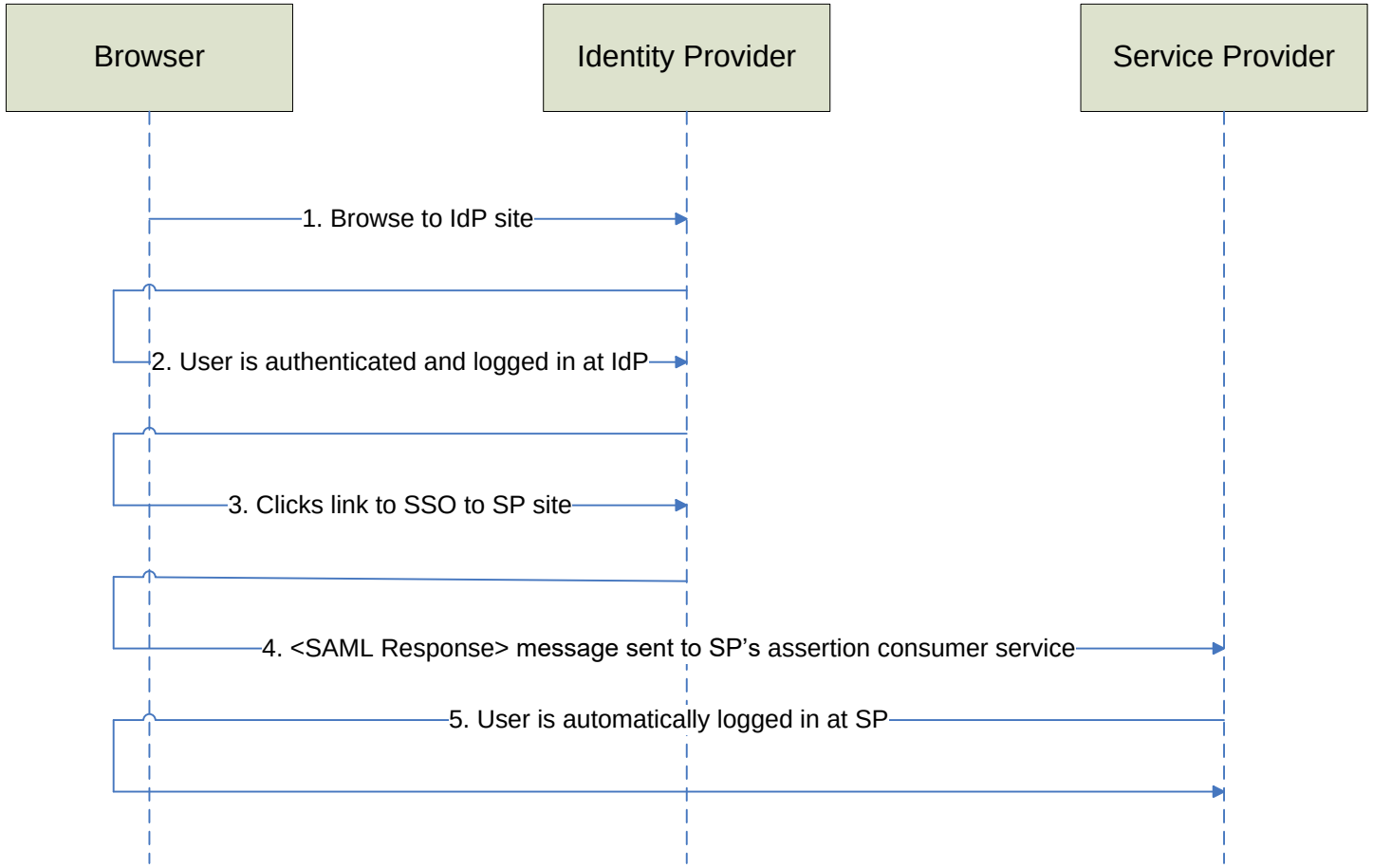
ComponentSpace

SAML SSO
for
ASP.NET Applications

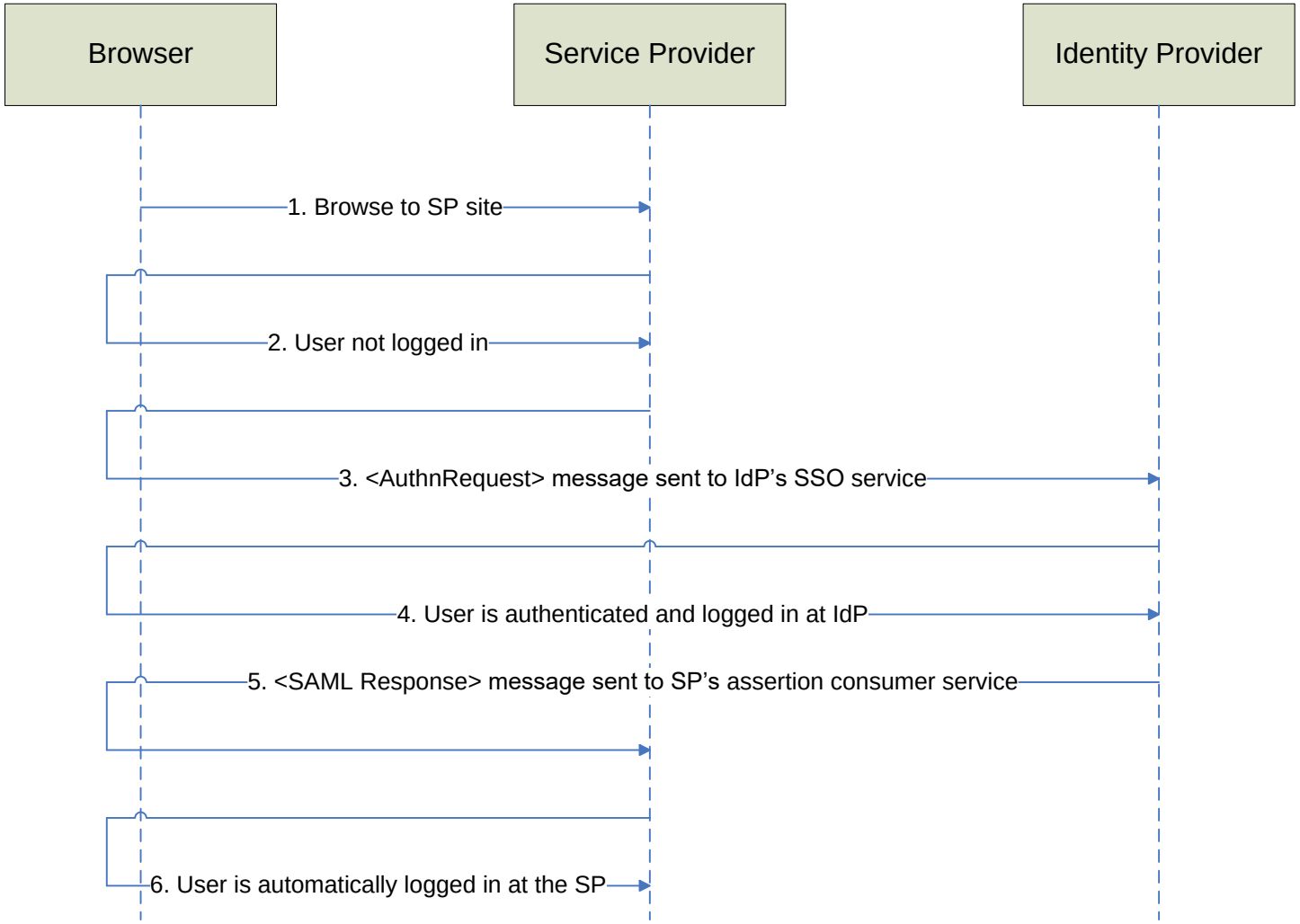
What is SAML SSO?

- Security Access Markup Language (SAML) v2.0
- Enables browser-based single sign-on between web applications
- Identity Provider (IdP) – asserts user's identity
- Service Provider (SP) – trusts user's asserted identity
- IdP-initiated SSO – user starts at IdP site
- SP-initiated SSO – user starts at SP site
- User is prompted to login at the IdP site if required

IdP-initiated SSO



SP-Initiated SSO



SAML Authn Request

```
<samlp:AuthnRequest
   ID="_ea882960-256b-46fd-8e7c-0422b273bc40" Version="2.0“
   IssueInstant="2013-06-30T00:11:36.212Z“
   Destination="http://localhost/ExampleIdentityProvider/SAML/SSOService.aspx"
   ForceAuthn="false" IsPassive="false"
   ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   AssertionConsumerServiceURL= "…/SAML/AssertionConsumerService.aspx"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
   <saml:Issuer
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      urn:componentspace:ExampleServiceProvider
   </saml:Issuer>
   <samlp:NameIDPolicy
       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
       AllowCreate="true" />
</samlp:AuthnRequest>
```

SAML Response

```
<samlp:Response
   ID="_41062871-1814-4827-bb43-8a1543181f5a"
   InResponseTo="_ea882960-256b-46fd-8e7c-0422b273bc40" Version="2.0"
   IssueInstant="2013-06-30T00:11:40.908Z"
   Destination="http://localhost/ExampleServiceProvider/SAML/AssertionConsumerService.aspx"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
   <saml:Issuer
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      urn:componentspace:ExampleIdentityProvider
   </saml:Issuer>
   <samlp:Status>
      <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
   </samlp:Status>
   <saml:Assertion>
      …
   </saml:Assertion>
</samlp:Response>
```

SAML Assertion

```
<saml:Assertion
   Version="2.0" ID="_b963a14a-c2b1-4c6f-9046-5f1c2ef370dd"
   IssueInstant="2013-06-30T00:11:40.923Z"     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
   <saml:Issuer>urn:componentspace:ExampleIdentityProvider</saml:Issuer>
   <saml:Subject>
      <saml:NameID>idp-user</saml:NameID>
      <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml:SubjectConfirmationData NotOnOrAfter="2013-06-30T00:14:40.923Z“
         Recipient="http://localhost/ExampleServiceProvider/SAML/AssertionConsumerService.aspx"
         InResponseTo="_ea882960-256b-46fd-8e7c-0422b273bc40" />
      </saml:SubjectConfirmation>
   </saml:Subject>
   <saml:Conditions
      NotBefore="2013-06-30T00:08:40.923Z" NotOnOrAfter="2013-06-30T00:14:40.923Z">
      <saml:AudienceRestriction>
         <saml:Audience>urn:componentspace:ExampleServiceProvider</saml:Audience>
      </saml:AudienceRestriction>
   </saml:Conditions>
   <saml:AuthnStatement>…</saml:AuthnStatement>
   <saml:AttributeStatement>…</saml:AttributeStatement>
</saml:Assertion>
```

SAML Assertion Content

- Contains assertions about the user's identity
- Typically includes the subject name identifier (NameID)
- NameID could be the user's email address
- Sometimes includes SAML attributes
- SAML attributes are often name/value strings
- Typical attributes include the user's first name and surname
- Attribute values can be strings or XML

SAML APIs

- ComponentSpace.SAML2 .NET DLL written in C#
- Supports all .NET languages
- .NET 2.0 and .NET 4.0 versions
- Targets ASP.NET web applications
- Original SAML low-level API
- More recent SAML high-level API
- Low level API provides ultimate control but you write more code
- High level API is simpler to use and covers the majority of scenarios

SAML High Level API

- Simplifies SAML SSO
- Reduces the amount of application code to write
- Configurable
- Handles loading of X.509 certificates
- Performs all security checks
- Verifies XML signatures

Initiating SSO at the IdP

```
SAMLIdentityProvider.InitiateSSO(
        Response,
        userName,
        attributes,
        targetUrl,
        partnerSP);
```

Initiating SSO at the SP

```
SAMLServiceProvider.InitiateSSO(
        Response,
        null,
        partnerIdP);
```

Sending a SAML Response at the IdP

```
SAMLIdentityProvider.SendSSO(
        Response,
        userName,
        attributes);
```

Receiving a SAML Response at the SP

```
SAMLServiceProvider.ReceiveSSO(
        Request,
        out isInResponseTo,
        out partnerIdP,
        out userName,
        out attributes,
        out targetUrl);
```

SAML Configuration

- Configuration for the local identity or service provider
- Configuration for the partner identity and service providers
- Specifies names, URLs, X.509 certificates, flags
- May be specified in a saml.config file within your application
- May be stored in a database and set programmatically

IdP SAML Configuration

```
<SAMLConfiguration xmlns="urn:componentspace:SAML:2.0:configuration">
  <IdentityProvider Name="http://localhost/ExampleIdentityProvider"
                    LocalCertificateFile="Certificates\idp.pfx"
                    LocalCertificatePassword="password"/>

  <PartnerServiceProviders>
    <!-- Web forms example -->
    <PartnerServiceProvider Name="http://localhost/ExampleServiceProvider"
                            WantAuthnRequestSigned="true"
                            SignSAMLResponse="true"
                            AssertionConsumerServiceUrl="http://localhost/ExampleServiceProvider/SAML/AssertionConsumerService.aspx"
                            PartnerCertificateFile="Certificates\sp.cer"/>
  </PartnerServiceProviders>
</SAMLConfiguration>
```

SP SAML Configuration

```
<SAMLConfiguration xmlns="urn:componentspace:SAML:2.0:configuration">
  <ServiceProvider Name="http://localhost/ExampleServiceProvider"
                   AssertionConsumerServiceUrl="~/SAML/AssertionConsumerService.aspx"
                   LocalCertificateFile="Certificates\sp.pfx"
                   LocalCertificatePassword="password"/>

  <PartnerIdentityProviders>
    <!-- Web forms example -->
    <PartnerIdentityProvider Name="http://localhost/ExampleIdentityProvider"
                             Description="Example Identity Provider"
                             SignAuthnRequest="true"
                             WantSAMLResponseSigned="true"
                             SingleSignOnServiceUrl="http://localhost/ExampleIdentityProvider/SAML/SSOService.aspx"
                             PartnerCertificateFile="Certificates\idp.cer"/>
  </PartnerIdentityProviders>
</SAMLConfiguration>
```

Security Considerations

- SAML SSO relies on the SP trusting the IdP and vice versa
- Trust is through the exchange of X.509 certificates
- Makes use of XML signatures and XML encryption
- HTTPS should be used for transport level security

XML Signatures

- XML Signature specifies a way to sign XML
- XML is signed using a private key and verified using a public key
- IdP signs the SAML response or assertion using its private key
- SP verifies the XML signature using the IdP's public key
- XML signature confirms who sent the XML and that the XML hasn't been modified
- Public keys distributed in X.509 certificates
- SHA-1 and SHA-2 signature algorithms are supported

XML Encryption

- XML Encryption specifies a way to encrypt XML
- XML is signed using a public key and verified using a private key
- IdP encrypts the SAML assertion using the SP's public key
- SP decrypts the encrypted assertion using its private key
- XML encryption provides end-to-end privacy of sensitive data
- Often not required as HTTPS is enough

X.509 Certificate Management

- Certificates and private keys may be stored in the –
  - file system (.PFX and .CER files)
  - Windows Certificate Store
- PFX files contain the certificate and password protected private key
- PFX should never be distributed to external parties
- CER files contain the certificate and public key
- Certificates are referenced within the SAML configuration
- Support for secondary certificates to manage expiring certificates

SAML Metadata

- XML format for exchanging configuration information
- Includes names, URLs, certificates, flags
- Its use is optional but encouraged
- ExportMetadata tool for generating metadata from saml.config
- ImportMetadata tool for importing metadata into saml.config

Deployment Considerations

- ComponentSpace.SAML2 DLL is published along with your application, typically in the bin folder
- Also saml.config files and X.509 certificates
- May be deployed to a single server, web farm or the cloud
- Web farm considerations include centralized SAML session storage
- Supports multi-tenancy applications – effectively a separate SAML configuration per tenant

Other Considerations

- Single logout (SLO) support
- ICertificateManager for custom X.509 certificate management
- IIDCache for custom ID caching at the SP
- ISSOSessionStore for custom SSO session storage
- ISAMLObserver for subscribing to SAML events

Debugging Tips

- SAML API throws exceptions on failure
- Capture and display the exception during development
- Support forums or support@componentspace.com
- http://www.componentspace.com/Forums/
- SAML trace provides detail information for use by ComponentSpace
- http://www.componentspace.com/Forums/17/Enabing-SAML-Trace
- When emailing ComponentSpace support, include a description of the issue, a screenshot of the browser, the saml.config with any passwords obfuscated, the SAML trace log as applicable