ComponentSpace SAML for ASP.NET Examples Guide

www.componentspace.com

Contents

Introduction

This document describes the example projects shipped with the product.

Refer to the SAML for ASP.NET Installation Guide for instructions on installing the product.

The example projects include SAML configurations. Refer to the SAML for ASP.NET Configuration Guide for information on SAML configuration.

The SAML for ASP.NET Developer Guide describes the SAML APIs called by the example projects.

Example SSO Projects

Single sign-on projects are found under the Examples\SSO folder.

The projects under the WebForms folder demonstrate SSO using ASP.NET web-forms applications.

These projects are described in this guide.

The projects under the MVC folder demonstrate SSO using ASP.NET MVC applications and Microsoft Identity/OWIN.

The SAML API calls are the same as those used in the web-forms applications although their placement reflects the differences in the application architectures.

SAML High-Level vs Low-Level API

The class library includes both high-level and low-level APIs. For the majority of use cases, it's recommended the high-level APIs are used as these provide the greatest ease of use and are configuration-driven. The low-level APIs are available for when maximum flexibility is required. The examples described in this guide use the high-level APIs.

Visual Studio Solution Files

Solution files for the various supported versions of Visual Studio are included in the installation folder.

Select the appropriate solution file to open the example projects in Visual Studio.

No changes are required for the example projects to build cleanly and run without error in Visual Studio.

Restoring Packages

As part of the build, Visual Studio should automatically restore the packages referenced by the example projects.

If not, from the Solution Explorer view, right click on the solution and restore the NuGet packages.

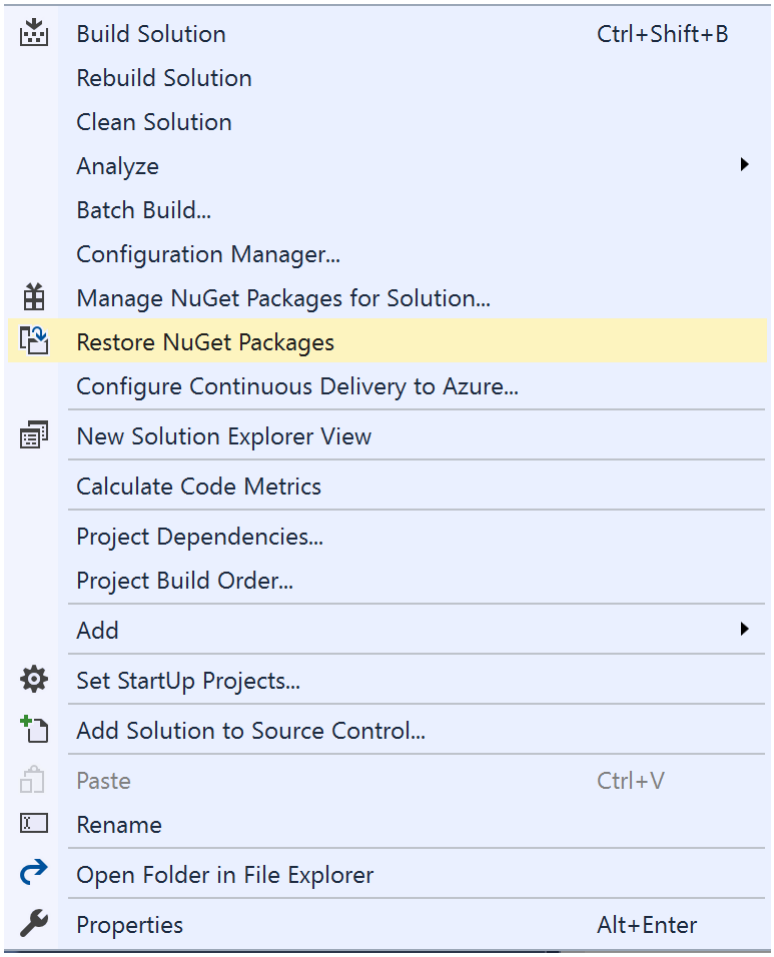


Figure 1 Restore NuGet Packages

Roslyn Compiler

If you get a runtime error indicating the Roslyn compiler cannot be found, run the following command from the Package Manager Console in Visual Studio and try again.

```
update-package Microsoft.CodeDom.Providers.DotNetCompilerPlatform -r
```

Setting the Startup Projects

Open the Visual Studio solution properties to edit the start-up project to ensure the required projects are run.

For example, start the ExampleIdentityProvider and ExampleServiceProvider projects for SSO between these applications.

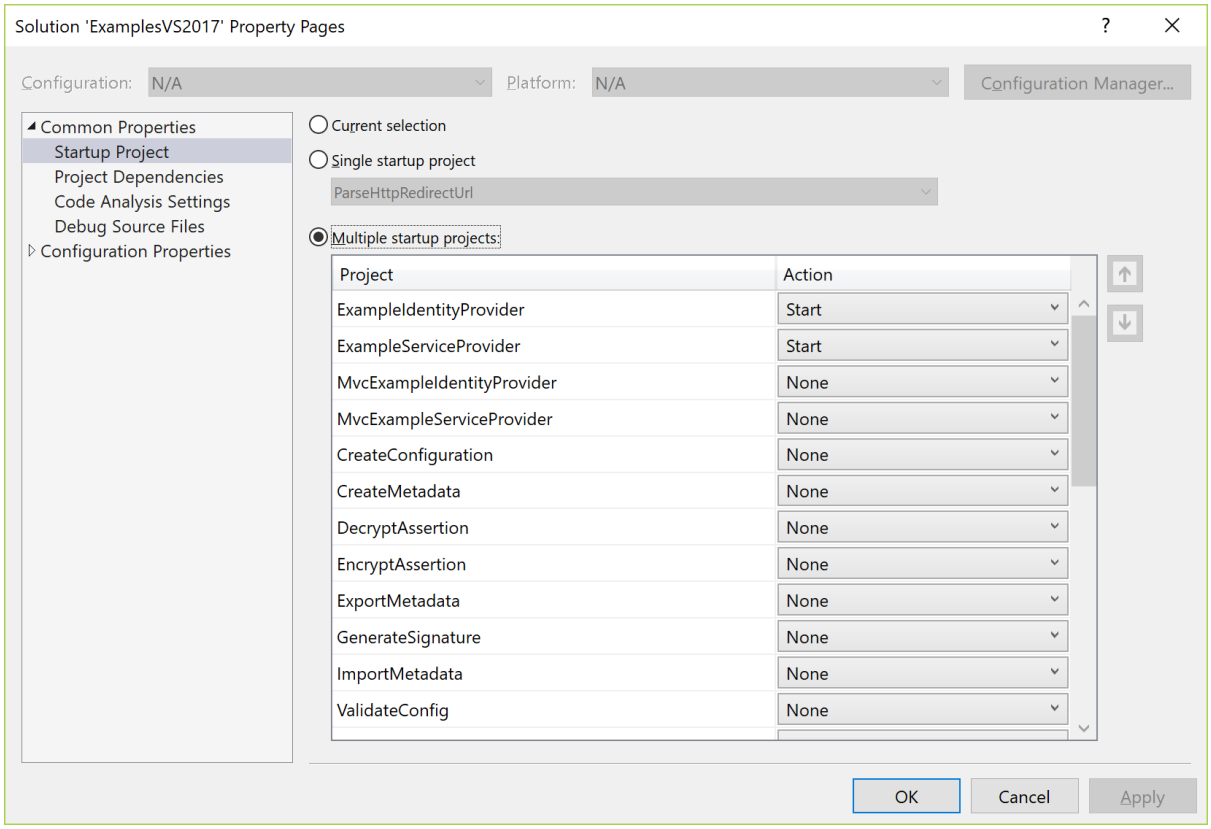


Figure 2 Start-up Projects

Example Identity Provider

The ExampleIdentityProvider project is an ASP.NET web-forms application based off the Visual Studio template.

It demonstrates acting as a SAML identity provider and supports:

- IdP-initiated SSO
- SP-initiated SSO
- IdP-initiated SLO
- SP-initiated SLO

Through changes to its SAML configuration, it can support all SAML v2.0 compliant third-party offerings.

Building and Running

The ExampleIdentityProvider should build without any errors or warnings.

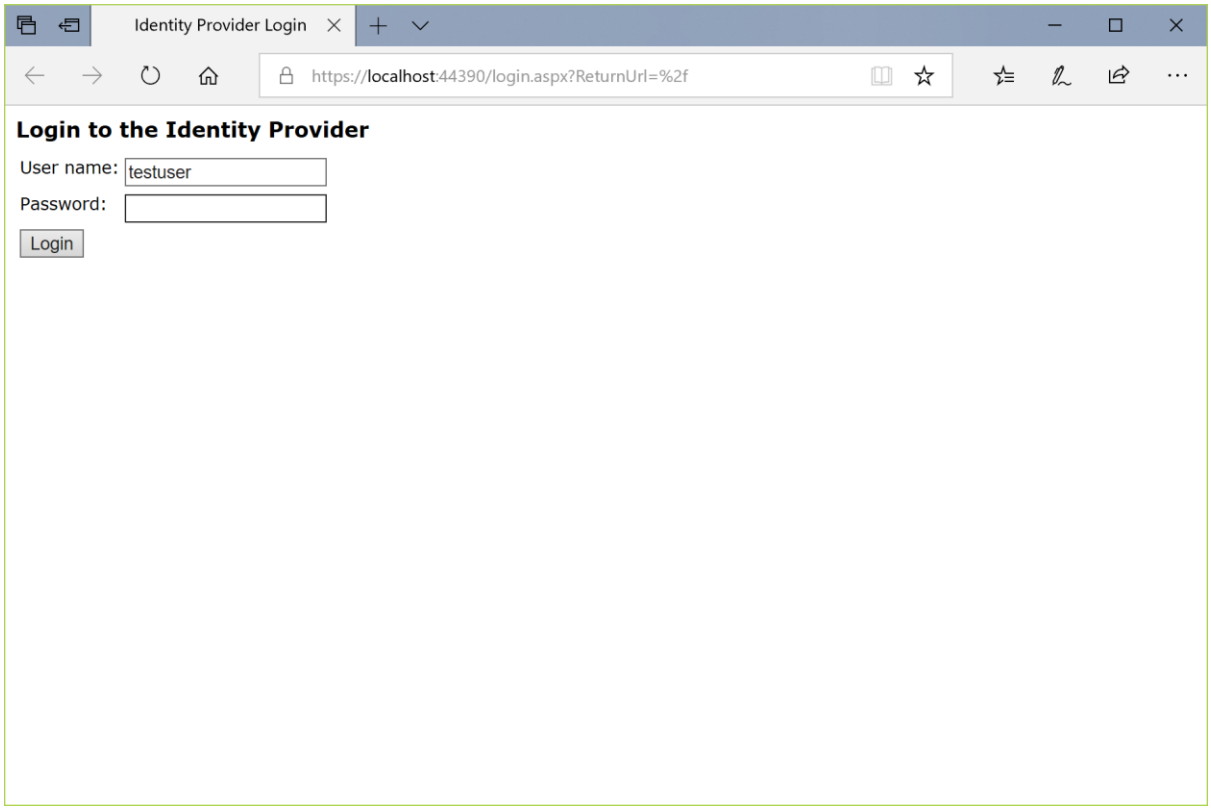
The application is configured to run at https://localhost:44390/.

If this is changed, the corresponding ExampleServiceProvider's SAML configuration must be updated to match the new URLs.

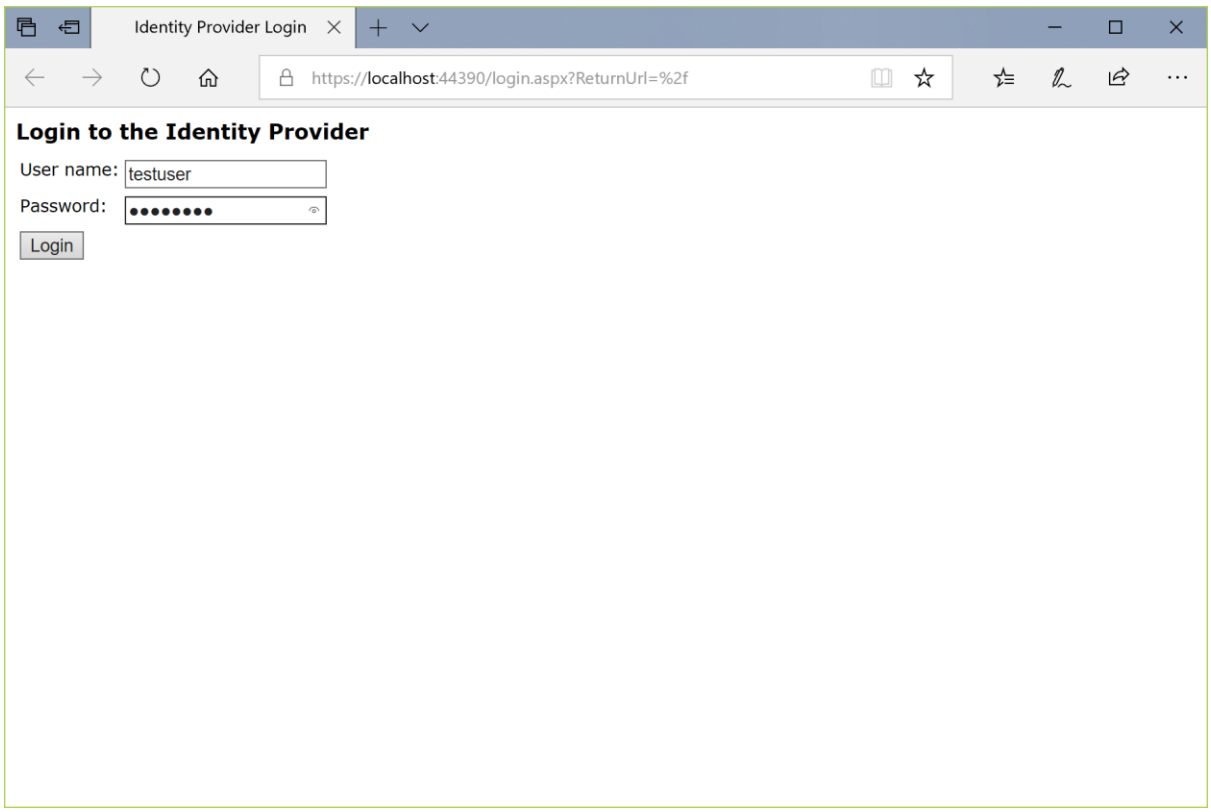
IdP-initiated SSO

Browse to the example identity provider's home page at https://localhost:44390/.

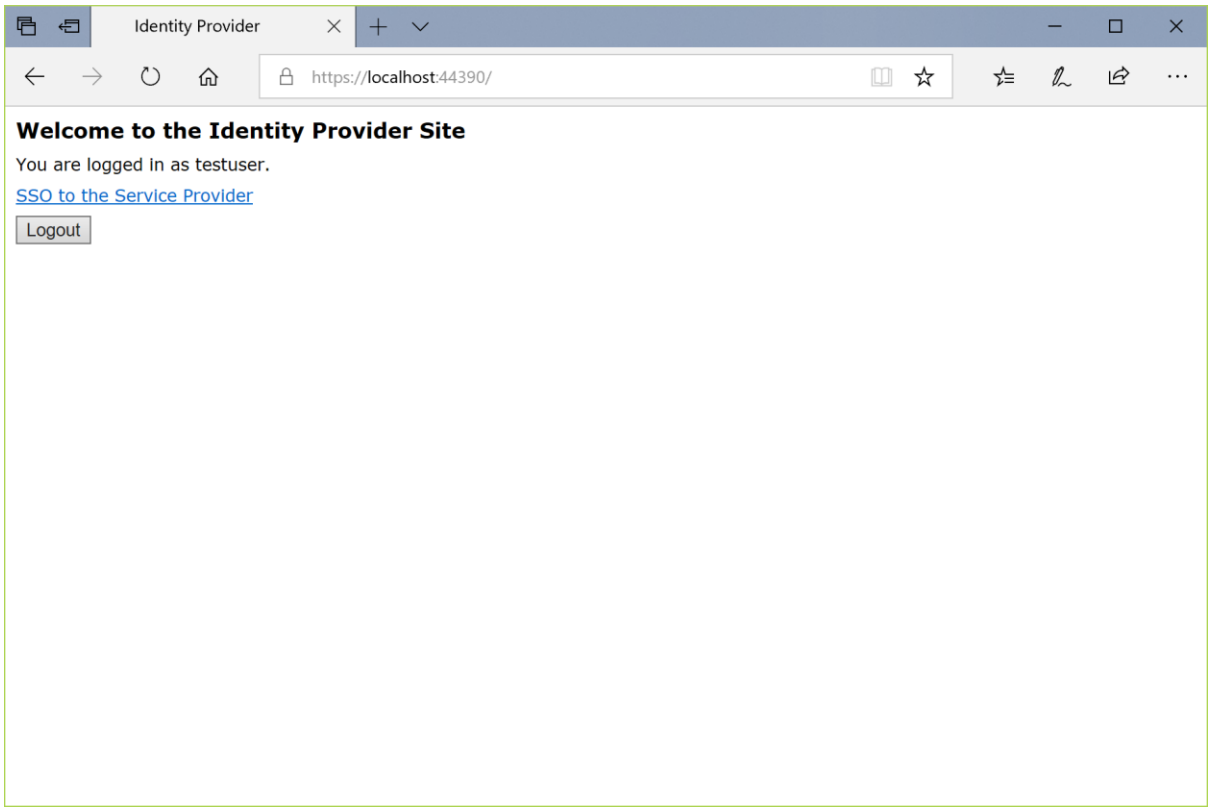
You're automatically redirected to the identity provider's login screen.



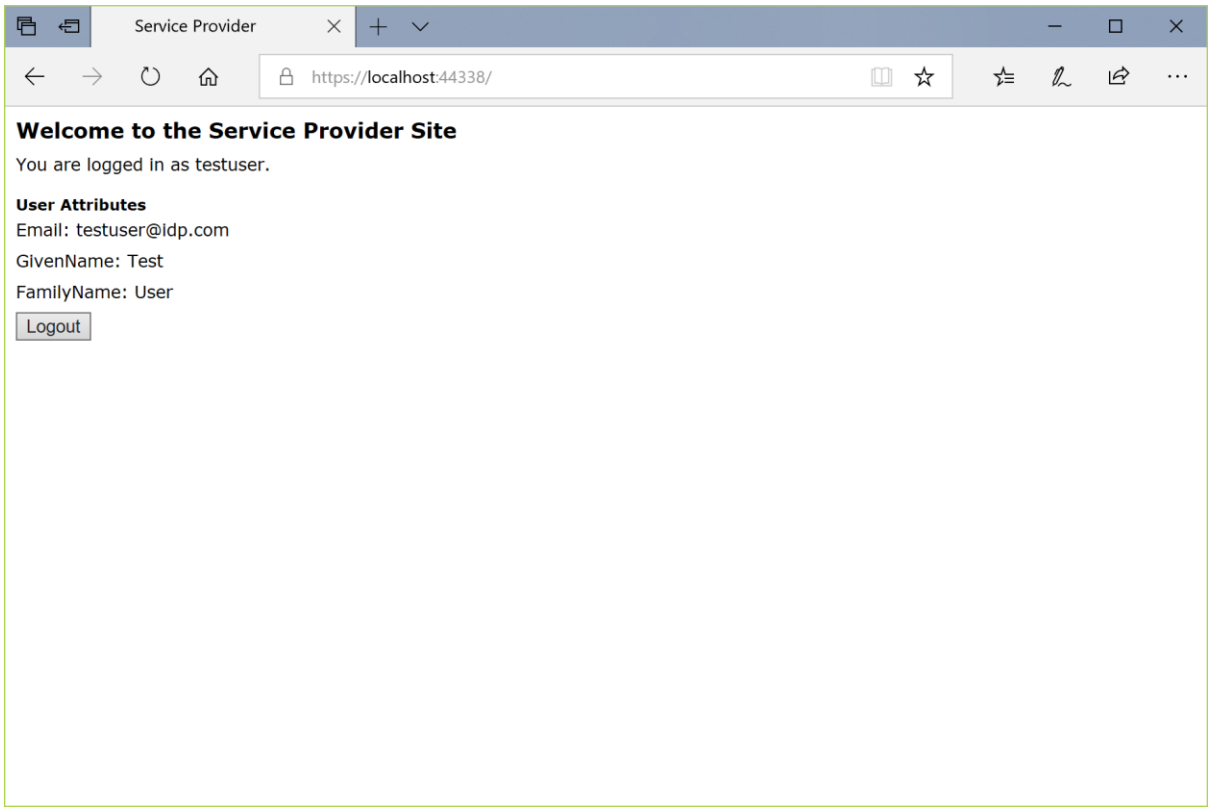
Login at the identity provider. The credentials are “testuser” and ”password”.



Click the link to SSO to the service provider.

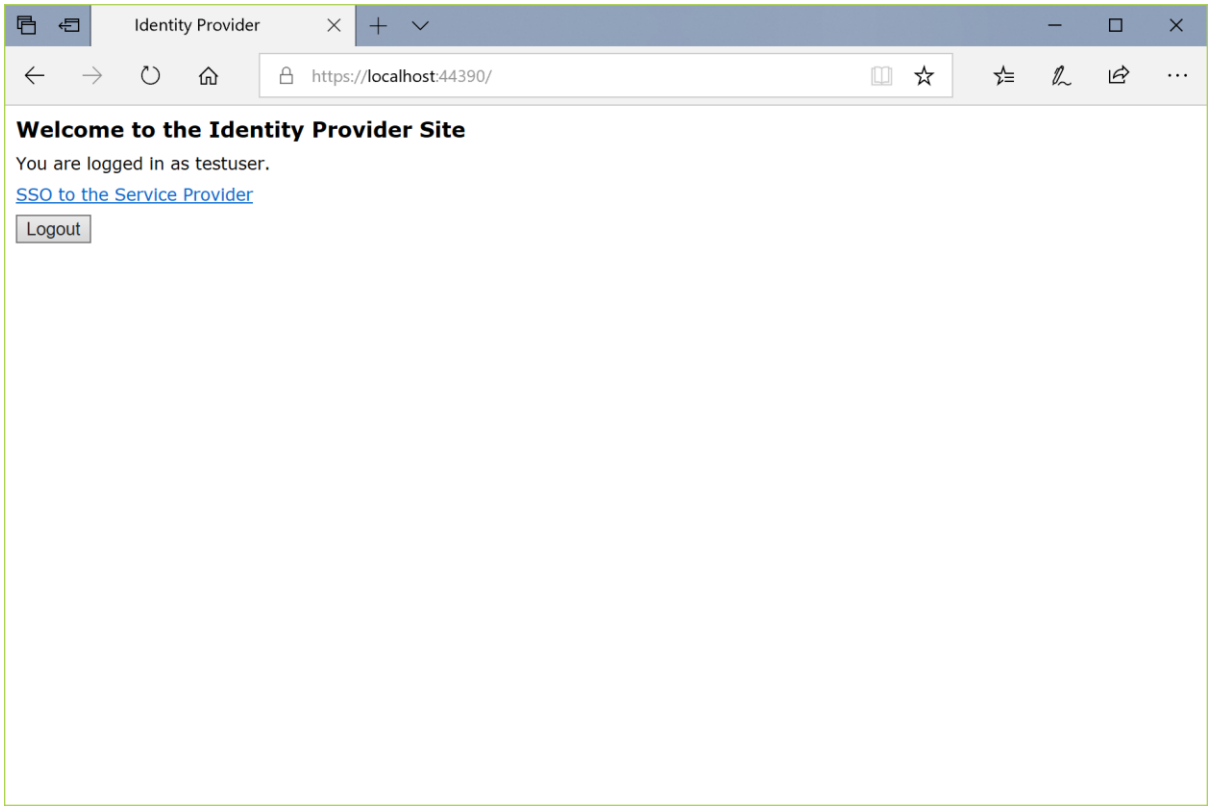


You are automatically logged in at the service provider.

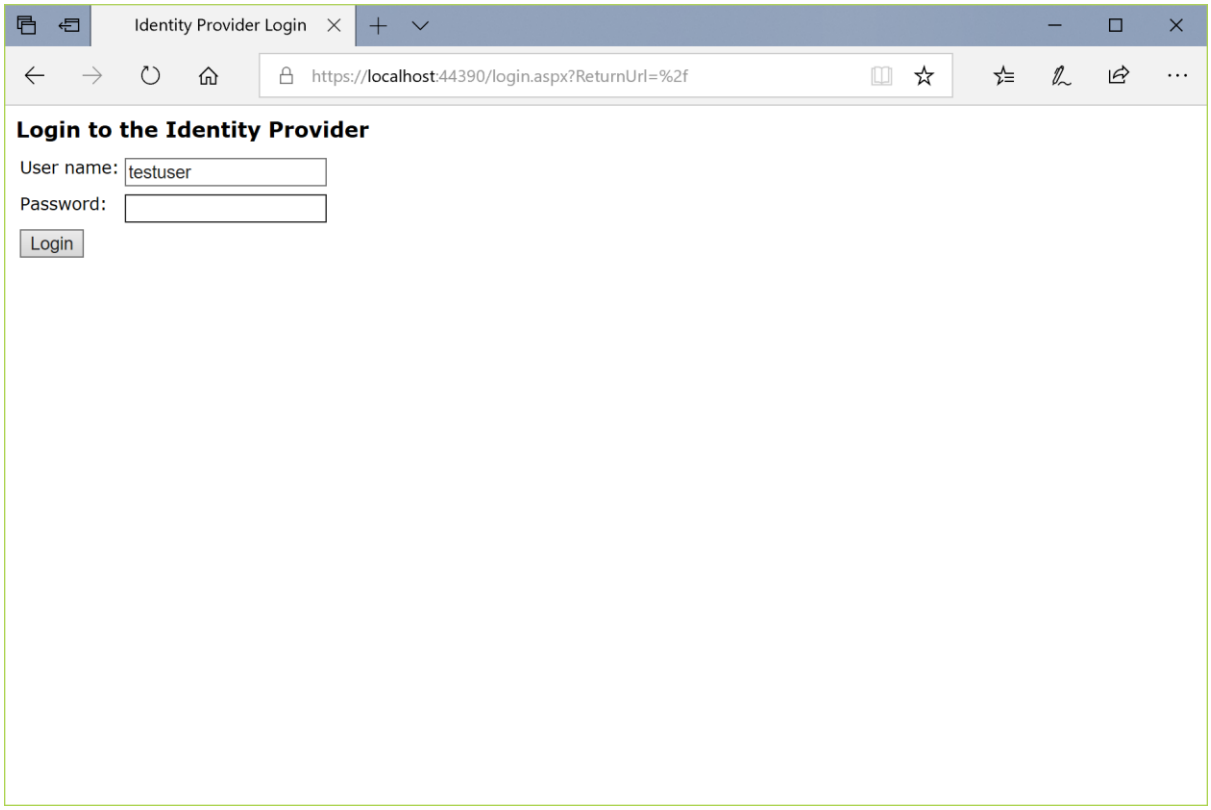


IdP-initiated SLO

Having completed SSO, in the same browser window, browse to the example identity provider's home page at https://localhost:44390/.



Click the Logout button. Logout occurs at both the identity provider and service provider.



Example Service Provider

The ExampleServiceProvider project is an ASP.NET web-forms application based off the Visual Studio template.

It demonstrates acting as a SAML service provider and supports:

- IdP-initiated SSO
- SP-initiated SSO
- IdP-initiated SLO
- SP-initiated SLO

Through changes to its SAML configuration, it can support all SAML v2.0 compliant third-party offerings.

Building and Running

The ExampleServiceProvider should build without any errors or warnings.

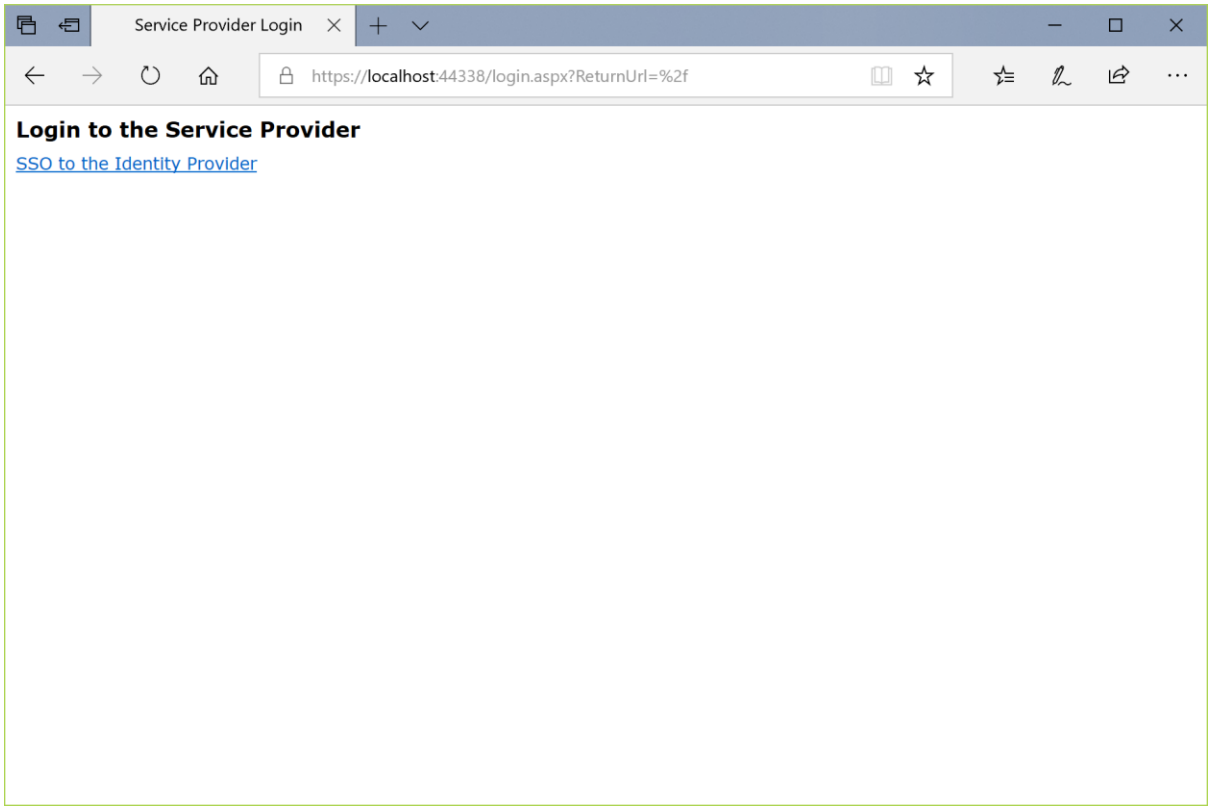
The application is configured to run at https://localhost:44338/.

If this is changed, the corresponding ExampleIdentityProvider's SAML configuration must be updated to match the new URLs.

SP-initiated SSO

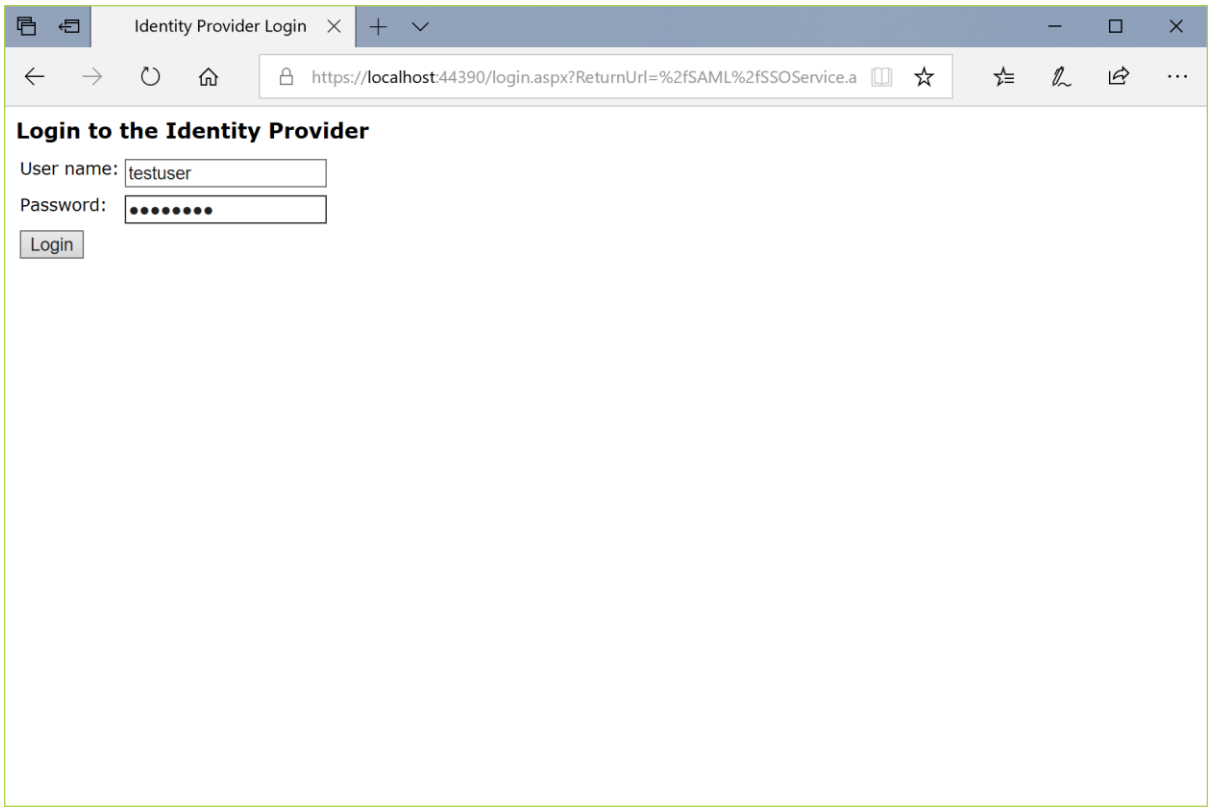
Browse to the example service provider's home page at https://localhost:44338/.

You're automatically redirected to the service provider's login screen.

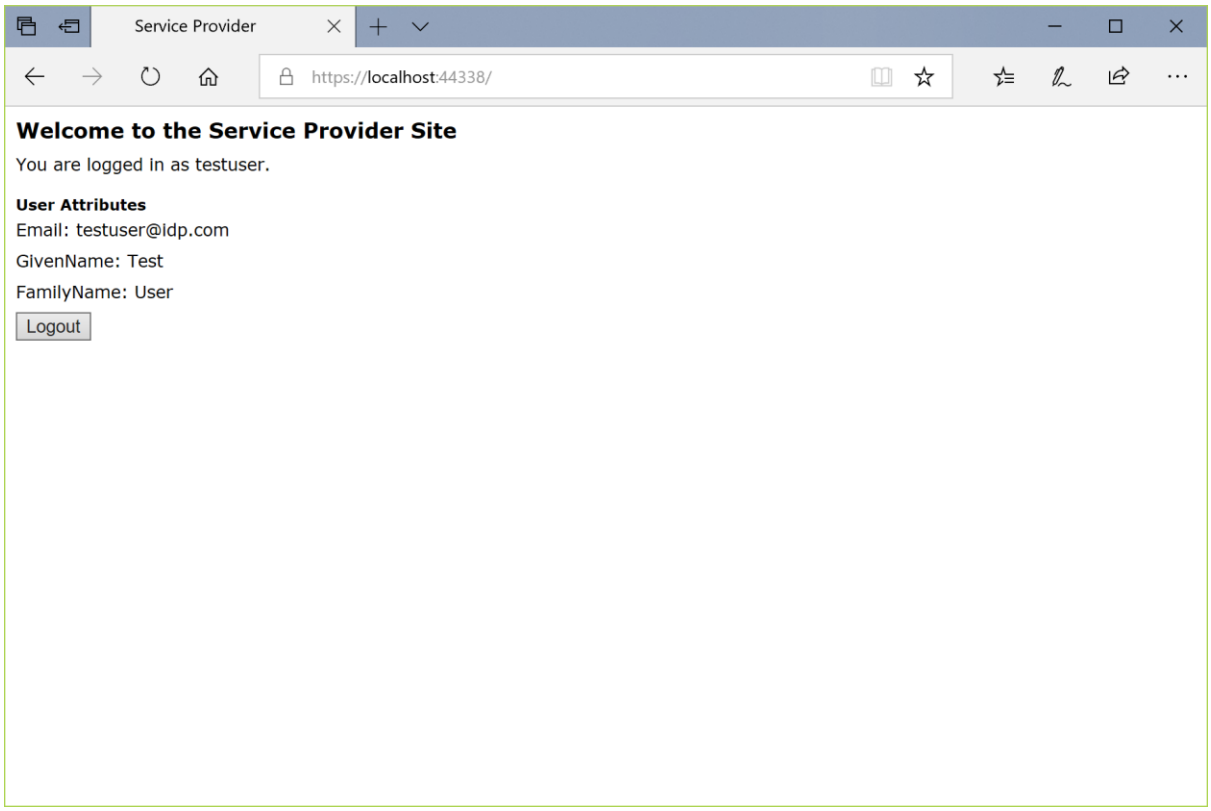


Click the link to SSO to the identity provider.

Login at the identity provider. The credentials are “testuser” and ”password”.

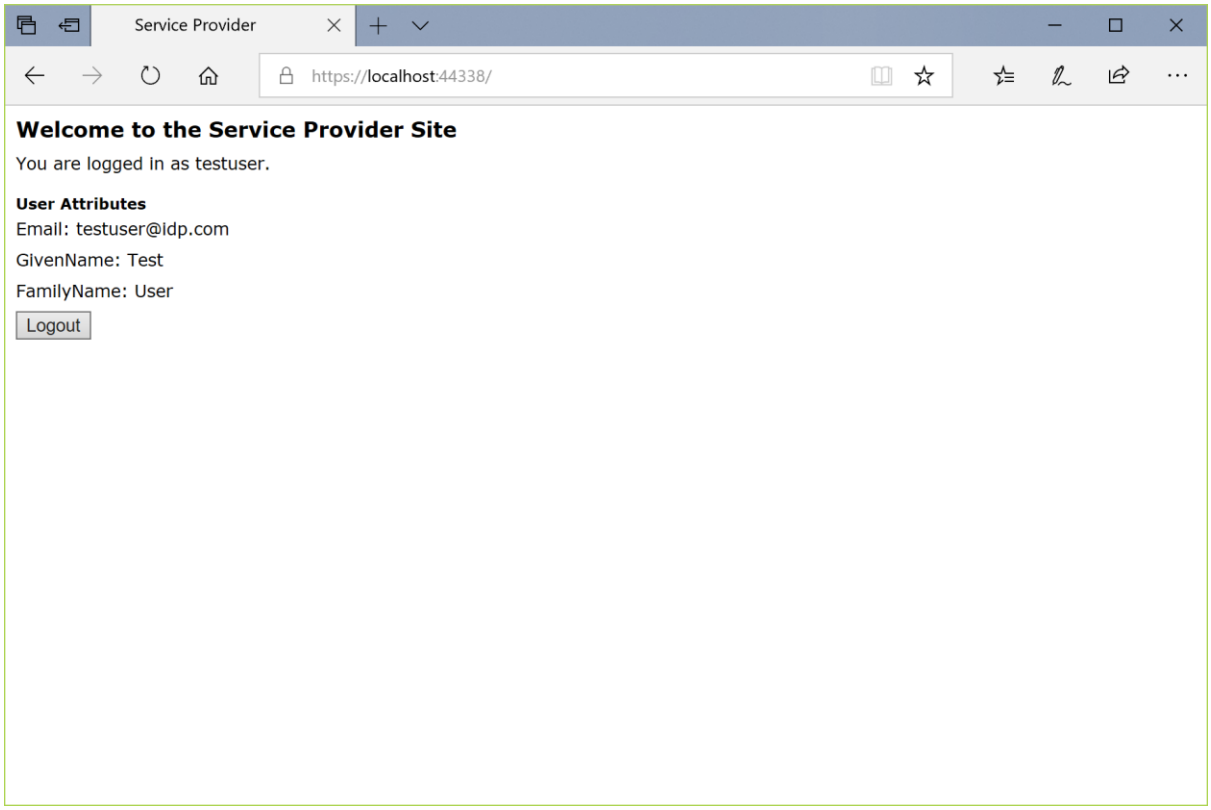


SSO completes with automatic login at the service provider. The user identity is that specified by the identity provider.

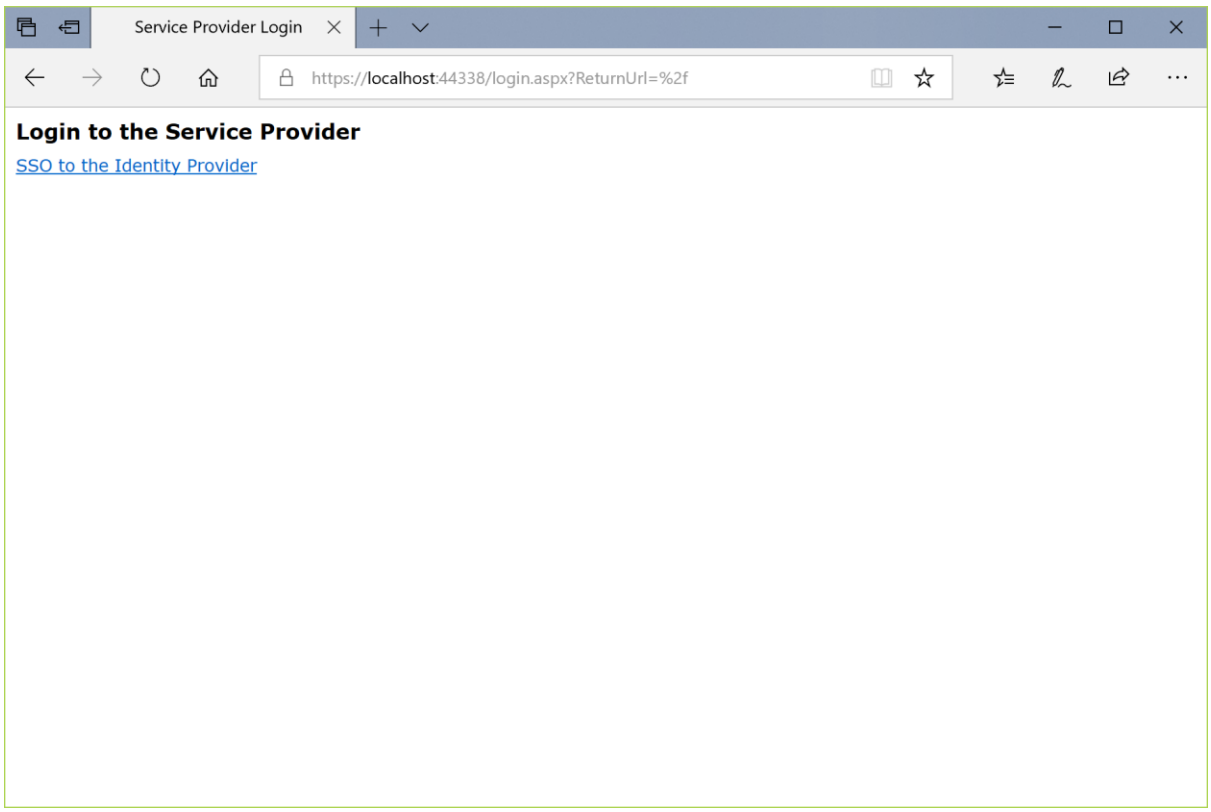


SP-initiated SLO

Having completed SSO, in the same browser window, browse to the example service provider's home page at https://localhost:44338/.



Click the Logout button. Logout occurs at both the identity provider and service provider.



MVC Example Identity Provider

The MvcExampleIdentityProvider project is an ASP.NET MVC application based off the Visual Studio template.

It demonstrates acting as a SAML identity provider and supports:

- IdP-initiated SSO
- SP-initiated SSO
- IdP-initiated SLO
- SP-initiated SLO

Through changes to its SAML configuration, it can support all SAML v2.0 compliant third-party offerings.

Building and Running

The MvcExampleIdentityProvider should build without any errors or warnings.

The application is configured to run at https://localhost:44363/.

If this is changed, the corresponding MvcExampleServiceProvider's SAML configuration must be updated to match the new URLs.

MVC Example Service Provider

The MvcExampleServiceProvider project is an ASP.NET MVC application based off the Visual Studio template.

It demonstrates acting as a SAML service provider and supports:

- IdP-initiated SSO
- SP-initiated SSO
- IdP-initiated SLO
- SP-initiated SLO

Through changes to its SAML configuration, it can support all SAML v2.0 compliant third-party offerings.

Building and Running

The MvcExampleServiceProvider should build without any errors or warnings.

The application is configured to run at https://localhost:44386/.

If this is changed, the corresponding MvcExampleIdentityProvider's SAML configuration must be updated to match the new URLs.

SAML Proxy

The SamlProxy project is an ASP.NET MVC web application based off the Visual Studio template.

It demonstrates acting as a SAML proxy for identity providers and service providers and supports:

- IdP-initiated SSO

- SP-initiated SSO
- IdP-initiated SLO
- SP-initiated SLO

An IdP-initiated SSO from an identity provider results in a new IdP-initiated SSO to the target service provider.

An SP-initiated SSO from a service provider results in a new SP-initiated SSO to the target identity provider.

The advantage of a SAML proxy is that it's a single access point for external identity providers or services providers single signing onto one or more internal identity providers or services providers. For example, an external identity provider needs to know about a single service provider only (i.e. the SAML Proxy) which acts on behalf of multiple internal service providers. This minimizes the configuration required at the identity provider and makes configuration changes easier to manage.

Through changes to its SAML configuration, it can support all SAML v2.0 compliant third-party offerings.

Building and Running

The SamlProxy should build without any errors or warnings.

The application is configured to run at https://localhost:44361/.

If this is changed, the corresponding SAML configurations must be updated to match the new URLs.

Code Walkthrough

Example Identity Provider

Configuration

The saml.config file includes the SAML configuration.

The configuration consists of a single SAML configuration specifying a single local identity provider and multiple partner service providers.

Refer to the SAML for ASP.NET Configuration Guide for more information.

InitiateSSO

The Default.aspx page includes the following code to support IdP-initiated SSO.

```
// Initiate single sign-on to the service provider (IdP-initiated SSO)
// by sending a SAML response containing a SAML assertion to the SP.
// Use the configured or logged in user name as the user name to send to the service provider (SP).
// Include some user attributes.
// If a target URL is specified the SP should display this page once SSO completes.
string partnerSP = WebConfigurationManager.AppSettings[AppSettings.PartnerSP];
string targetUrl = WebConfigurationManager.AppSettings[AppSettings.TargetUrl];

string userName = WebConfigurationManager.AppSettings[AppSettings.SubjectName];

if (string.IsNullOrEmpty(userName))
```

```
{
   userName = User.Identity.Name;
}

IDictionary<string, string> attributes = new Dictionary<string, string>();

foreach (string key in WebConfigurationManager.AppSettings.Keys)
{
   if (key.StartsWith(AppSettings.Attribute))
   {
      attributes[key.Substring(AppSettings.Attribute.Length + 1)] =
WebConfigurationManager.AppSettings[key];
   }
}

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

The user's name and some associated claims, to be included in the SAML assertion sent to the service provider, are retrieved from the application's configuration. The user ID and attributes, if any, sent to the service provider may be different depending on your business requirements.

The partner service provider name is retrieved from the application configuration. The method for determining which service provider to select may be different depending on your business requirements.

SAMLIdentityProvider.InitiateSSO is called to construct and send a SAML response to the service provider.

Control now moves to the service provider site.

InitiateSLO

The Default.aspx page includes the following code to support IdP-initiated SLO.

```
// Logout locally.
FormsAuthentication.SignOut();

if (SAMLIdentityProvider.CanSLO())
{
   // Request logout at the service providers.
   SAMLIdentityProvider.InitiateSLO(Response, null, null);
}
else
{
   FormsAuthentication.RedirectToLoginPage();
}
```

If the user clicks the logout button, a check is made to see whether the user has completed SSO and, if so, SAMLIdentityProvider.InitiateSLO is called to construct and send a logout request to the service provider(s).

Control now moves to the service provider site.

ReceiveSSO/SendSSO

The SAML/SSOService.aspx page includes the following code to support SP-initiated SSO.

```
// Either an authn request has been received or login has just completed in response to a previous
authn request.
// The SSO pending session flag is false if an authn request is expected. Otherwise, it is true if
// a login has just completed and control is being returned to this page.
bool ssoPending = Session[ssoPendingSessionKey] != null && (bool)Session[ssoPendingSessionKey]
== true;

if (!(ssoPending && User.Identity.IsAuthenticated))
{
  string partnerSP = null;

  // Receive the authn request from the service provider (SP-initiated SSO).
  SAMLIdentityProvider.ReceiveSSO(Request, out partnerSP);

  // If the user isn't logged in at the identity provider, force the user to login.
  if (!User.Identity.IsAuthenticated)
  {
    Session[ssoPendingSessionKey] = true;
    FormsAuthentication.RedirectToLoginPage();
    return;
  }
}

Session[ssoPendingSessionKey] = null;

// The user is logged in at the identity provider.
// Respond to the authn request by sending a SAML response containing a SAML assertion to the SP.
// Use the configured or logged in user name as the user name to send to the service provider (SP).
// Include some user attributes.
string userName = WebConfigurationManager.AppSettings[AppSettings.SubjectName];

if (string.IsNullOrEmpty(userName))
{
  userName = User.Identity.Name;
}

IDictionary<string, string> attributes = new Dictionary<string, string>();

foreach (string key in WebConfigurationManager.AppSettings.Keys)
{
  if (key.StartsWith(AppSettings.Attribute))
  {
    attributes[key.Substring(AppSettings.Attribute.Length + 1)] =
WebConfigurationManager.AppSettings[key];
  }
}
```

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

SAMLIdentityProvider.ReceiveSSO receives and processes the SAML authentication request from the service provider.

If the user isn't authenticated locally, they're redirected to the login page.

Once authenticated, attributes to be included in the SAML assertion sent to the service provider are retrieved from the application's configuration.

SAMLIdentityProvider.SendSSO is called to construct and send a SAML response to the service provider.

Control now returns to the service provider site.

ReceiveSLO/SendSLO

The SAML/SLOService.aspx page includes the following code to support IdP-initiated and SP-initiated SLO.

```
// Receive the single logout request or response.
// If a request is received then single logout is being initiated by the service provider.
// If a response is received then this is in response to single logout having been initiated by the identity provider.
bool isRequest = false;
bool hasCompleted = false;
string logoutReason = null;
string partnerSP = null;
string relayState = null;

SAMLIdentityProvider.ReceiveSLO(Request, Response, out isRequest, out hasCompleted, out logoutReason, out partnerSP, out relayState);

if (isRequest)
{
   // Logout locally.
   FormsAuthentication.SignOut();

   // Respond to the SP-initiated SLO request indicating successful logout.
   SAMLIdentityProvider.SendSLO(Response, null);
}
else
{
   if (hasCompleted)
   {
      // IdP-initiated SLO has completed.
      Response.Redirect("~/");
   }
}
```

SAMLIdentityProvider.ReceiveSLO is called to receive and process the logout message from the service provider.

For IdP-initiated SLO, a logout response is received. If the results indicate SLO has completed, the user is redirected to the home page.

For SP-initiated SLO, a logout request is received. The user is logged out locally and SAMLIdentityProvider.SendSLO is called to construct and send a SAML logout response to the service provider. Control now returns to the service provider site.

Example Service Provider

Configuration

The saml.config file includes the SAML configuration.

The configuration consists of a single SAML configuration specifying a single local service provider and multiple partner identity providers.

Refer to the SAML for ASP.NET Configuration Guide for more information.

InitiateSSO

The Login.aspx page includes the following action to support SP-initiated SSO.

```
// Remember the return URL.
string returnUrl = Request.QueryString["ReturnUrl"];

// To login at the service provider, initiate single sign-on to the identity provider (SP-initiated SSO).
string partnerIdP = WebConfigurationManager.AppSettings[AppSettings.PartnerIdP];

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

The partner identity provider name is retrieved from the application configuration. The method for determining which identity provider to select may be different depending on your business requirements.

SAMLServiceProvider.InitiateSSO is called to construct and send a SAML authentication request to the identity provider.

Control now moves to the identity provider site.

InitiateSLO

The Default.aspx page includes the following code to support SP-initiated SLO.

```
// Logout locally.
FormsAuthentication.SignOut();

if (SAMLServiceProvider.CanSLO(WebConfigurationManager.AppSettings[AppSettings.PartnerIdP]))
{
   // Request logout at the identity provider.
   string partnerIdP = WebConfigurationManager.AppSettings[AppSettings.PartnerIdP];
   SAMLServiceProvider.InitiateSLO(Response, null, null, partnerIdP);
}
else
{
   FormsAuthentication.RedirectToLoginPage();
}
```

If the user clicks the logout button, a check is made to see whether the user has completed SSO and, if so, SAMLServiceProvider.InitiateSLO is called to construct and send a logout request to the identity provider.

Control now moves to the identity provider site.

ReceiveSSO

The SAML/AssertionConsumerService.aspx page includes the following code to support IdP-initiated and SP-initiated SSO.

```
bool isInResponseTo = false;
string partnerIdP = null;
string authnContext = null;
string userName = null;
IDictionary<string, string> attributes = null;
string targetUrl = null;

// Receive and process the SAML assertion contained in the SAML response.
// The SAML response is received either as part of IdP-initiated or SP-initiated SSO.
SAMLServiceProvider.ReceiveSSO(Request, out isInResponseTo, out partnerIdP, out authnContext,
out userName, out attributes, out targetUrl);

// If a target URL is supplied, ensure it's local to avoid potential open redirection attacks.
if (targetUrl != null && !IsLocalUrl(targetUrl))
{
  targetUrl = null;
}

// If no target URL is provided, provide a default.
if (targetUrl == null)
{
  targetUrl = "~/";
}

// Login automatically using the asserted identity.
// This example uses forms authentication. Your application can use any authentication method you
choose.
// There are no restrictions on the method of authentication.
FormsAuthentication.SetAuthCookie(userName, false);

// Save the attributes.
Session[AttributesSessionKey] = attributes;

// Redirect to the target URL.
Response.Redirect(targetUrl, false);
```

SAMLServiceProvider.ReceiveSSO receives and processes the SAML response from the identity provider. The SAML response is either the result of IdP-initiated or SO-initiated SSO.

The user is logged in automatically at the service provider using information retrieved from the SAML assertion. The user ID and attributes, if any, received by the service provider may be different depending on your business requirements.

For IdP-initiated SSO, the optional relay state sent by the identity provider specifies a URL to support deep web linking. If specified, the user is redirected to this page. Otherwise the user is redirected to the home page.

ReceiveSLO/SendSLO

The SAML/SLOService.aspx page includes the following code to support IdP-initiated and SP-initiated SLO.

```
// Receive the single logout request or response.
// If a request is received then single logout is being initiated by the identity provider.
// If a response is received then this is in response to single logout having been initiated by the service provider.
bool isRequest = false;
string logoutReason = null;
string partnerIdP = null;
string relayState = null;

SAMLServiceProvider.ReceiveSLO(Request, out isRequest, out logoutReason, out partnerIdP, out relayState);

if (isRequest)
{
   // Logout locally.
   FormsAuthentication.SignOut();

   // Respond to the IdP-initiated SLO request indicating successful logout.
   SAMLServiceProvider.SendSLO(Response, null);
}
else
{
   // SP-initiated SLO has completed.
   Response.Redirect("~/");
}
```

SAMLServiceProvider.ReceiveSLO is called to receive and process the logout message from the identity provider.

For SP-initiated SLO, a logout response is received. The user is redirected to the home page.

For IdP-initiated SLO, a logout request is received. The user is logged out locally and SAMLServiceProvider.SendSLO is called to construct and send a SAML logout response to the identity provider. Control now returns to the identity provider site.

Error Handling

As these are example applications, no error handling is included. Exceptions are not caught and therefore are displayed in the browser.

In a production application, exceptions should be caught and processed.

Refer to the SAML for ASP.NET Developer Guide for more information.