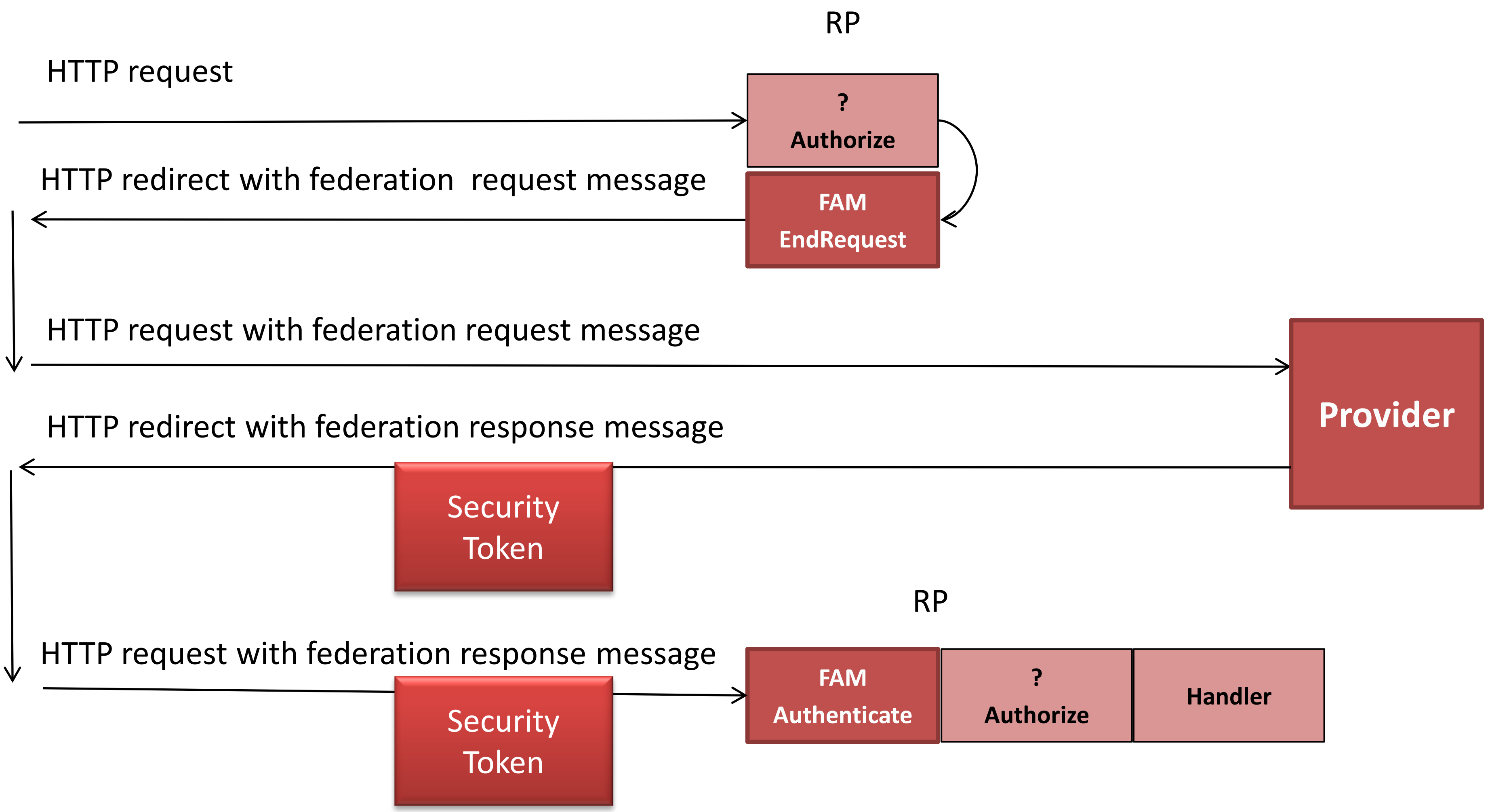
```
public static class ClaimValueTypes
{
   public const string Base64Binary = "http://www.w3.org/2001/XMLSchema#base64Binary";
   public const string Boolean = "http://www.w3.org/2001/XMLSchema#boolean";
   public const string Date = "http://www.w3.org/2001/XMLSchema#date";
   public const string Datetime = "http://www.w3.org/2001/XMLSchema#dateTime";
   public const string DaytimeDuration = "…./WD-xquery-operators-20020816#dayTimeDuration";
   public const string Double = "http://www.w3.org/2001/XMLSchema#double";
   public const string HexBinary = "http://www.w3.org/2001/XMLSchema#hexBinary";
   public const string Integer = "http://www.w3.org/2001/XMLSchema#integer";
   public const string KeyInfo = "http://www.w3.org/2000/09/xmldsig#KeyInfo";
   public const string Rfc822Name = "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name";
   public const string RsaKeyValue = "http://www.w3.org/2000/09/xmldsig#RSAKeyValue";
   public const string String = "http://www.w3.org/2001/XMLSchema#string";
   public const string Time = "http://www.w3.org/2001/XMLSchema#time";
   public const string X500Name = "urn:oasis:names:tc:xacml:1.0:data-type:x500Name";
   public const string YearMonthDuration = "http://...20020816#yearMonthDuration";
}
```

Protocol Handling

- ASP.NET modules
  - WSFederationAuthenticationModule (FAM)
  - SessionAuthenticationModule (SAM)

```
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true">
    <add name="..." type="WSFederationAuthenticationModule, ..." />
    <add name="..." type="SessionAuthenticationModule, ..." />
  </modules>
</system.webServer>
```

WS-Federation Authn Module (FAM)

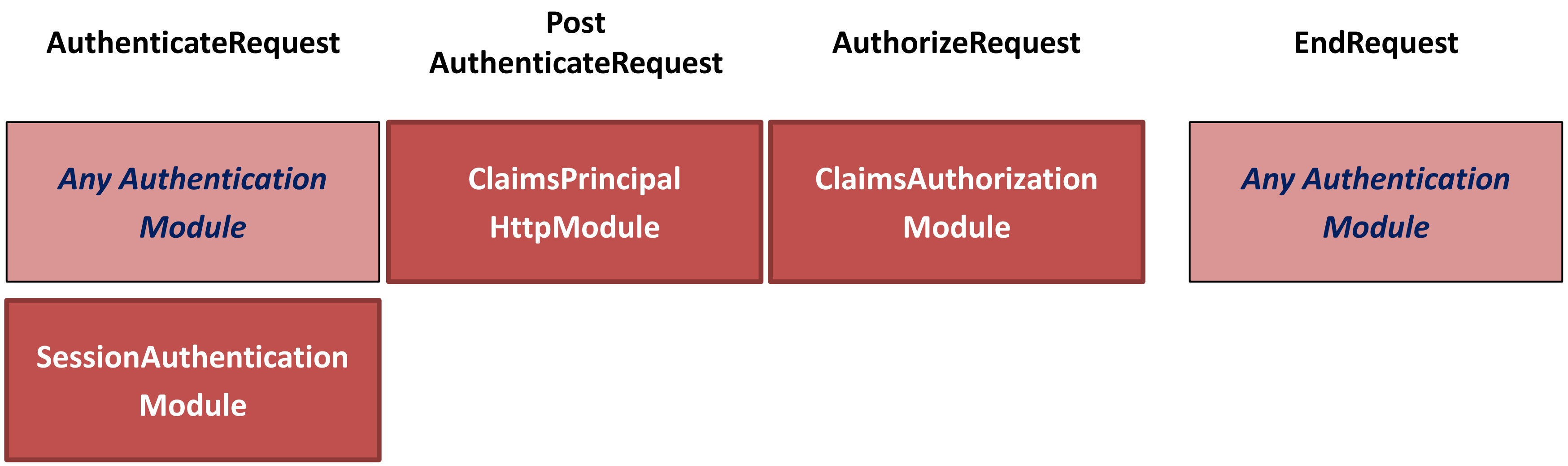

RP
HTTP request
?
Authorize
HTTP redirect with federation request message
FAM
EndRequest
HTTP request with federation request message
Provider
HTTP redirect with federation response message
Security Token
RP
HTTP request with federation response message
Security Token
FAM
Authenticate
?
Authorize
Handler

ASP.NET Integration

AuthenticateRequest	Post AuthenticateRequest	AuthorizeRequest	EndRequest
WSFedAuthentication Module	WSFedAuthentication Module	Authorization Module	WSFedAuthentication Module
SessionAuthentication Module	SessionAuthentication Module		

ASP.NET Integration

- Using a *legacy* authentication mechanism
  - e.g. Forms authentication, MVC filters


AuthenticateRequest
Post AuthenticateRequest
AuthorizeRequest
EndRequest
Any Authentication Module
ClaimsPrincipal HttpModule
ClaimsAuthorization Module
Any Authentication Module
SessionAuthentication Module

Configuration (web.config)

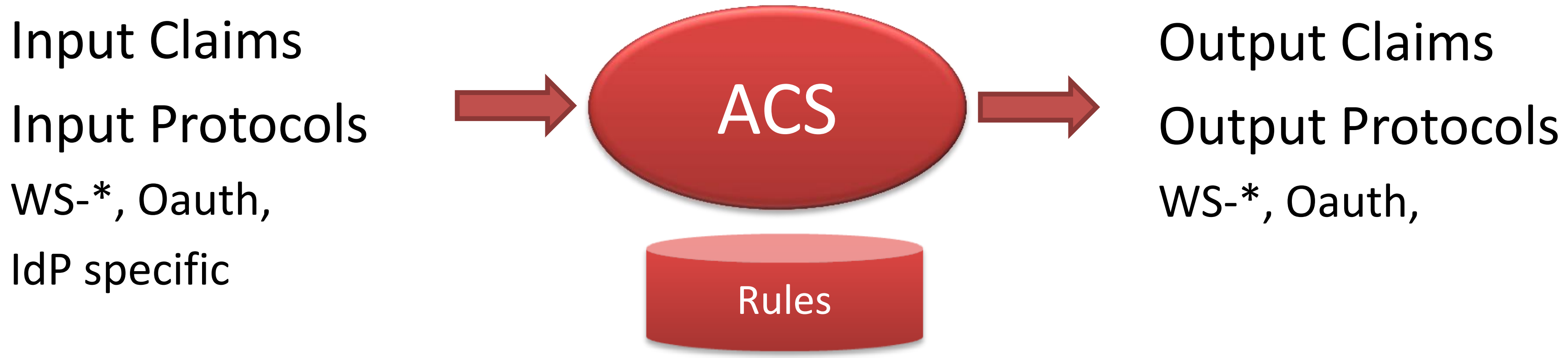
```
<microsoft.identityModel>
  <service>
    <audienceUris><add value="https://www.rp.ciws:8443/" />
    <federatedAuthentication>
      <wsFederation passiveRedirectEnabled="true"
          issuer="https://demos-ciws.accesscontrol.appfabriclabs.com/v2/wsfederation"
          realm="https://www.rp.ciws:8443" requireHttps="true" />
      <cookieHandler requireSsl="true" />
    <applicationService><claimTypeRequired>
        <claimType type="…/name" optional="true" />
        <claimType type="…/role" optional="true" />
        <!--<claimType type="…/nameidentifier" optional="true" />-->
        <!--<claimType type="…/identityprovider" optional="true" />-->
    <issuerNameRegistry type="...ConfigurationBasedIssuerNameRegistry>
      <trustedIssuers>
        <add thumbprint="67F6E7E72A93776D3AEA5EB73537077626B90FDF"
              name="https://demos-ciws.accesscontrol.appfabriclabs.com/" />
    <certificateValidation certificateValidationMode="None" />
```

Configuration (web.config)

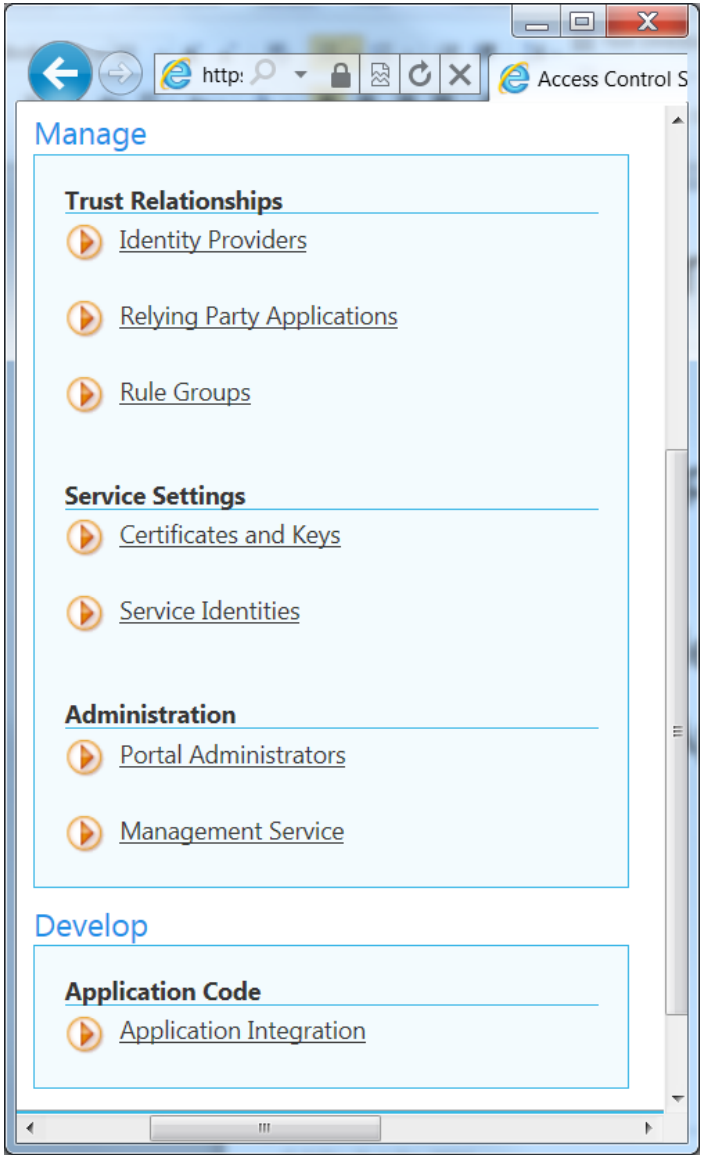
- Additional changes introduced by fedutil
  - <authorization><deny users="?"/>
  - <authentication mode="None" />
  - Allow users="*" to path="FederationMetadata"

Access Control Service (ACS)

- Part of Azure AppFabric
- Version 1 in production, version 2 in “labs”
- SaaS – Software as a Service
  - Service namespace ≈ tenant
- Claims transformer



ACS configuration



Access Control S

Manage

Trust Relationships

- Identity Providers
- Relying Party Applications
- Rule Groups

Service Settings

- Certificates and Keys
- Service Identities

Administration

- Portal Administrators
- Management Service

Develop

Application Code

- Application Integration

- Identity Providers
  - Relation with *upstream* IdPs
- Relying Parties
  - Relation with downstream RPs
- Rule groups
  - Claims transformation rules

- Certificates and keys
- Service identities
  - Direct identities and credentials

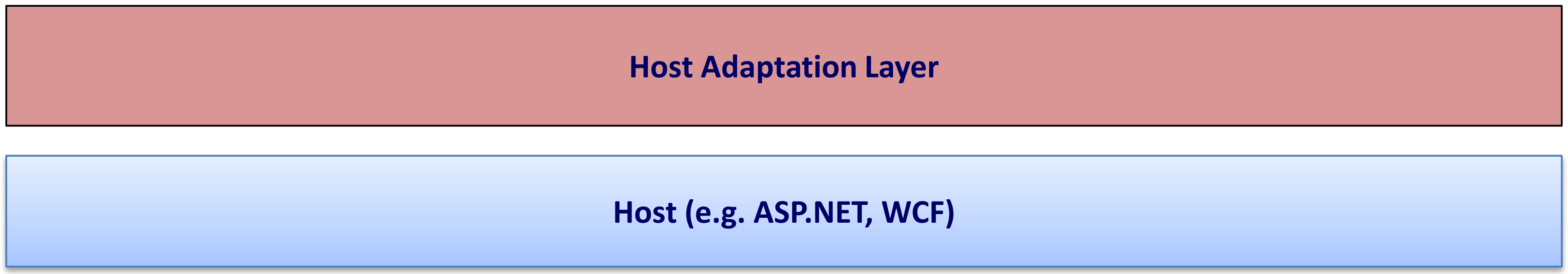
- Application integration
  - Addresses

Hands-On-Lab 2

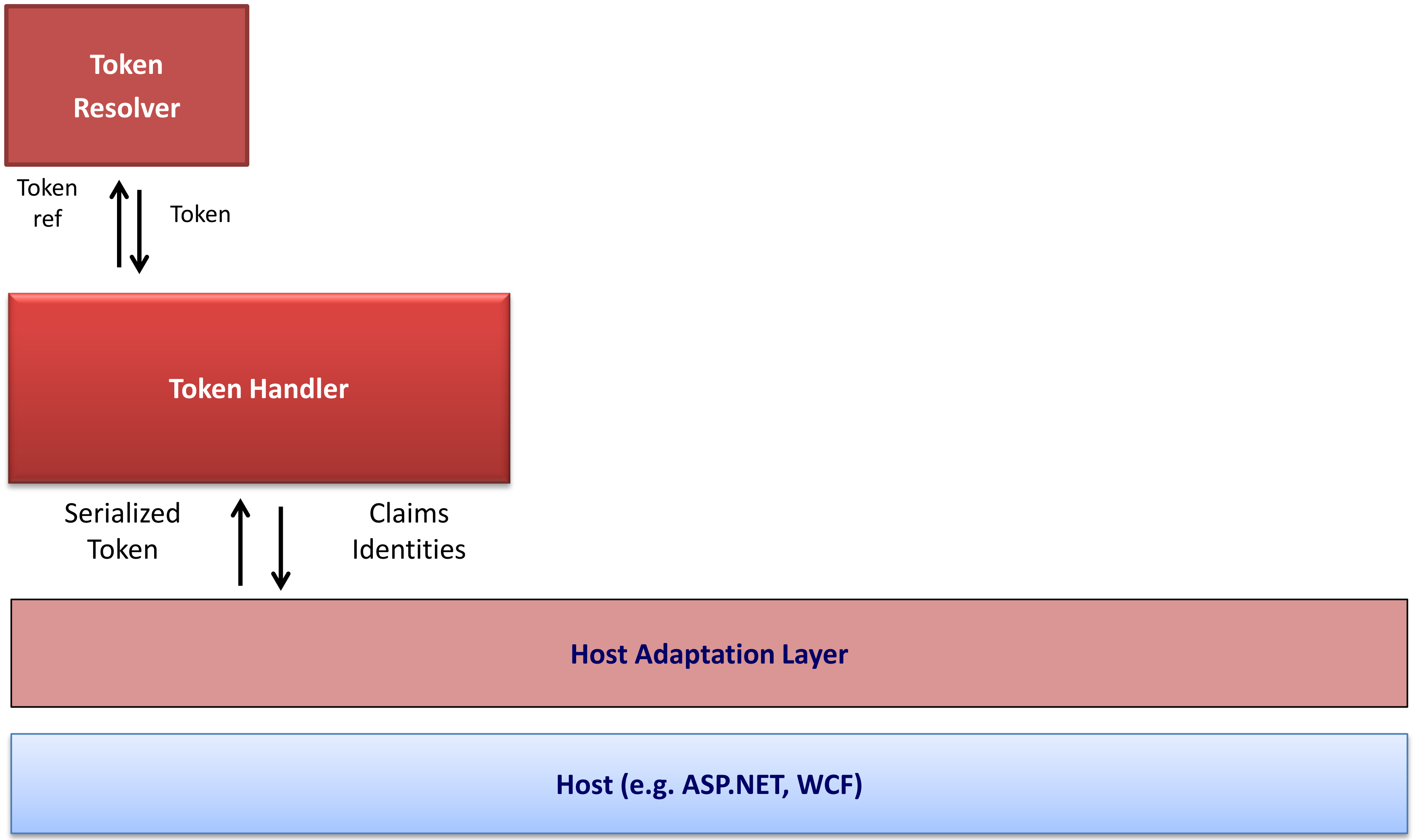
- Create a new ACS tenant
- Configure a new RP
- Activate support for Google IdP
- Define some rules
- Change relying party to use this new tenant

- Start at https://portal.appfabriclabs.com/

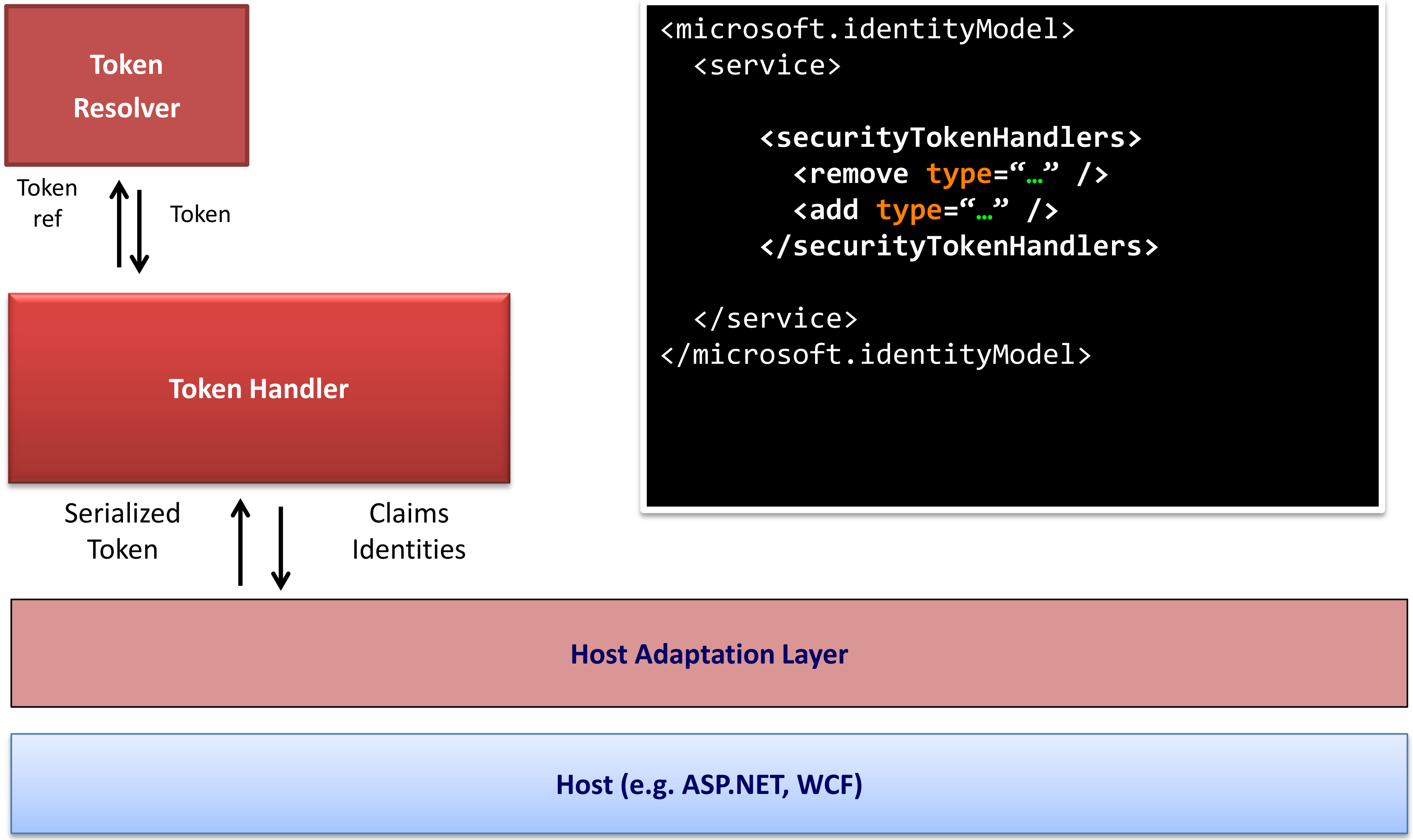
WIF Consumer Pipeline



WIF Consumer Pipeline

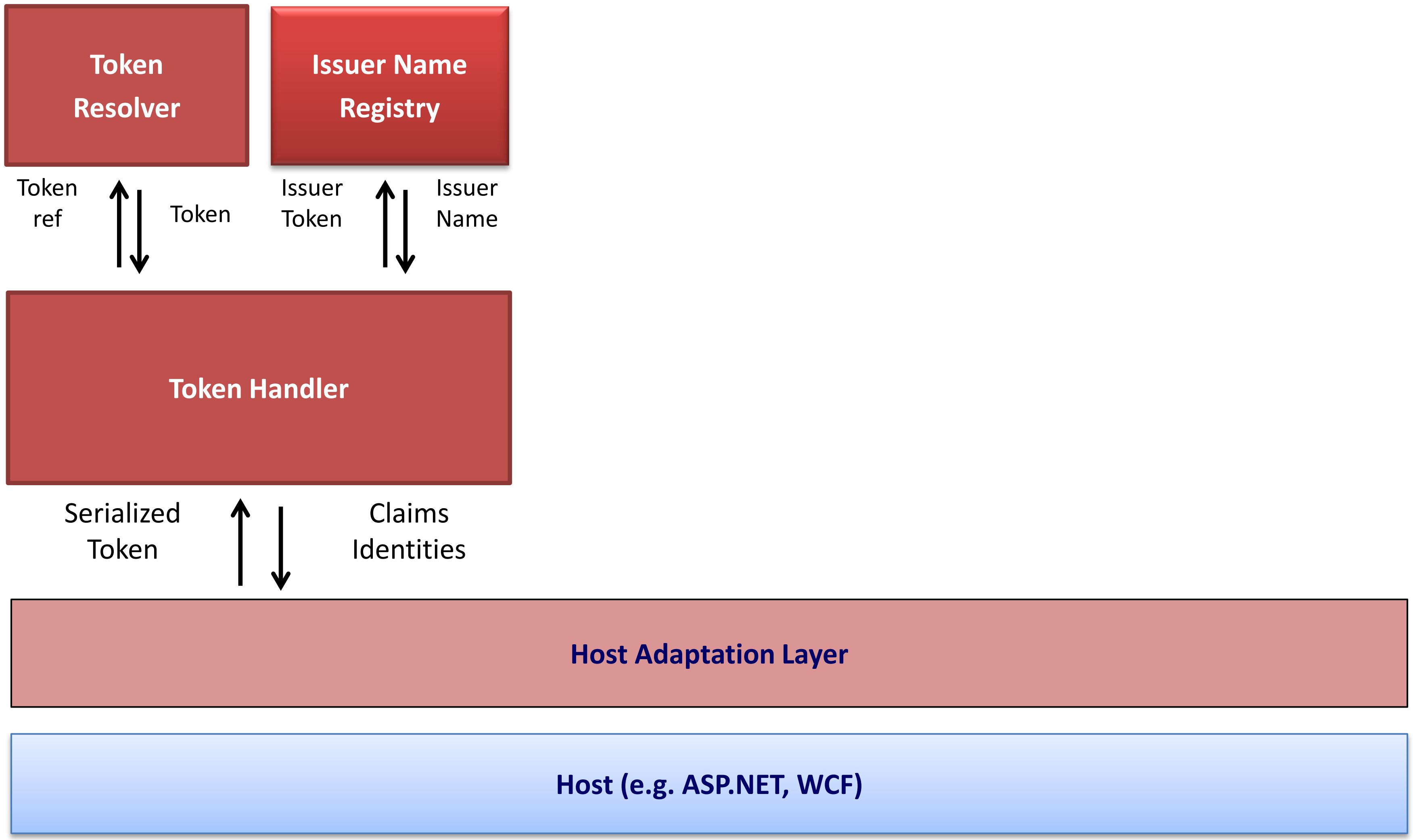


WIF Consumer Pipeline

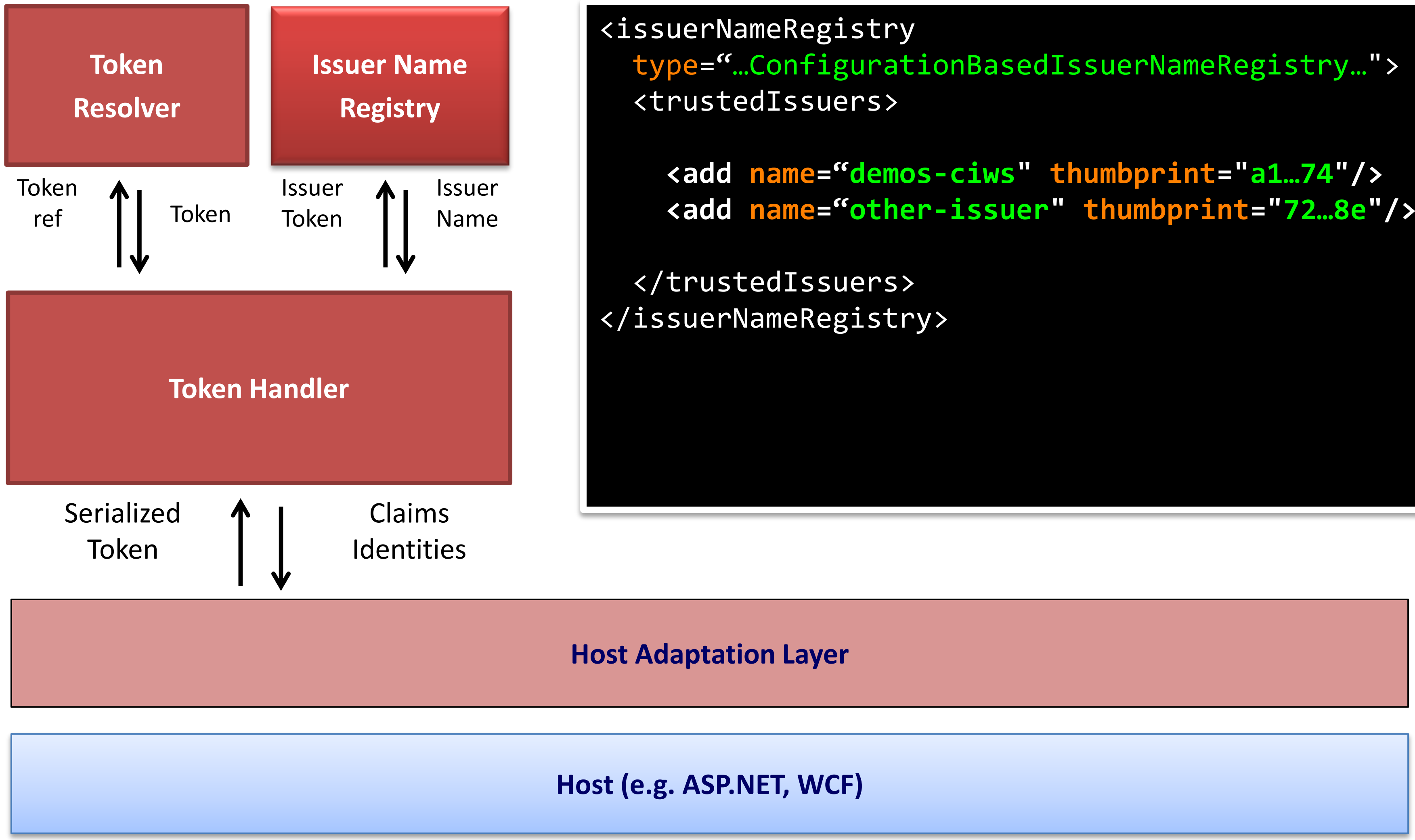


```
<microsoft.identityModel>
  <service>

      <securityTokenHandlers>
        <remove type="…" />
        <add type="…" />
      </securityTokenHandlers>

  </service>
</microsoft.identityModel>
```

WIF Consumer Pipeline



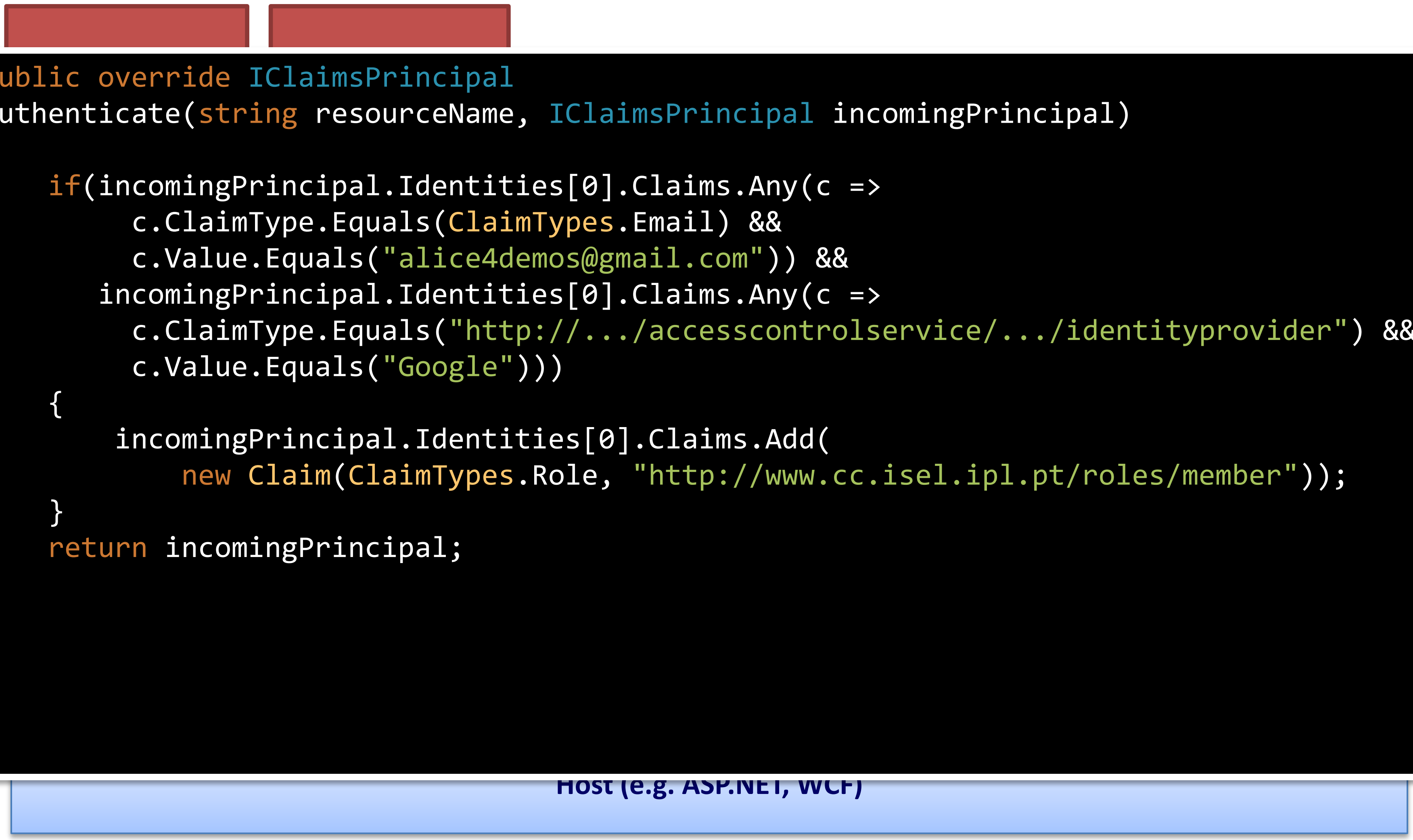
WIF Consumer Pipeline



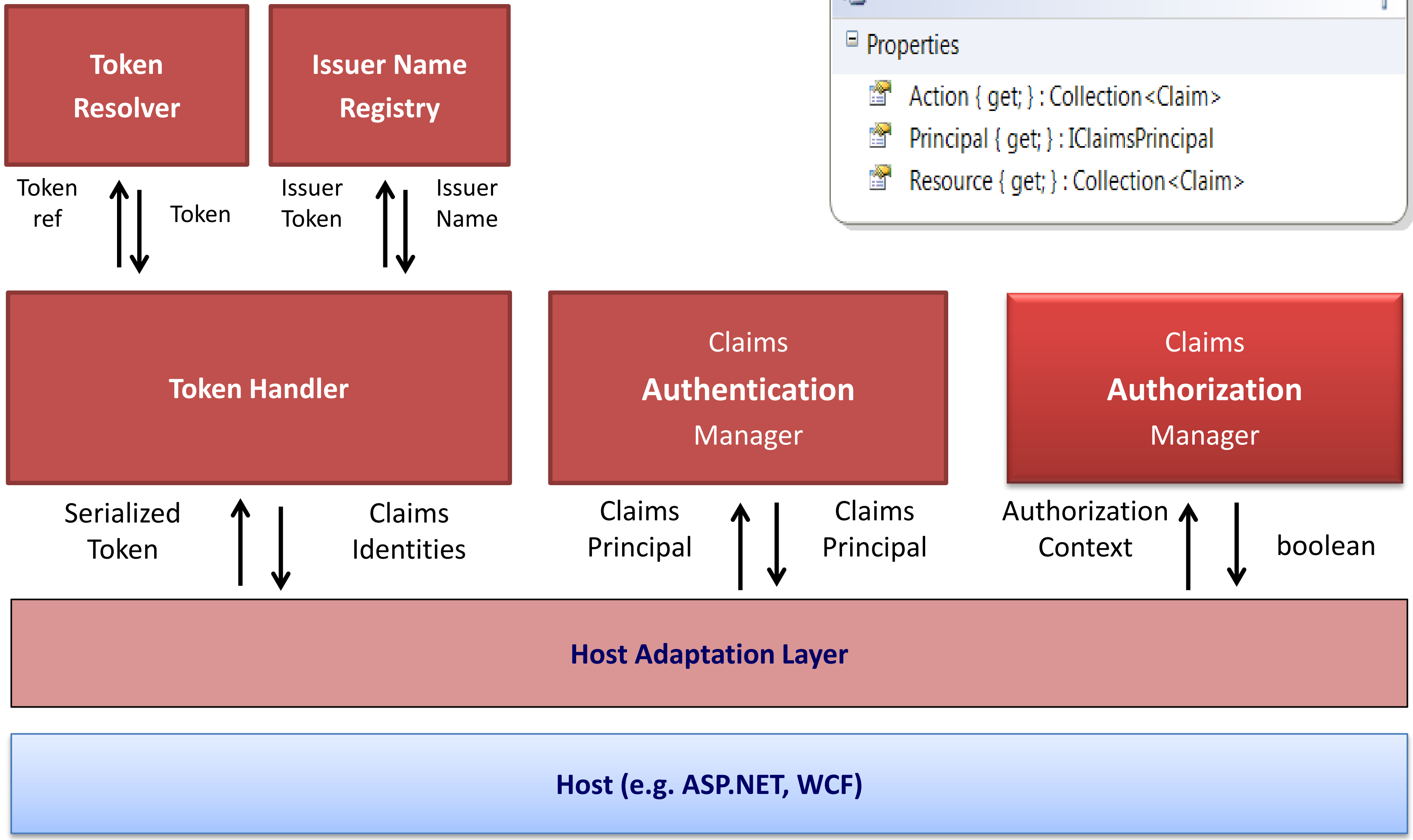
```
<issuerNameRegistry
  type="…ConfigurationBasedIssuerNameRegistry…">
  <trustedIssuers>

    <add name="demos-ciws" thumbprint="a1…74"/>
    <add name="other-issuer" thumbprint="72…8e"/>

  </trustedIssuers>
</issuerNameRegistry>
```

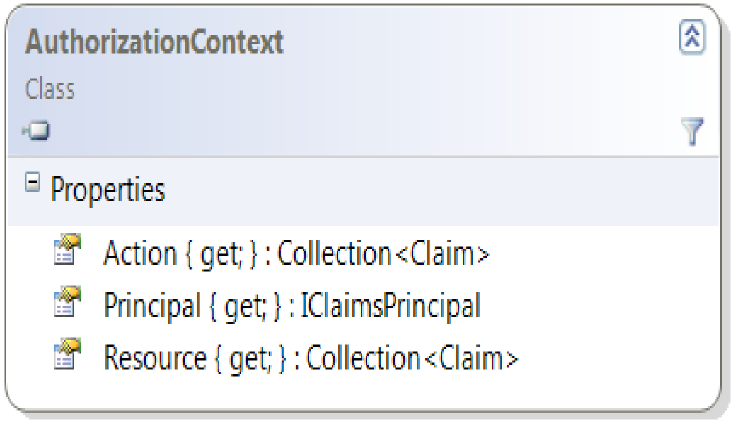
WIF Consumer Pipeline

```
public override IClaimsPrincipal
Authenticate(string resourceName, IClaimsPrincipal incomingPrincipal)
{
    if(incomingPrincipal.Identities[0].Claims.Any(c =>
         c.ClaimType.Equals(ClaimTypes.Email) &&
         c.Value.Equals("alice4demos@gmail.com")) &&
       incomingPrincipal.Identities[0].Claims.Any(c =>
         c.ClaimType.Equals("http://.../accesscontrolservice/.../identityprovider") &&
         c.Value.Equals("Google")))
    {
        incomingPrincipal.Identities[0].Claims.Add(
            new Claim(ClaimTypes.Role, "http://www.cc.isel.ipl.pt/roles/member"));
    }
    return incomingPrincipal;
}
```



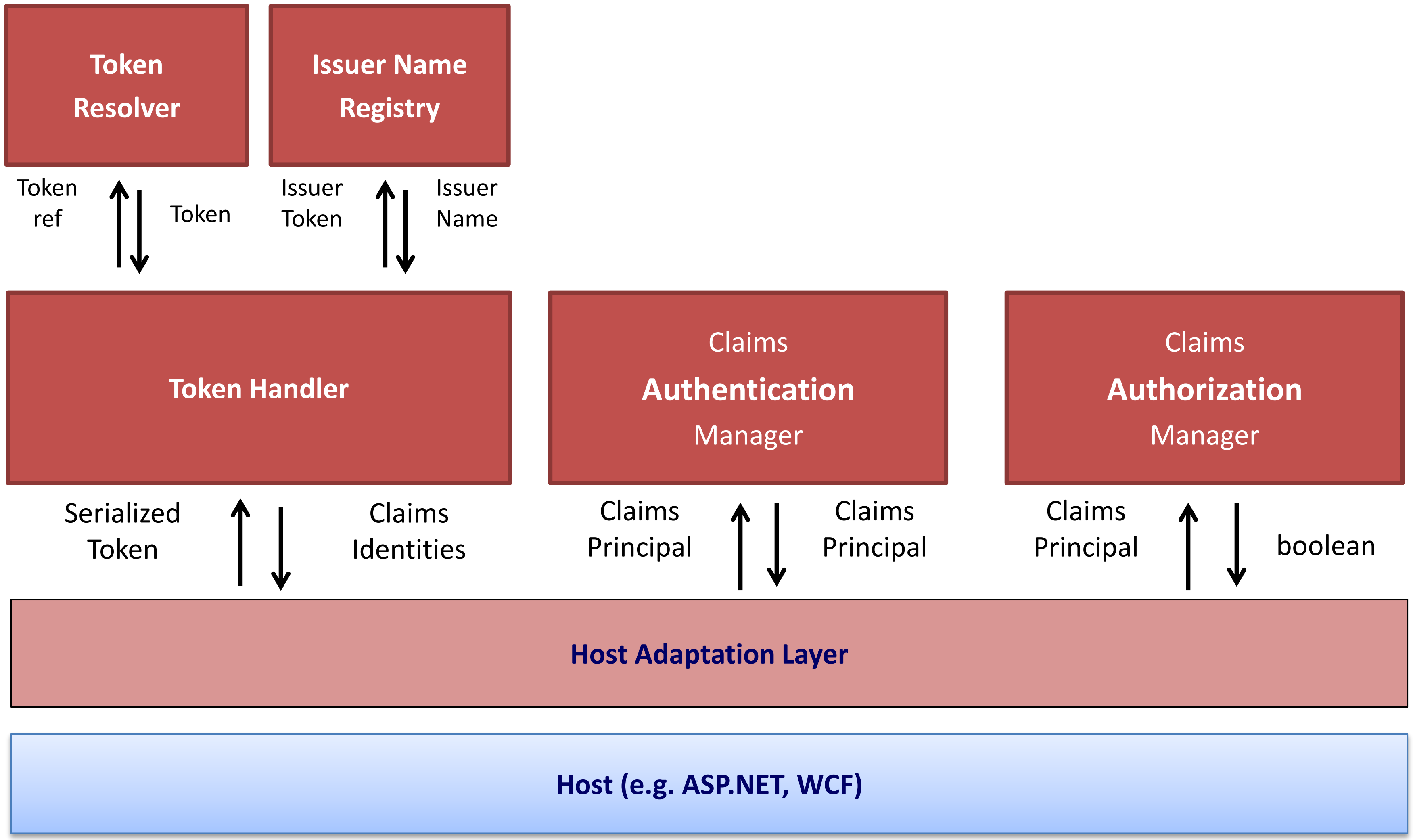
WIF Consumer Pipeline



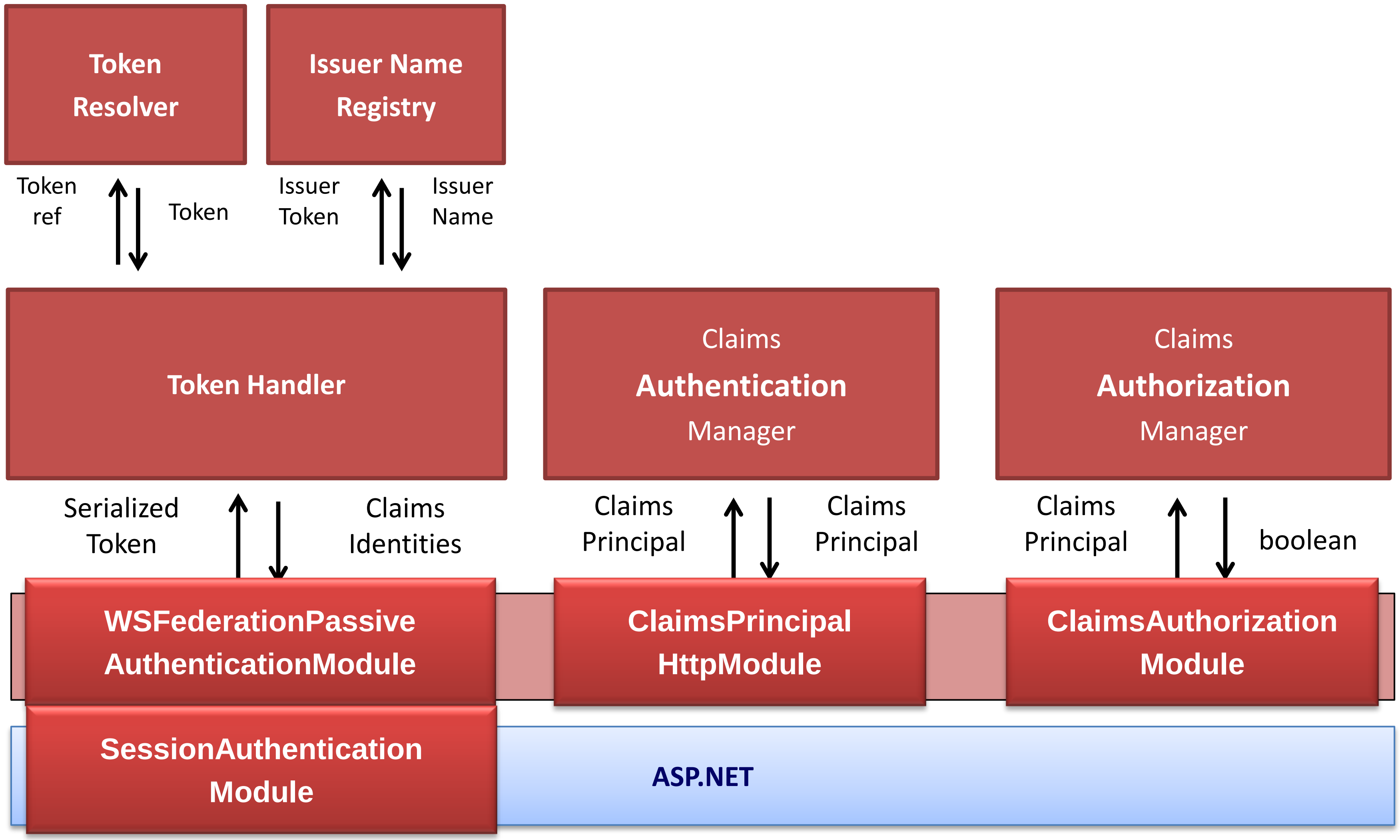

AuthorizationContext
Class

Properties
- Action { get; } : Collection<Claim>
- Principal { get; } : IClaimsPrincipal
- Resource { get; } : Collection<Claim>

WIF Consumer Pipeline

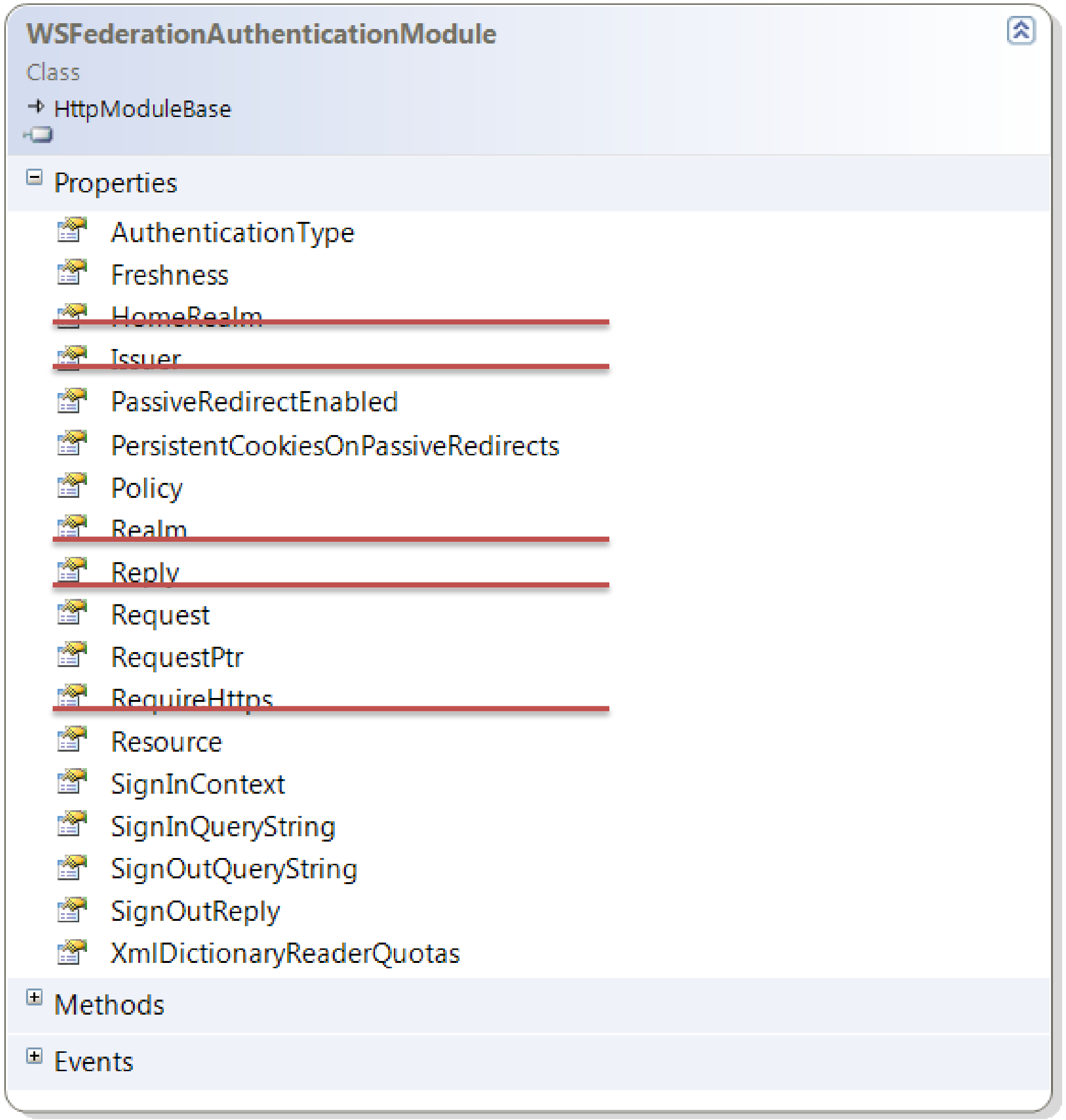


WIF Consumer Pipeline (ASP.NET)



WSFAM

FAM: properties



WSFederationAuthenticationModule
Class
HttpModuleBase

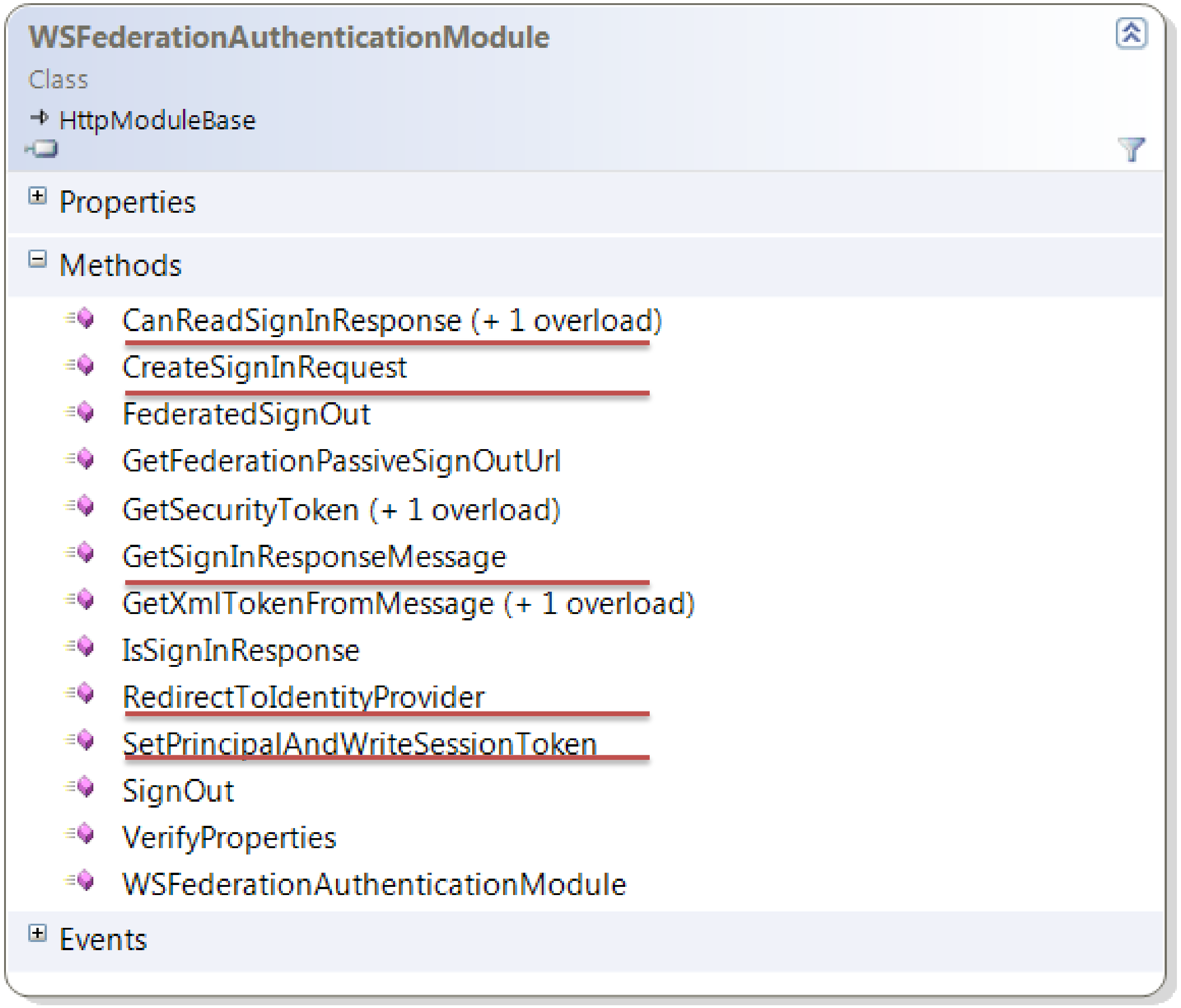
Properties

- AuthenticationType
- Freshness
- ~~HomeRealm~~
- ~~Issuer~~
- PassiveRedirectEnabled
- PersistentCookiesOnPassiveRedirects
- Policy
- ~~Realm~~
- ~~Reply~~
- Request
- RequestPtr
- ~~RequireHttps~~
- Resource
- SignInContext
- SignInQueryString
- SignOutQueryString
- SignOutReply
- XmlDictionaryReaderQuotas

Methods

Events

FAM: Public methods


WSFederationAuthenticationModule
Class
HttpModuleBase

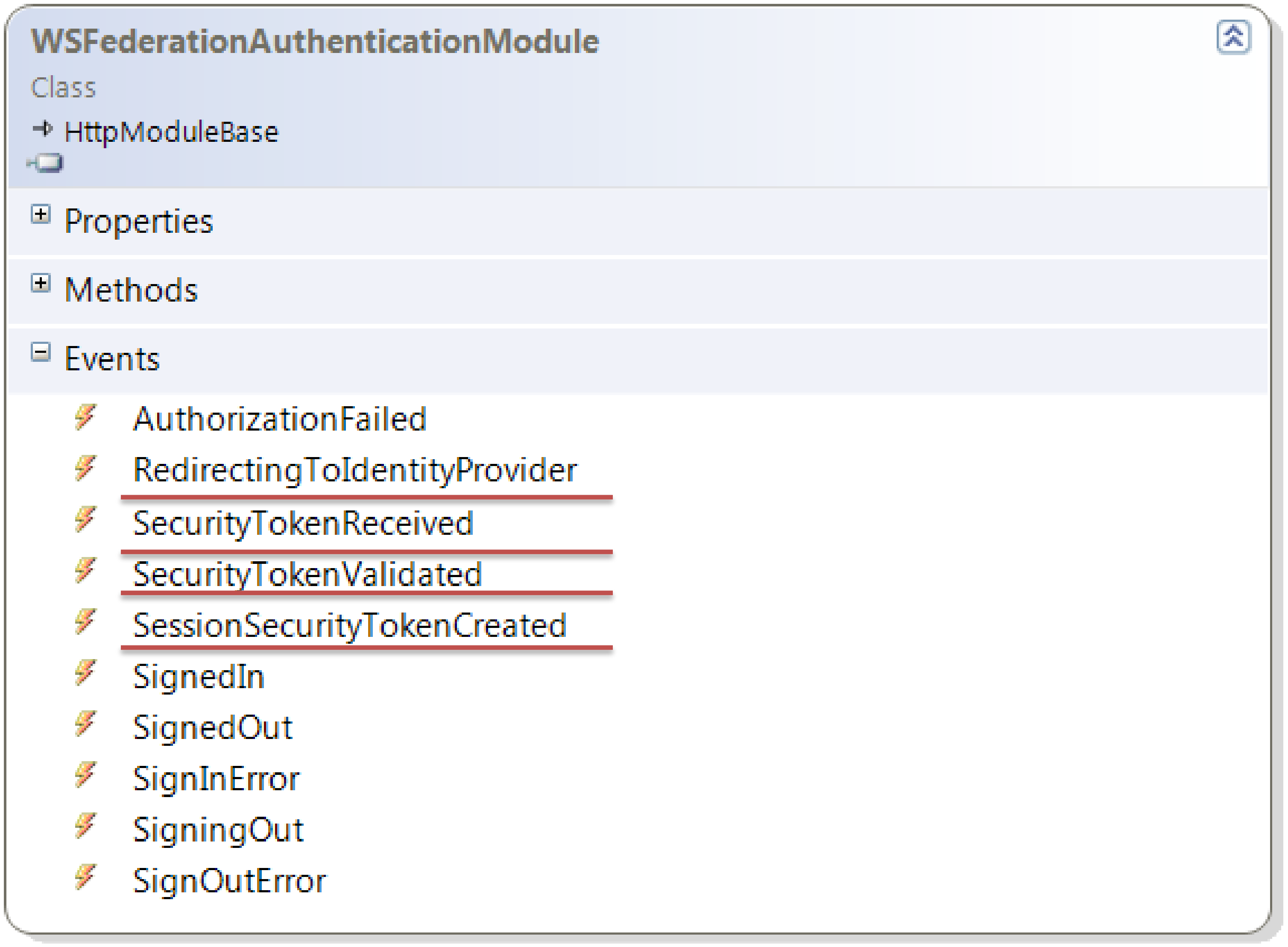
Properties

Methods

- CanReadSignInResponse (+ 1 overload)
- CreateSignInRequest
- FederatedSignOut
- GetFederationPassiveSignOutUrl
- GetSecurityToken (+ 1 overload)
- GetSignInResponseMessage
- GetXmlTokenFromMessage (+ 1 overload)
- IsSignInResponse
- RedirectToIdentityProvider
- SetPrincipalAndWriteSessionToken
- SignOut
- VerifyProperties
- WSFederationAuthenticationModule

Events

FAM: events



WSFederationAuthenticationModule
Class
→ HttpModuleBase

⊞ Properties

⊞ Methods

⊟ Events

- AuthorizationFailed
- RedirectingToIdentityProvider
- SecurityTokenReceived
- SecurityTokenValidated
- SessionSecurityTokenCreated
- SignedIn
- SignedOut
- SignInError
- SigningOut
- SignOutError

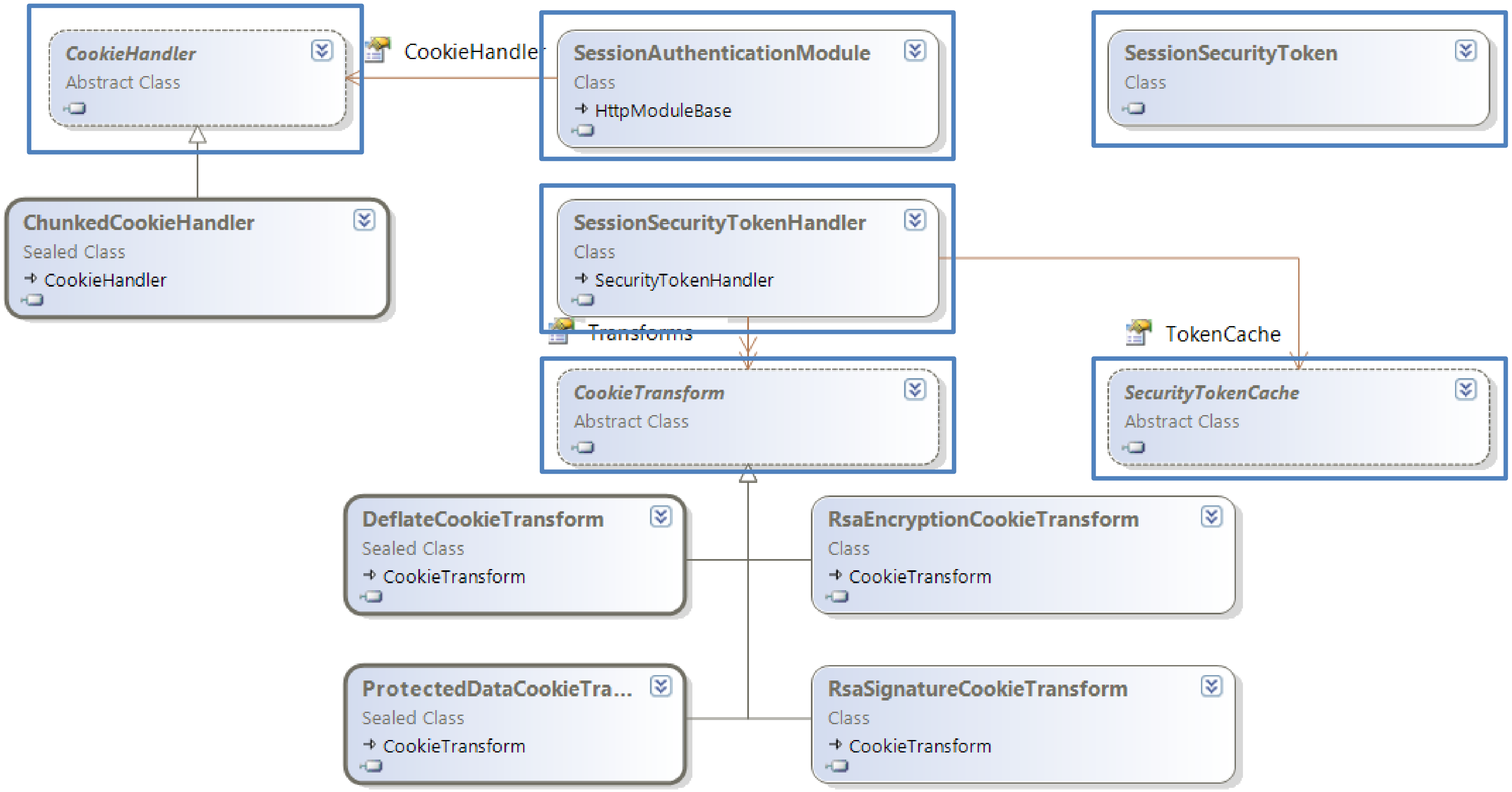
ASP.NET SESSION MANAGEMENT

Session management

- The sign in response contains the issued token
  - The next HTTP requests don’t
  - Preserve authentication info between requests
- Cookies and session tokens

- **SessionAuthenticationModule**
  - Handles **OnAuthenticateRequest**
  - Tries to read session token from cookie
  - Sets request identity from token claims
  - Updates session token and cookies

Session management

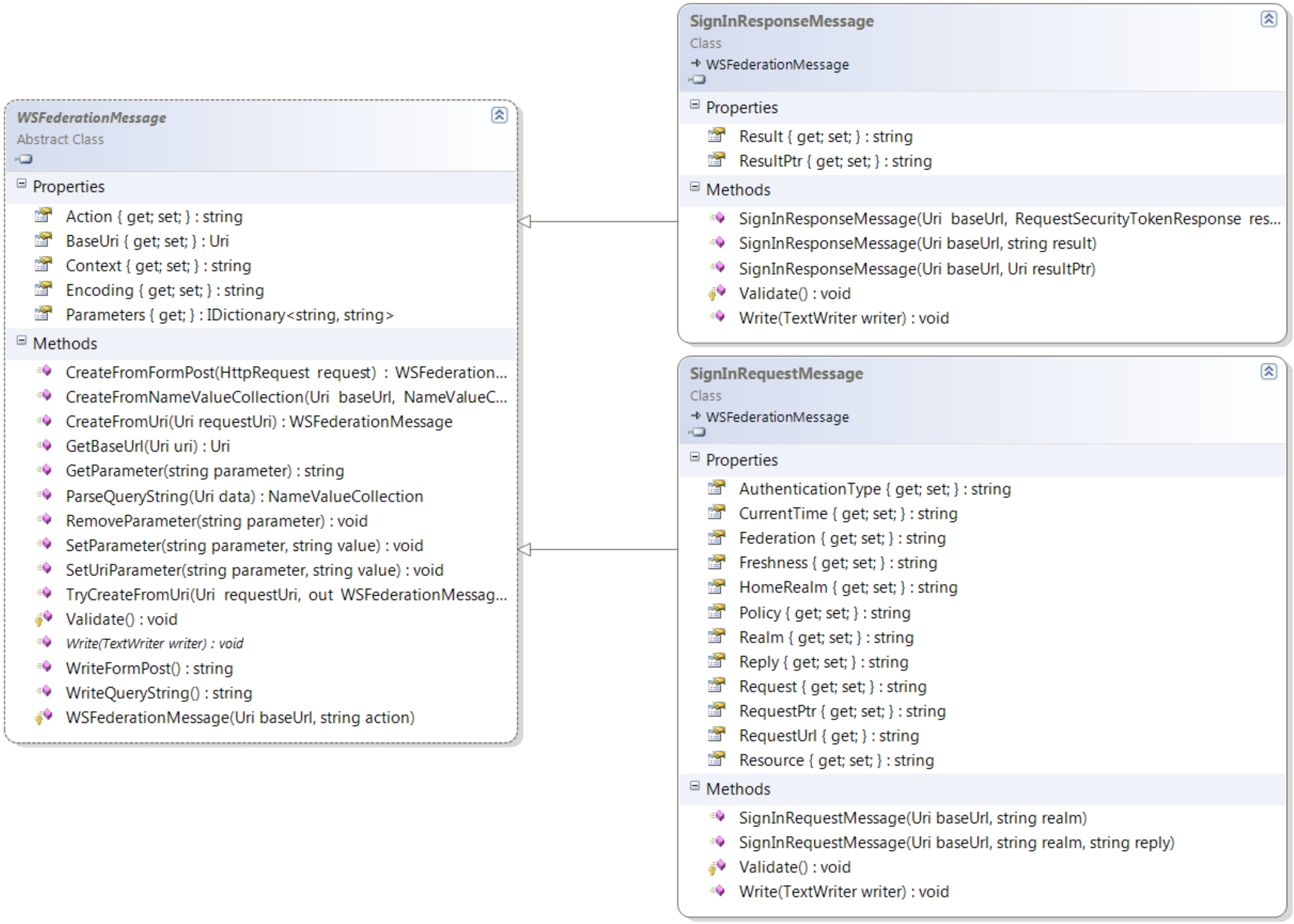


Network Load Balancing (NLB)

- **ProtectDataCookieTransform** used by default
  - Data Protection API – machine/account bound
  - Not suitable for NLB environment

- Solution
  - Use the RSA based cookie transforms
  - Encryption and signature
  - Use the same key pair in all machines
  - Typical in HTTPS scenarios

Federation messages



WSFederationMessage
Abstract Class

Properties
- Action { get; set; } : string
- BaseUri { get; set; } : Uri
- Context { get; set; } : string
- Encoding { get; set; } : string
- Parameters { get; } : IDictionary<string, string>

Methods
- CreateFromFormPost(HttpRequest request) : WSFederation...
- CreateFromNameValueCollection(Uri baseUrl, NameValueC...
- CreateFromUri(Uri requestUri) : WSFederationMessage
- GetBaseUrl(Uri uri) : Uri
- GetParameter(string parameter) : string
- ParseQueryString(Uri data) : NameValueCollection
- RemoveParameter(string parameter) : void
- SetParameter(string parameter, string value) : void
- SetUriParameter(string parameter, string value) : void
- TryCreateFromUri(Uri requestUri, out WSFederationMessag...
- Validate() : void
- *Write(TextWriter writer) : void*
- WriteFormPost() : string
- WriteQueryString() : string
- WSFederationMessage(Uri baseUrl, string action)

SignInResponseMessage
Class
→ WSFederationMessage

Properties
- Result { get; set; } : string
- ResultPtr { get; set; } : string

Methods
- SignInResponseMessage(Uri baseUrl, RequestSecurityTokenResponse res...
- SignInResponseMessage(Uri baseUrl, string result)
- SignInResponseMessage(Uri baseUrl, Uri resultPtr)
- Validate() : void
- Write(TextWriter writer) : void

SignInRequestMessage
Class
→ WSFederationMessage

Properties
- AuthenticationType { get; set; } : string
- CurrentTime { get; set; } : string
- Federation { get; set; } : string
- Freshness { get; set; } : string
- HomeRealm { get; set; } : string
- Policy { get; set; } : string
- Realm { get; set; } : string
- Reply { get; set; } : string
- Request { get; set; } : string
- RequestPtr { get; set; } : string
- RequestUrl { get; } : string
- Resource { get; set; } : string

Methods
- SignInRequestMessage(Uri baseUrl, string realm)
- SignInRequestMessage(Uri baseUrl, string realm, string reply)
- Validate() : void
- Write(TextWriter writer) : void

Controls

- FederatedPassiveSignIn
- FederatedPassiveSignInStatus

WIF CONFIGURATION

WIF configuration

- Host independent configuration
  - Token processing
  - Authorization manager
  - **<service>** element
  - **ServiceConfiguration** type
- ASP.NET specific configuration
  - FAM and SAM modules; cookie handling
  - **<federatedAuthentication>** element
  - **FederatedAuthentication** static class

Service Configuration

- Token processing
  - Audience restrictions
  - Security token handlers
- Issuer naming
- Certificate validation
  - None | Peer | Chain | Custom
  - Revocation mode
- Certificate resolution
  - External
  - Service
- Managers (generic pipeline)
  - Authentication
  - Authorization

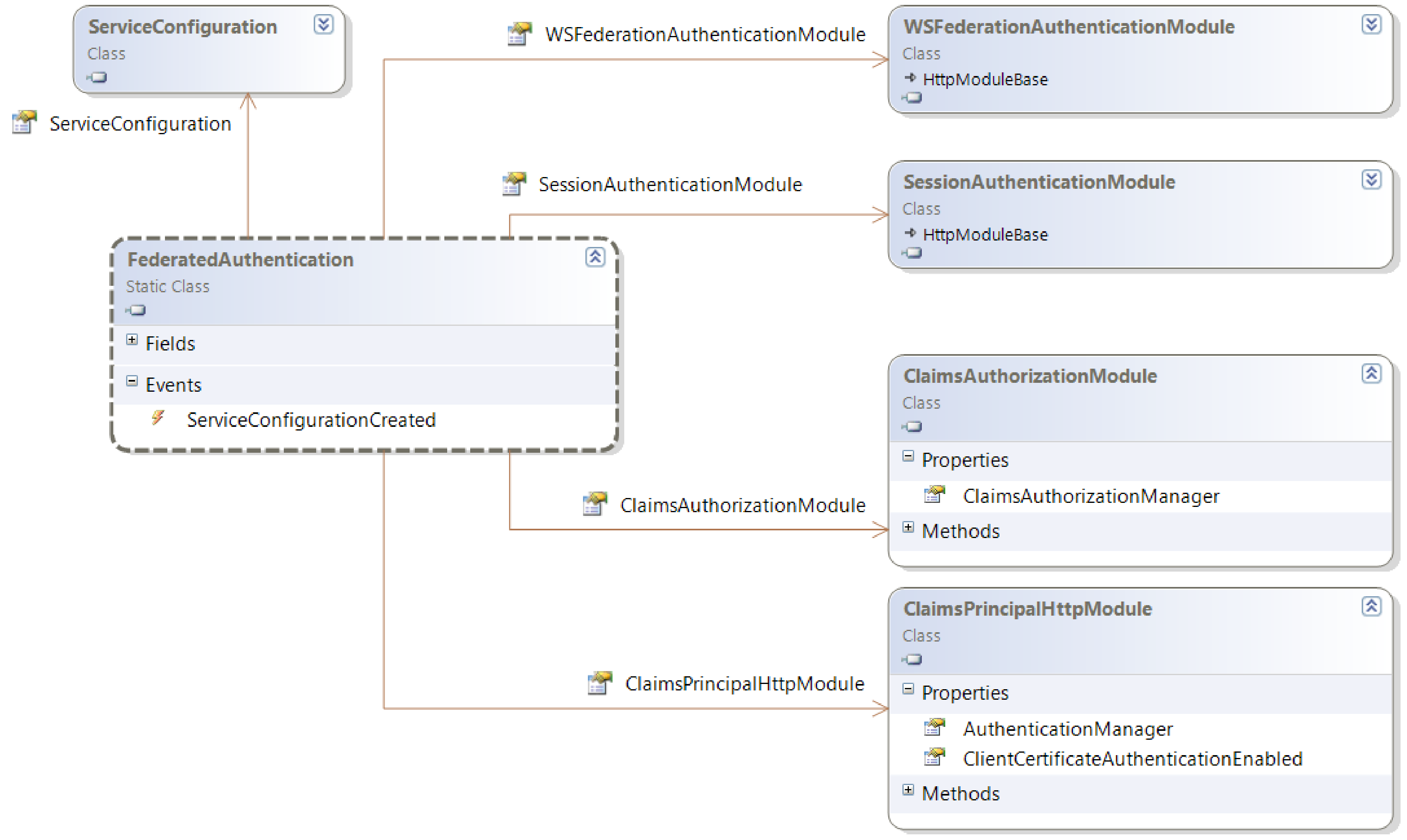
Service configuration

- XML

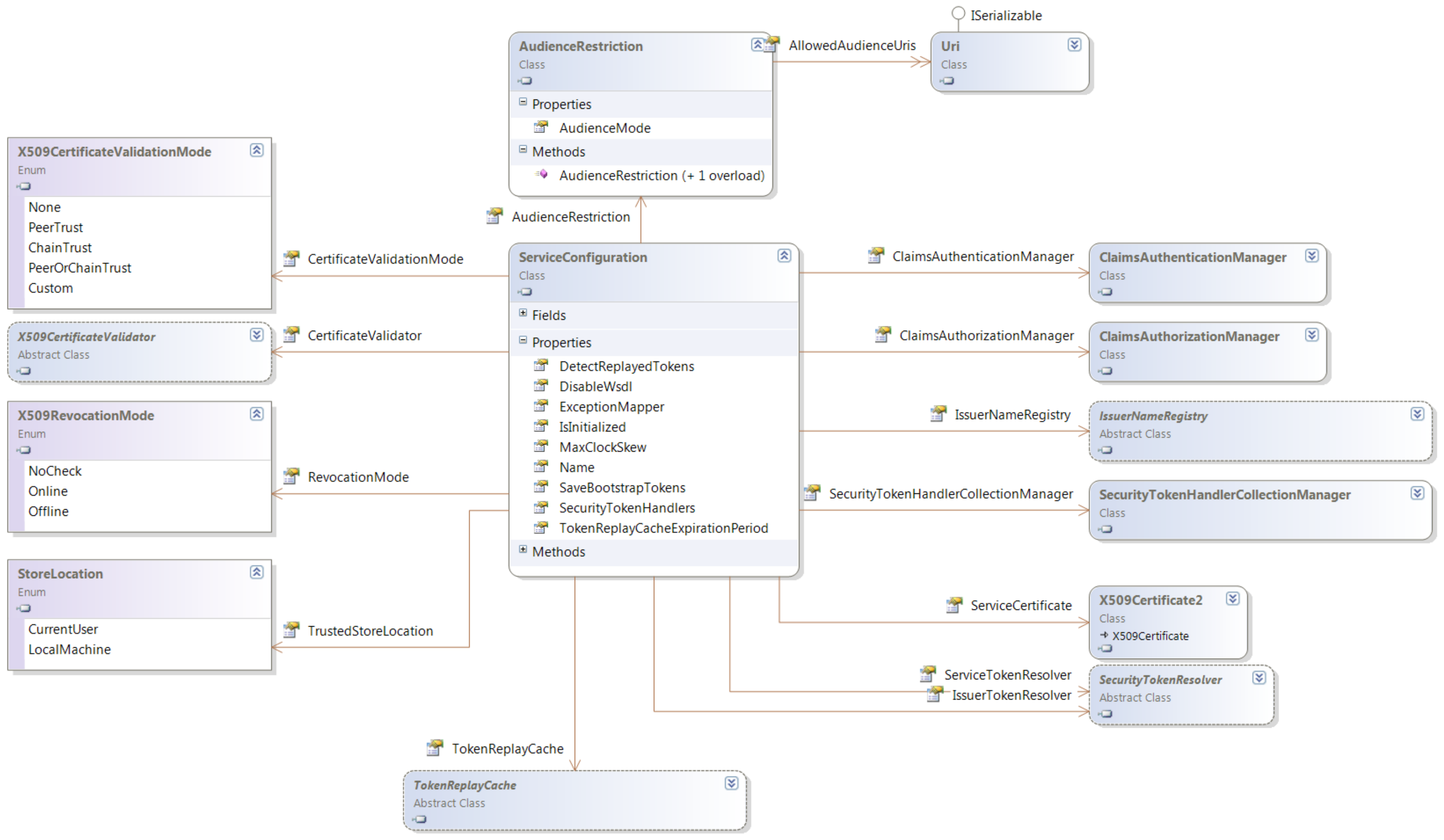
```
<microsoft.identityModel>
  <service>
    <federatedAuthentication>
    [default configuration]
  <service name = “configname”>
    [alternate configuration]
```

- Code
  - **ServiceConfiguration** ctor(configname)
  - **FederatedAuthentication** static methods

FederatedAuthentication


ServiceConfiguration
Class
WSFederationAuthenticationModule
WSFederationAuthenticationModule
Class
HttpModuleBase
ServiceConfiguration
SessionAuthenticationModule
SessionAuthenticationModule
Class
HttpModuleBase
FederatedAuthentication
Static Class
Fields
Events
ServiceConfigurationCreated
ClaimsAuthorizationModule
Class
Properties
ClaimsAuthorizationManager
Methods
ClaimsAuthorizationModule
ClaimsPrincipalHttpModule
Class
Properties
AuthenticationManager
ClientCertificateAuthenticationEnabled
Methods
ClaimsPrincipalHttpModule

ServiceConfiguration class



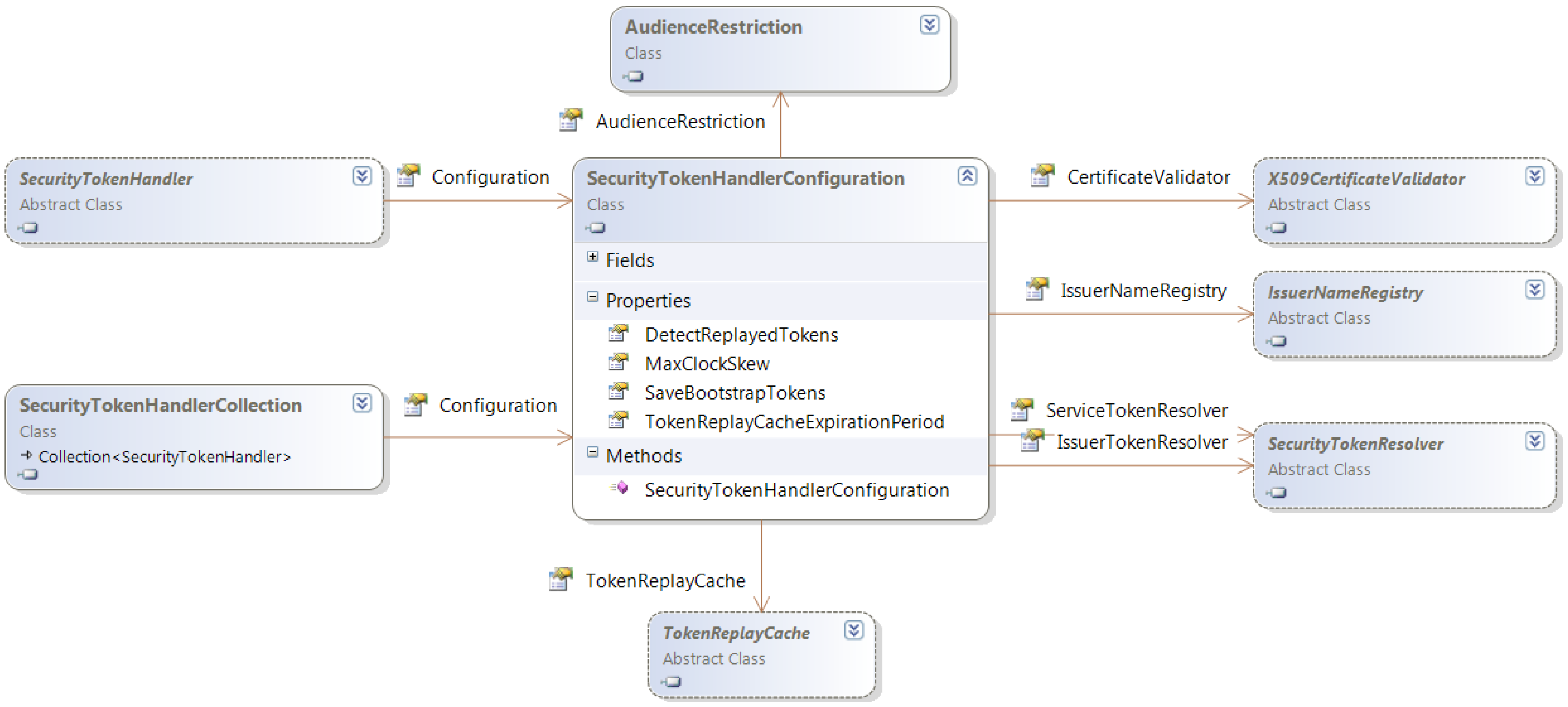
Token handlers

- XML

```
<securityTokenHandlers>
   <securityTokenHandlerConfiguration> …
<securityTokenHandlers name = “configname”>
   <securityTokenHandlerConfiguration> …
```

- Code
  - Named collections – handler collection manager

SecurityTokenHandlerConfiguration



Configuration extensibility

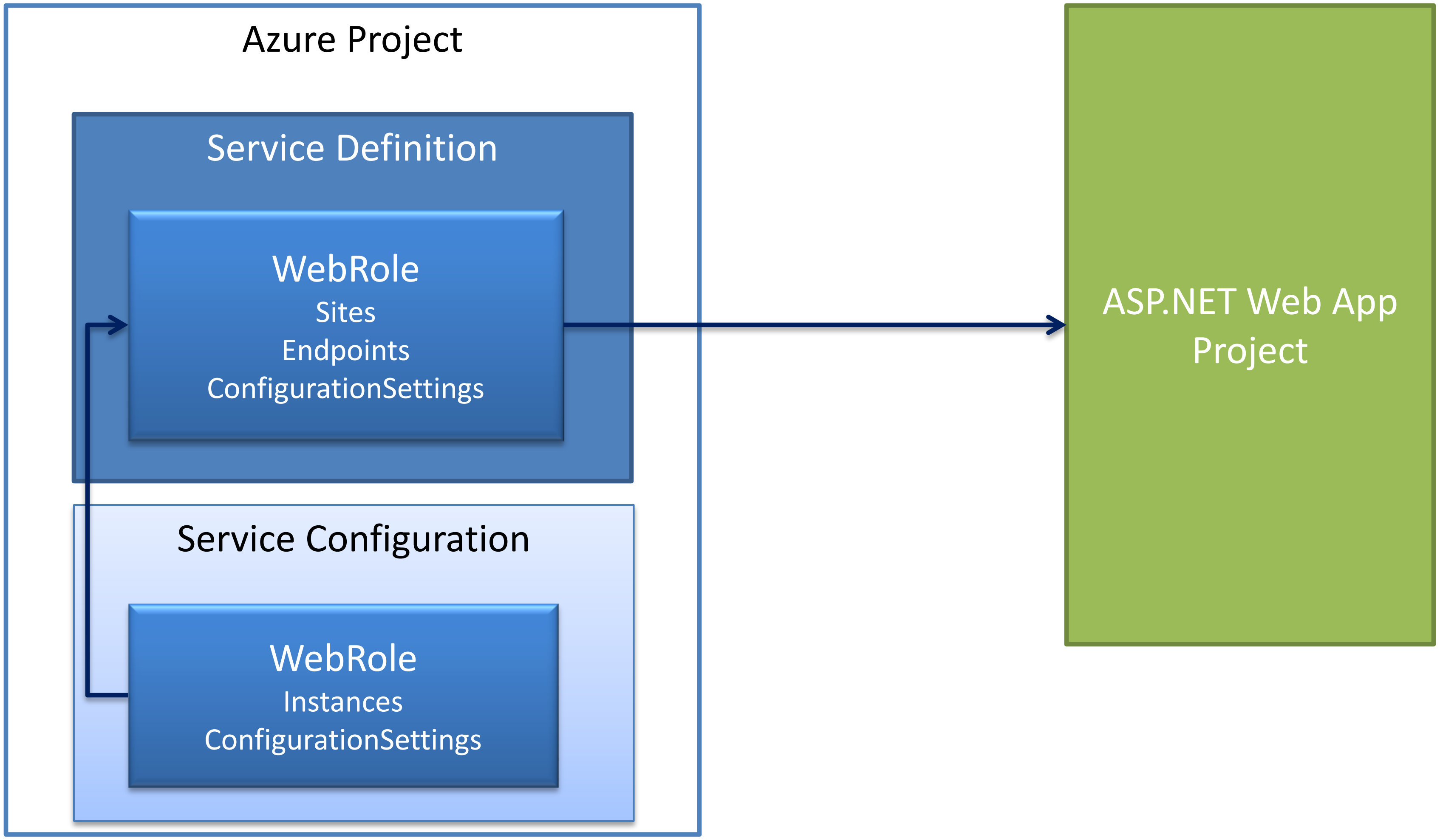
- Inner XML passed as XmlNodeList on the ctor
- Elements
  - IssuerNameRegistry
  - ClaimsAuthorizationManager
  - SecurityTokenHandler
- Example

```
<issuerNameRegistry type="ConfigurationBasedIssuerNameRegistry, ...">
   <trustedIssuers>
      <add name=“issuername" thumbprint=“certificate hash" />
   </trustedIssuers>
</issuerNameRegistry>
```

Hands-On-Lab 3

- Explore WIF and ASP.NET extensibility
  - NLB scenarios and session management
  - Explicit authentication
    - ASP.NET WebForms and WIF controls
    - ASP.NET MVC account controller
  - Home realm selection
  - Authentication and Authorization Managers

Azure project



Definition and configuration

- Definition file (.csdef)
  - Static
  - Endpoints
  - Settings names
  - …
- Configuration file (.cscfg)
  - Dynamic
  - Settings values
  - Instance count
  - …

Azure Project

- Build output
  - Package file (.cspkg)
  - Configuration file (.cscfg)

- Package file
  - Compressed archive
  - Service model
  - Project outputs

Azure and WIF

- Certificate management
- HTTPS endpoints
- WIF assemblies
- Host names and environment differences
- Load balancing

Certificate management (1)

- Certificates and private keys upload
  - Via the management portal only
  - Administrator task (private key management)
  - Service scoped
  - PFX (PKCS #12) format
    - Certificate and private key
    - Certificate only

Certificate management (2)

- Role definition

```
<Certificates>
    <Certificate name=“rp-ciws.cloudapp.net”
      storeLocation=“LocalMachine” storeName=“My” />
 <Endpoints>
   <InputEndpoint name=“…” protocol=“https” port=“433”
    certificate=“rp-ciws.cloudapp.net” />
```

- Role configuration

```
<Certificate name=“rp-ciws.cloudapp.net”
  thumbprint=“78…CC” />
```

$\boxed{\text{“78…CC”}} \rightarrow$ Matches certificate uploaded by admin

WIF assemblies

- WIF **is not installed** on the Azure platform

- Solutions
  - Upload Microsoft.IdentityModel.dll
    - Copy Local =“true”
    - Some limitations
  - Use a role startup task
    - Installs WIF runtime on role startup

Role startup

- Role definition

```
<WebRole name="…">
  <Startup>
    <Task commandLine="\\Startup\SetupWifRuntime.cmd"
      executionContext="elevated" taskType="simple" />
```

- Copy to output directory
  - SetupWifRuntime.bat

```
sc config wuauserv start= demand
wusa.exe "%~dp0Windows6.1-KB974405-x64.msu" /quiet /norestart
sc config wuauserv start= disabled
```

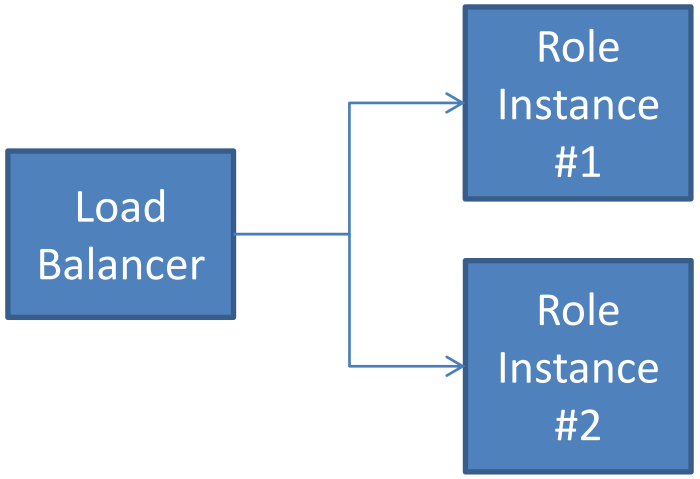
  - Windows6.1-KB974405-x64.msu (windows update)

Host names

- Three different environments
  - Local machine - Developer Fabric
  - Staging deploys – random host names
  - Production deploys – chosen host name
    - E.g. **rp-ciws.cloudapp.net**
- Federation with identity provider
  - Tied to a host name

Load balancing

- Role instances are behind a NLB
  - No “sticky sessions”
- Use custom session management
  - Cookie protection using RSA key
  - Custom session token cache



Hands-On-Lab 4

- Claims based relying party on Azure
  - Create Azure project
  - Add web role based on existing project
  - Configure certificates
  - Test on local machine – development fabric
  - Create hosted service
  - Upload certificates
  - Publish service
  - Test on Azure

WCF

- Same
  - WIF claims class model
  - WIF processing pipeline (token handlers, issuer registries, …)
  - SAML token format

- Different
  - Token transport protocols (WS-Trust, not WS-Federation)
  - Metadata (WSDL, WS-Policy and WS-SecurityPolicy)
  - Token binding (WS-Security and holder-of-key)

Protocols – passive scenario

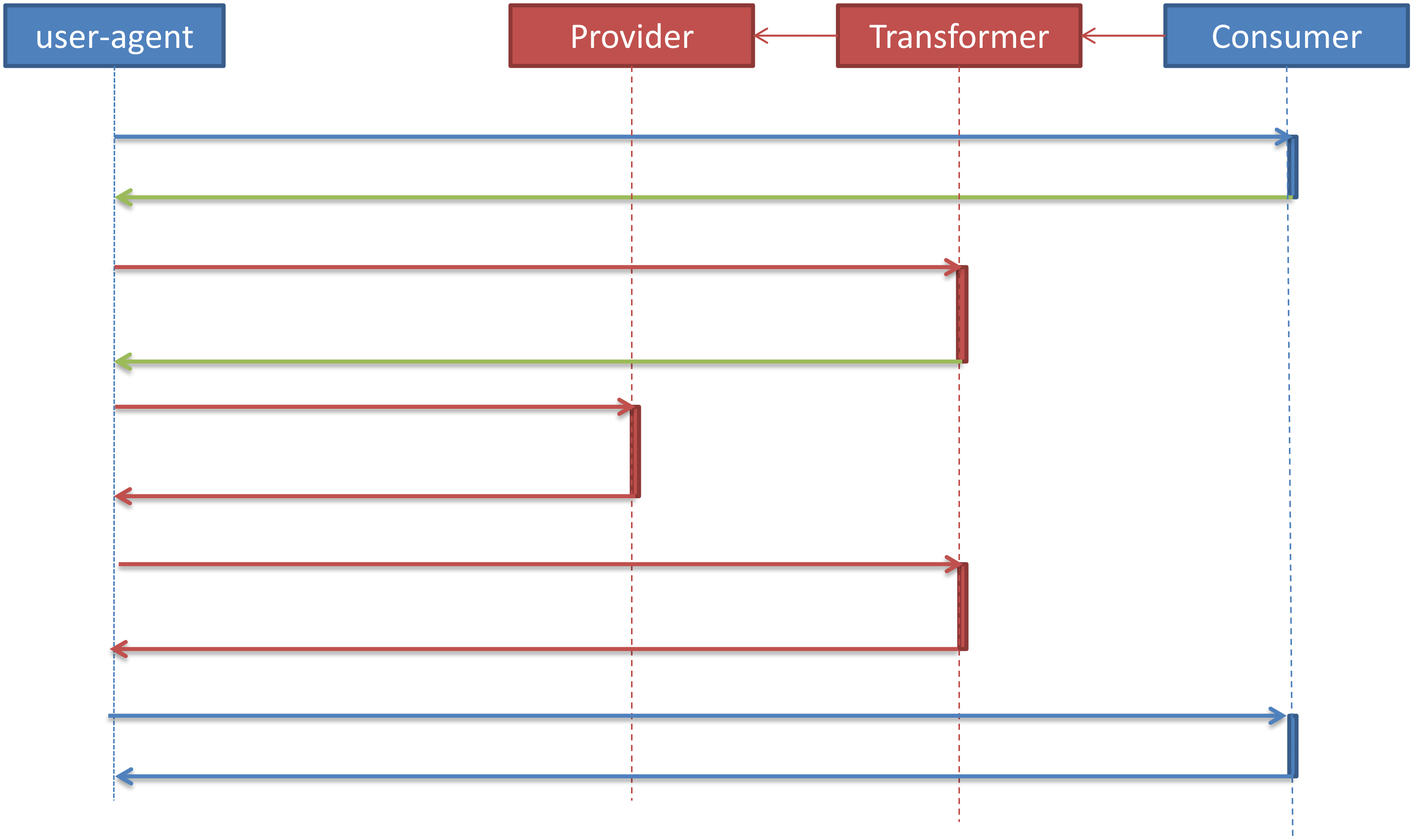
- User-agent follows HTTP and HTML+script specs.
  - HTTP redirections and auto-post forms
- No user-agent intervention on exchanged messages
  - No message protection (only transport protection)
  - No proof-of-possession
- Embedded interactions with UI
  - Custom authentication and identity disclosure interactions

Protocols – active scenario

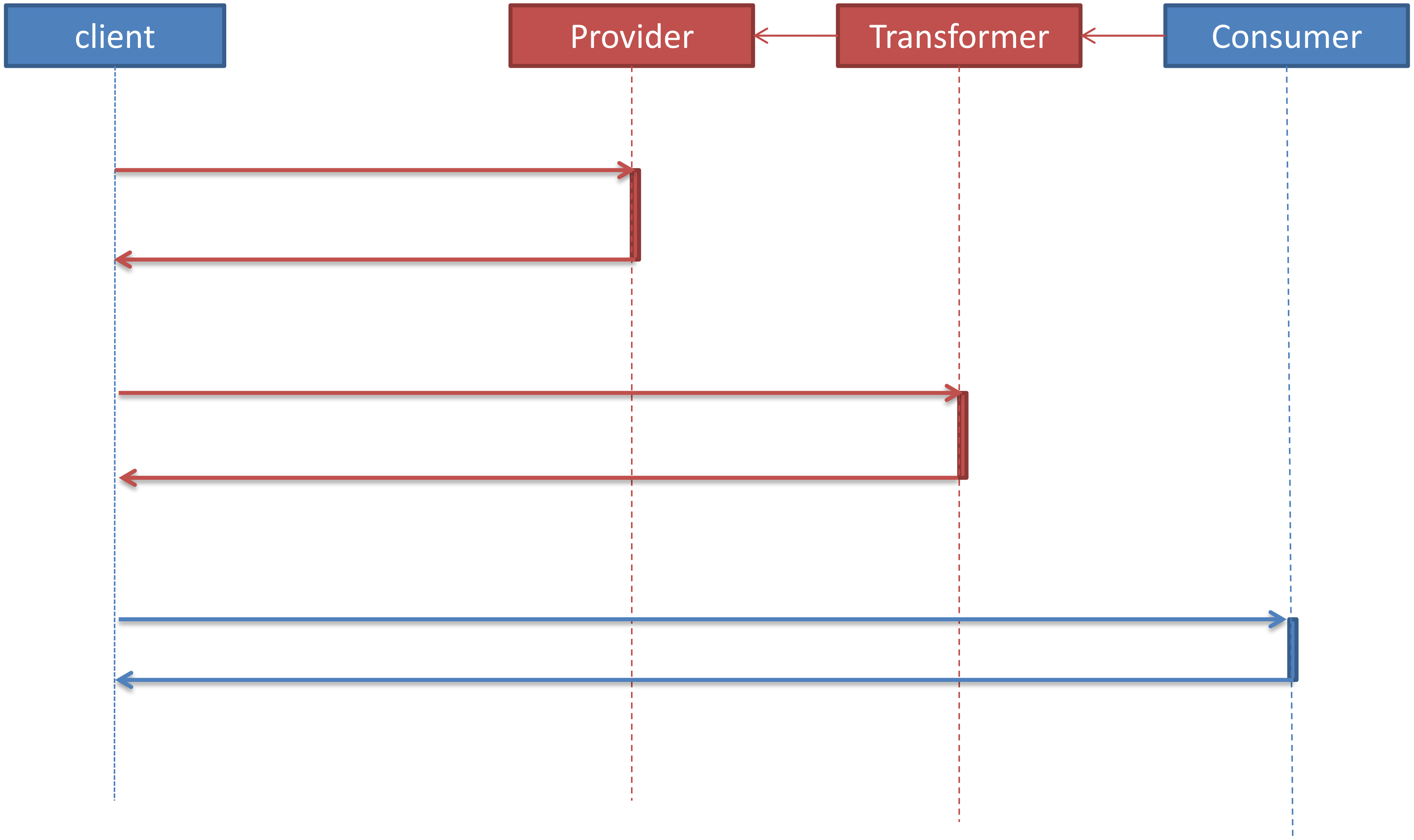
- Client has the initiative to obtain tokens
  - Guided by code, configuration or policy

- Client processes the messages
  - Message protection (signature and encryption)
  - Message authentication (token proof-of-possession)

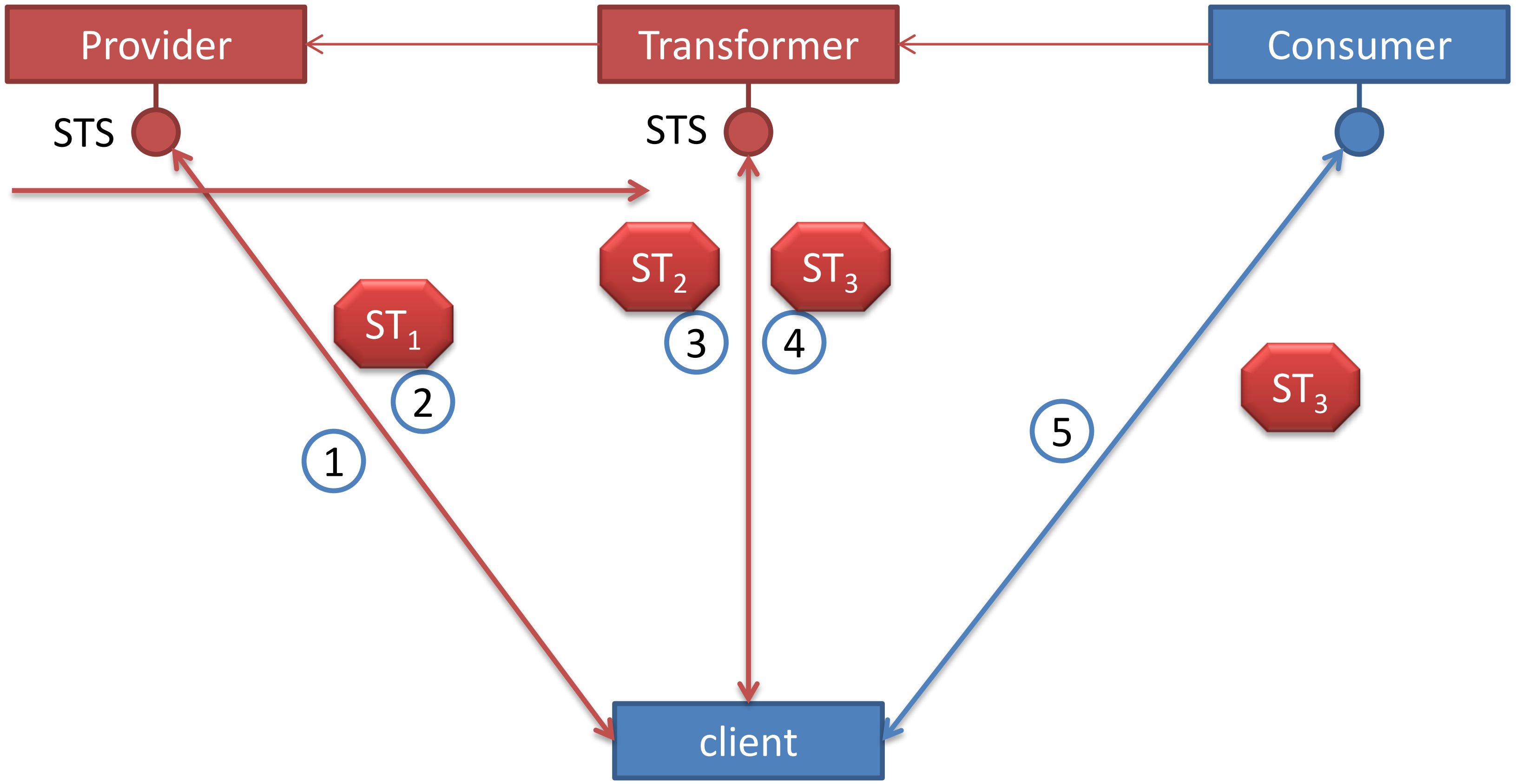
Passive

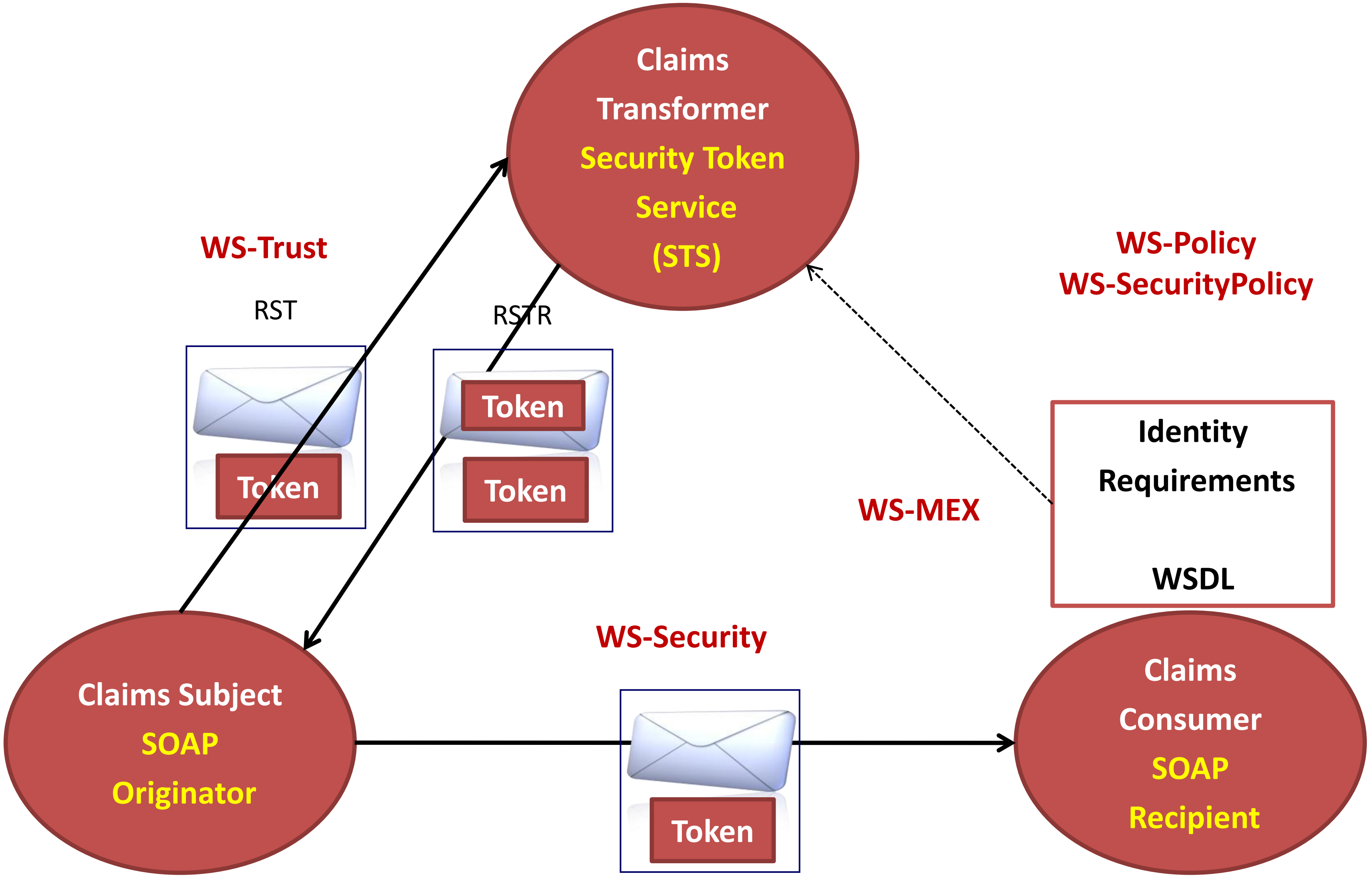


Active



Active

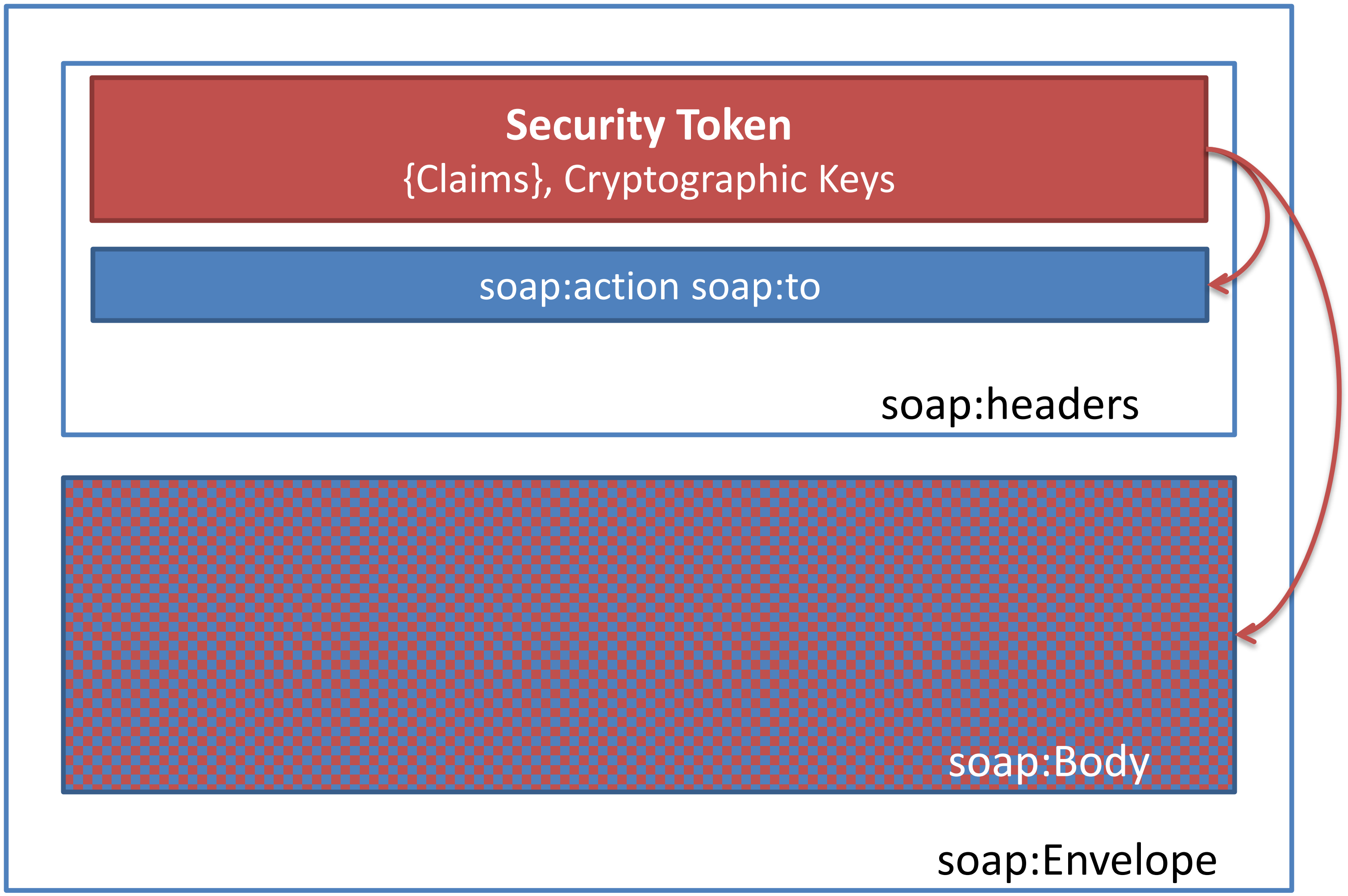

Provider
Transformer
Consumer
STS
STS
ST$_1$
ST$_2$
ST$_3$
ST$_3$
1
2
3
4
5
client


Claims
Transformer
Security Token
Service
(STS)
WS-Trust
RST
Token
RSTR
Token
Token
WS-Policy
WS-SecurityPolicy
Identity
Requirements
WSDL
WS-MEX
WS-Security
Claims Subject
SOAP
Originator
Token
Claims
Consumer
SOAP
Recipient

WS-*

- **WS-Security**
  - SOAP message protection XML-Signature/Encryption
  - SOAP message authentication using *binded* **security tokens**
- **WS-Trust**
  - Request-reply protocol for **requesting issued tokens**
  - Security Token Service (STS)
- **WS-SecurityPolicy**
  - Metadata containing the **token requirements** (issuer, claims) - **policies**
- **WS-MetadataExchange**
  - Protocol for obtaining the **policies**

WS-Security


Security Token
{Claims}, Cryptographic Keys
soap:action soap:to
soap:headers
soap:Body
soap:Envelope

WS-Trust messages

- **RequestSecurityToken** (RST) message
  - Requestor identity claims (security token)
  - Requested token characteristics:
    - Token type, claims, key type
  - Token consumer (**AppliesTo**)

- **RequestedSecurityTokenResponse** (RSTR) message
  - Issued Token
  - Proof of Possession information

Proof of Possession

- Symmetric key
  - Derived by both the requestor and the STS
  - **Encrypted to token consumer – STS must know the token consumer**
- Public key
  - Sent in the RST
  - Present in the issued token

WCF concepts

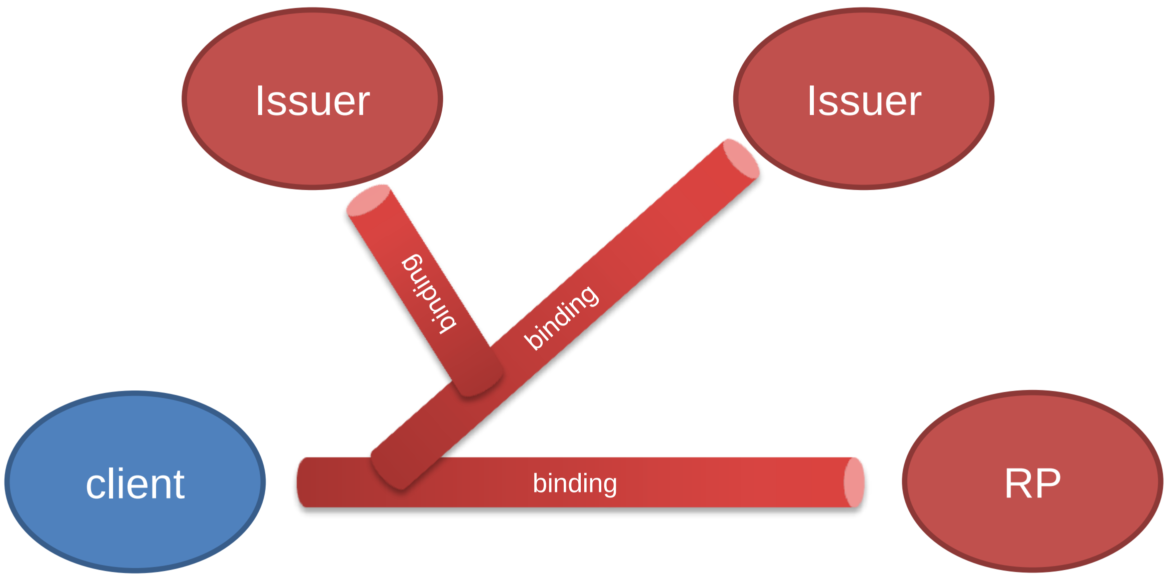
- Endpoint
  - Contract
  - **Binding**
  - Address
- **Behaviors**
  - Service scope
  - Endpoint scope
  - Contract scope

WCF and security

- Bindings - defines the security “type”
  - Publicized in the metadata
  - Token type and issuer, required claims, algorithms
  - Scoped to an endpoint
- Credentials – defines the security “values”
  - Usernames and passwords, certificates, certificate validation
  - Scope
    - **ServiceCredentials** (**IServiceBehavior**)
    - **ClientCredentials** (**IEndpointBehavior**)

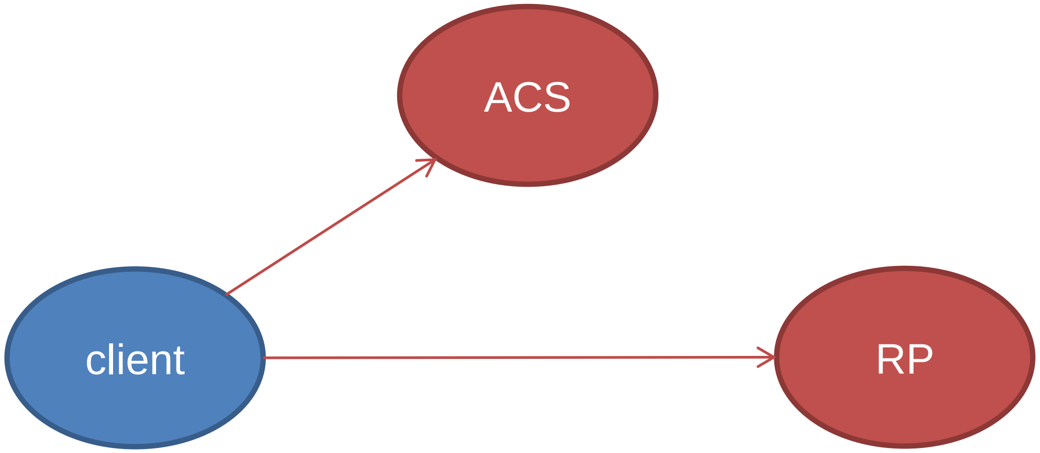
Federation bindings

- Federation bindings
  - Token request done automatically by the binding
  - Token request uses inner binding (issuer binding)



Simple scenario

- Using ACS as a claims provider
  - RP accepts issued token
  - Client uses username+password to authn with ACS



Simple scenario - Service

- Define binding and add endpoint

```
var binding = new WS2007FederationHttpBinding(
                                    WSFederationHttpSecurityMode.Message);
binding.Security.Message.IssuedKeyType = SecurityKeyType.SymmetricKey;
binding.Security.Message.IssuedTokenType =
                                    SecurityTokenTypes.OasisWssSaml2TokenProfile11;
host.AddServiceEndpoint(typeof(…), binding, “http://soap.rp.ciws");
```

Simple scenario - Service

- Define service certificate
  - Host service credentials

```
host.Credentials.ServiceCertificate.Certificate =
    X509StoreExt.GetCertificateFromStoreBySubjectName(
        StoreName.My,
        StoreLocation.LocalMachine,
        "soap.rp.ciws");
```

Simple scenario - Service

- Configure WIF

```
var wifconfig = new ServiceConfiguration();
wifconfig.CertificateValidationMode =
      X509CertificateValidationMode.None;
wifconfig.AudienceRestriction.AllowedAudienceUris.Add(
      new Uri("http://soap.rp.ciws"));
var issuerReg = wifconfig.IssuerNameRegistry as
      ConfigurationBasedIssuerNameRegistry;
 issuerReg.AddTrustedIssuer("67…DF",
               "https://demos-ciws.accesscontrol.appfabriclabs.com");

FederatedServiceCredentials.ConfigureServiceHost(host, wifconfig);
```

Simple scenario - Client

using (var cf = new ChannelFactory<…>(GetBinding2, address)) {

cf.Credentials.ServiceCertificate**.DefaultCertificate** =

X509StoreExt.GetCertificateFromStoreBySubjectName(

StoreName.My, StoreLocation.LocalMachine, "soap.rp.ciws");

cf.Credentials.ServiceCertificate.Authentication.**RevocationMode** =

X509RevocationMode.**NoCheck**;

cf.Credentials.UserName.**UserName** = "Alice";

cf.Credentials.UserName.**Password** = “…";

…

}

Simple scenario - Client

```
private static Binding GetBinding1(){
   var homeBinding = new
         WS2007HttpBinding(SecurityMode.TransportWithMessageCredential);
   homeBinding.Security.Message.ClientCredentialType =
         MessageCredentialType.UserName;
   homeBinding.Security.Message.NegotiateServiceCredential = true;
   homeBinding.Security.Message.EstablishSecurityContext = false;

   var rpBinding = new
         WS2007FederationHttpBinding(WSFederationHttpSecurityMode.Message);
   rpBinding.Security.Message.IssuedKeyType = SecurityKeyType.SymmetricKey;
   rpBinding.Security.Message.IssuedTokenType =
         SecurityTokenTypes.OasisWssSaml2TokenProfile11;

   rpBinding.Security.Message.IssuerAddress = new EndpointAddress(
           "https://demos-ciws.accesscontrol.appfabriclabs.com/v2/wstrust/13/username");
   rpBinding.Security.Message.IssuerBinding = homeBinding;
   return rpBinding;
}
```

Issuer characterization

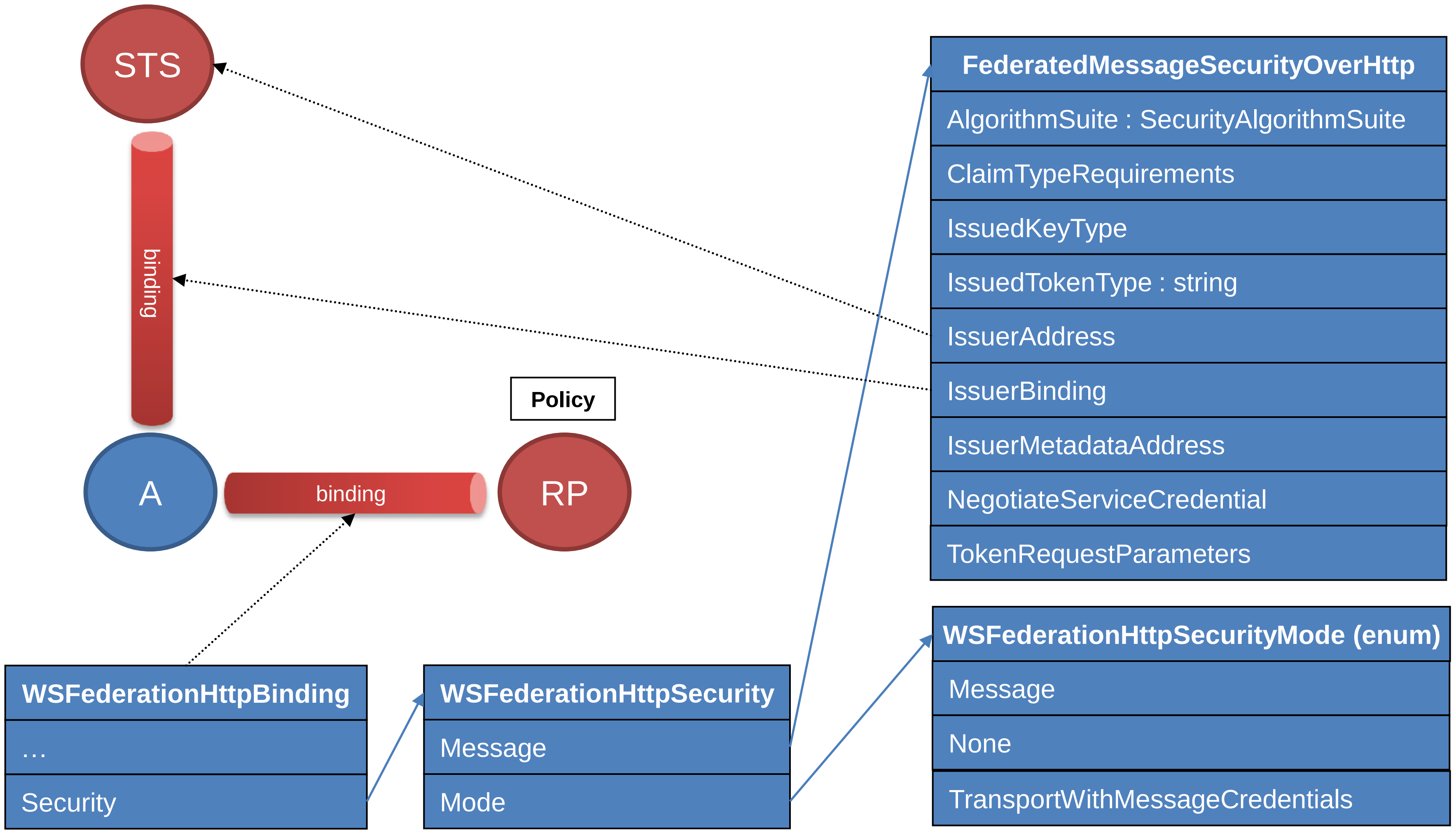
- Consumer defines issuer metadata address and endpoint address
  - Client can use **svcutil** tool or **MetadataResolver** class to obtain service endpoint
- Consumer defines issuer metadata address only
  - Client can use **svcutil** tool or **MetadataResolver** class to obtain service endpoint.
  - The first issuer compatible endpoint will be user; can be changed in config
- Consumer doesn’t define any issuer address
  - Client must configure issuer information
  - Binding’s issuer address and binding
  - Credentials’ local issuer address and binding

WCF and WIF Integration

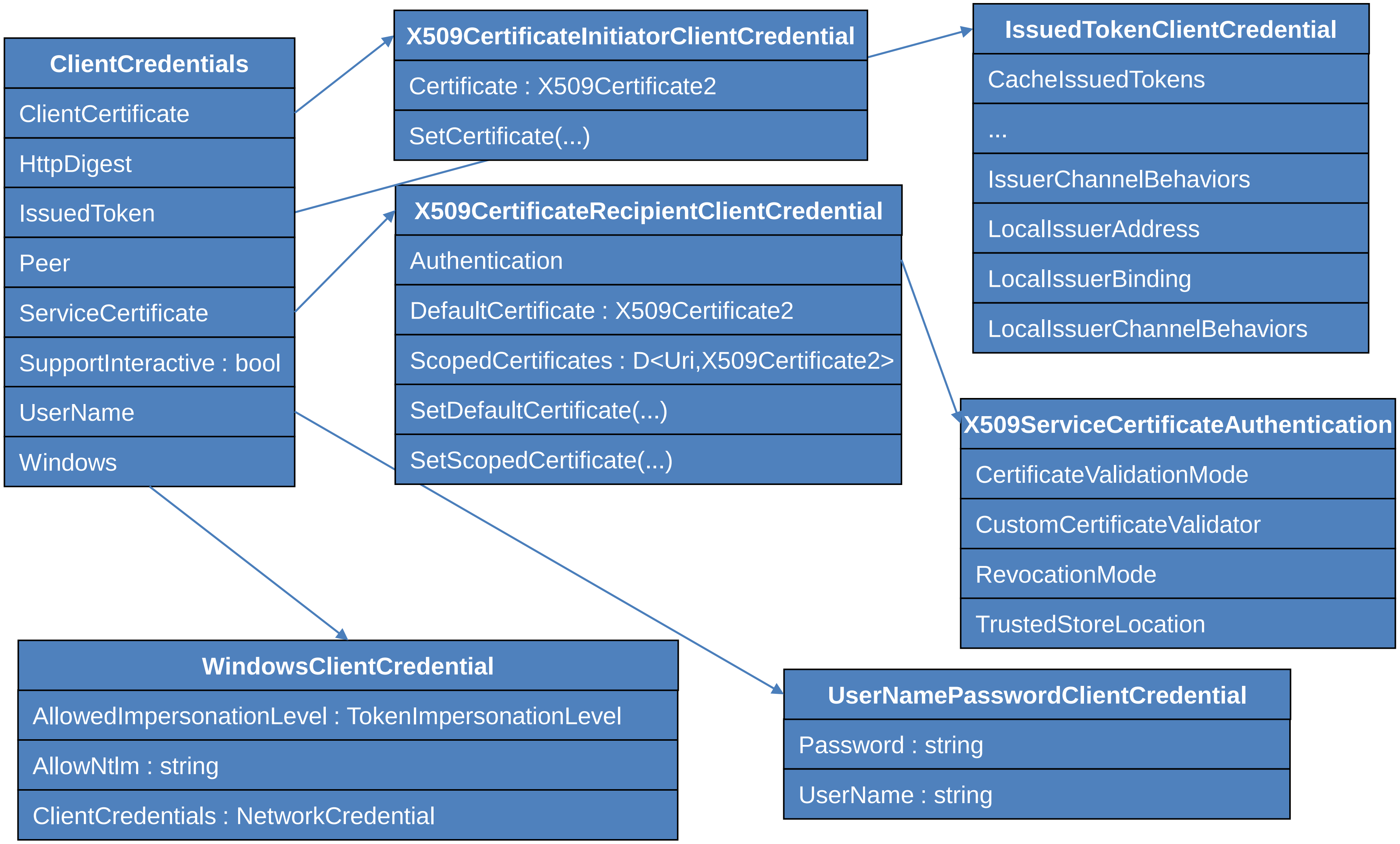
- Bindings
  - No changes, same model
  - E.g. WS2007FederationHttpBinding, …

- **ServiceCredentials**
  - Replaced by the WIF pipelining
    - Uses WCF extensibility points
  - Not compatible with WCF native model
    - Some **ServiceCredentials** settings don’t work
  - WIF claims class model and authn/authz managers

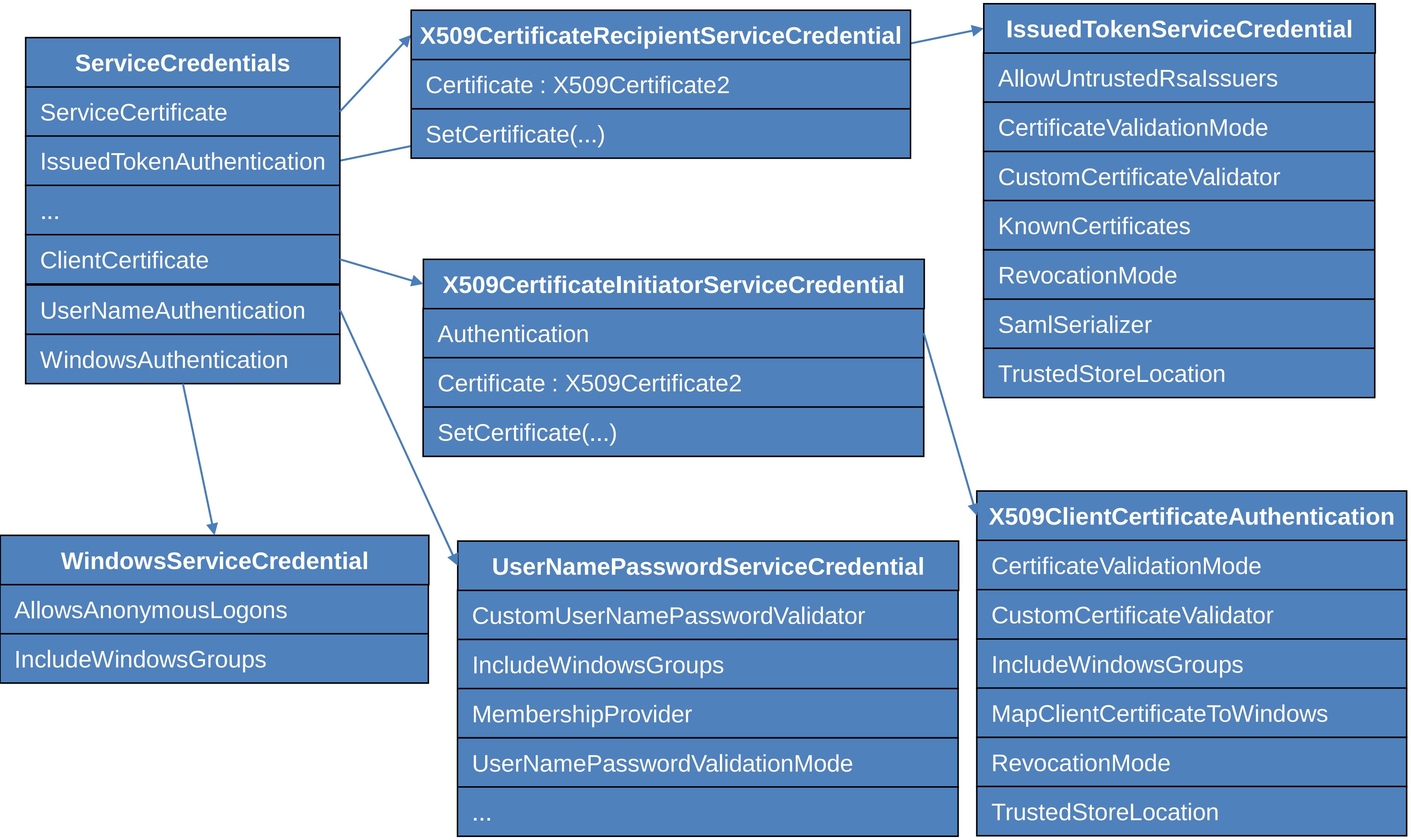
WS2007FederationHttpBinding



ClientCredentials


ClientCredentials
ClientCertificate
HttpDigest
IssuedToken
Peer
ServiceCertificate
SupportInteractive : bool
UserName
Windows
X509CertificateInitiatorClientCredential
Certificate : X509Certificate2
SetCertificate(...)
X509CertificateRecipientClientCredential
Authentication
DefaultCertificate : X509Certificate2
ScopedCertificates : D<Uri,X509Certificate2>
SetDefaultCertificate(...)
SetScopedCertificate(...)
IssuedTokenClientCredential
CacheIssuedTokens
...
IssuerChannelBehaviors
LocalIssuerAddress
LocalIssuerBinding
LocalIssuerChannelBehaviors
X509ServiceCertificateAuthentication
CertificateValidationMode
CustomCertificateValidator
RevocationMode
TrustedStoreLocation
WindowsClientCredential
AllowedImpersonationLevel : TokenImpersonationLevel
AllowNtlm : string
ClientCredentials : NetworkCredential
UserNamePasswordClientCredential
Password : string
UserName : string

ServiceCredentials


ServiceCredentials
ServiceCertificate
IssuedTokenAuthentication
...
ClientCertificate
UserNameAuthentication
WindowsAuthentication
X509CertificateRecipientServiceCredential
Certificate : X509Certificate2
SetCertificate(...)
IssuedTokenServiceCredential
AllowUntrustedRsaIssuers
CertificateValidationMode
CustomCertificateValidator
KnownCertificates
RevocationMode
SamlSerializer
TrustedStoreLocation
X509CertificateInitiatorServiceCredential
Authentication
Certificate : X509Certificate2
SetCertificate(...)
X509ClientCertificateAuthentication
CertificateValidationMode
CustomCertificateValidator
IncludeWindowsGroups
MapClientCertificateToWindows
RevocationMode
TrustedStoreLocation
WindowsServiceCredential
AllowsAnonymousLogons
IncludeWindowsGroups
UserNamePasswordServiceCredential
CustomUserNamePasswordValidator
IncludeWindowsGroups
MembershipProvider
UserNamePasswordValidationMode
...

Hands-on-Lab 5

- Create an IIS based WCF service
- Create a console client
- Use fedutil to federate with ACS
- Update service reference
- Reconfigure client settings