Lab 3

Windows Azure Access Control Service

1. This lab extends the guidance in Chapter 2, "*Claims-Based Single Sign-On for the Web and Windows Azure*" of the book "*A Guide to Claims–based Identity and Access Control, 2nd Edition*" (<http://msdn.microsoft.com/en-us/library/ff423674.aspx>). It demonstrates the use of Windows Azure AppFabric Access Control Service (ACS) as an authentication provider and token issuer. It shows how to configure ACS though the portal website, how to use the ACS management API, and how you can configure ADFS to trust ACS.
   1. To perform the exercises in this lab you must have previous created a Windows Azure account and uploaded a suitable X.509 certificate for use when signing certificates delivered by ACS. For information about creating a Windows Azure account, go to the Windows Azure Home page at <http://www.microsoft.com/windowsazure/>.

# Objectives

After working though this lab, you will understand how you can use ACS as a token issuer (STS) that makes use of a range of identity providers for authentication. You will learn how to configure ACS for different types of identity providers, and use the identities for authentication in your applications. You will also learn how to use the Management API exposed by ACS to programmatically add and configure relying party applications. The final optional exercise in this lab shows you how to use ACS as a trusted issuer with Microsoft Active Directory Federation Services (ADFS).

* 1. **Note:** Before you start working with these exercises ensure you have run the dependency checking utility. You must also run Visual Studio as an administrator when opening the solutions in these exercises. If you simply double-click the solution file, Visual Studio may fail to load the projects. See the "*Introduction*" document for information about the dependency checker utility and how to run Visual Studio as an administrator.

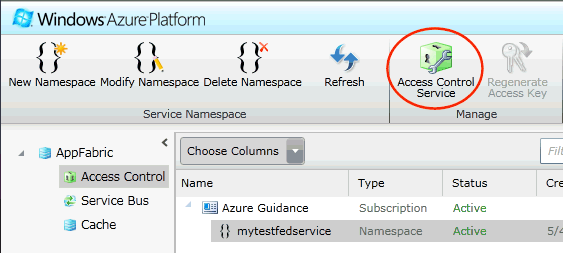
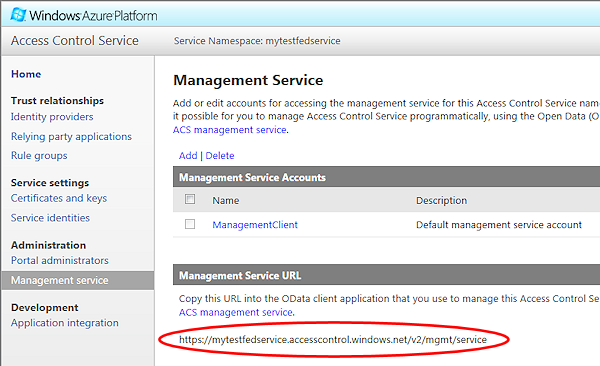
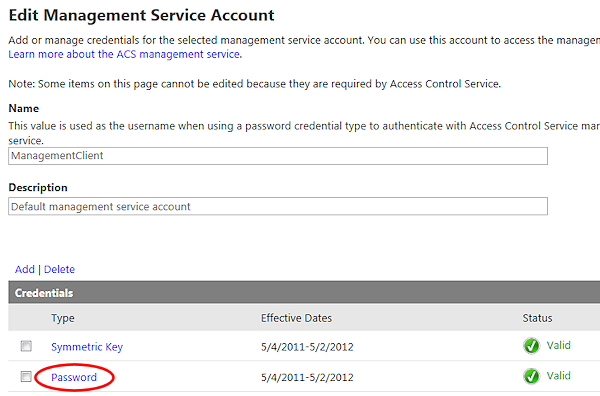
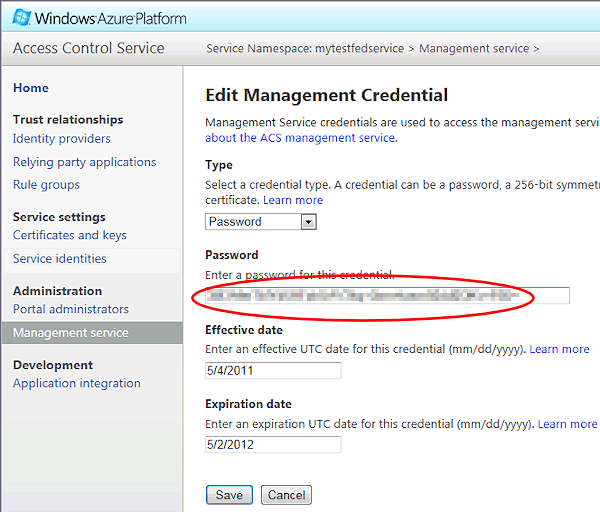
This lab contains the following exercises:

* + [Exercise 1](#Ex01): **Adding ACS as a Trusted Issuer**. In this exercise you will start with a version of the a-Order application similar to that you used in previous labs, and modify it to use Windows Azure AppFabric Access Control Service (ACS) as the trusted issuer and identity provider in addition to the Adatum federation provider and simulated issuer.
  + [Exercise 2](#Ex02): **Adding the Facebook Identity Provider and Home Realm Discovery**. In this exercise you will add Facebook as an identity provider to your ACS namespace. This illustrates how, by taking advantage of ACS, you can easily change the options a user has for authentication when using your applications; without requiring any modification of the application or of your own local token issuer or federation provider.
  + [Exercise 3](#Ex03): **Adding a Custom OpenID Identity Provider**. In this exercise you will use the ACS Management API to programmatically add a relying party application that uses the OpenID identity provider.
  + [Exercise 4](#Ex04): **Replacing the Adatum Federation Provider with ADFS**. In this optional additional exercise you will replace the existing Adatum federation provider with an ADFS instance, and configure this to use ACS as a token issuer and identity provider.

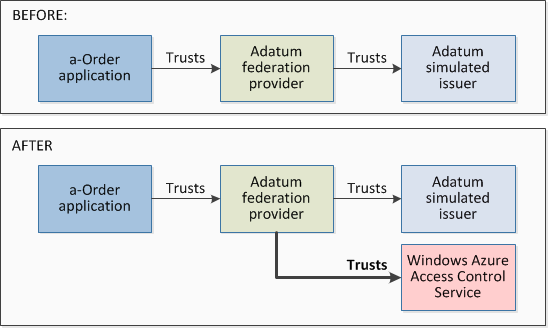
# Locating Your ACS Password Key and Namespace

* 1. The "End" solutions for the exercises in this lab, and the tasks in Exercise 3 of this lab, require you to enter the ACS password key and namespace into a setup program that uses the ACS Management API to configure your namespace.

To locate your ACS password key and namespace values

* 1. Sign into the Windows Azure portal at <https://windows.azure.com/default.aspx>.
  2. Select **Service Bus, AccessControl & Caching** in the left pane, and click **Access Control** in the **AppFabric** tree.
  3. Select your namespace in the central pane and click the **Access Control Service** icon at the top of the window.
     1. 
  4. In the Access Control Service page, select **Management service** in the left pane. Make a note of the **Management Service URL** value shown at the bottom of the page. You will need to enter just the namespace part of this URL when you configure the setup programs in this lab.
     1. 
     2. In the example above, where the management URL is **https://mytestfedservice.accesscontrol.windows.net/v2/mgmt/service**, the namespace that you would enter into the setup programs is just **mytestfedservice**.
  5. In the **Management Service Accounts** list click **ManagementClient**. This opens a page that shows information about the default management service account.
     1. 
     2. You can create a different account for accessing the management service if you wish and specify the credentials for that account instead of the default service management account.
  6. In the Edit Management Service Account page click the **Password** link. This opens the Edit Management Credential page.
     1. 
  7. Your management account password is displayed in the **Password** section of this page. Copy and paste it into a text document ready for when you configure the setup programs in this lab.

# Exercise 1: Adding ACS as a Trusted Issuer

* 1. In this exercise you will start with a version of the a-Order application similar to that you used in previous labs, and modify it to use Windows Azure AppFabric Access Control Service (ACS) as the trusted issuer and identity provider in addition to the Adatum federation provider and simulated issuer. The a-Order application authenticates through the Adatum federation provider. At the moment, the Adatum federation provider authenticates only through the Adatum simulated issuer, which not acts solely as an identity provider.
  2. In this first exercise you will configure ACS for use with the Adatum federation provider, and then modify the Adatum federation provider so that it trusts ACS. This will allow users to authenticate either through the Adatum simulated issuer, or by selecting a different identity provider that is supported by and exposed through ACS.
  3. 

This exercise contains the following tasks:

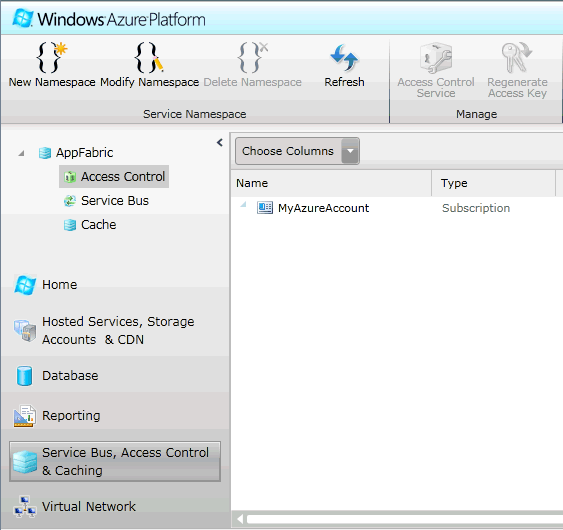
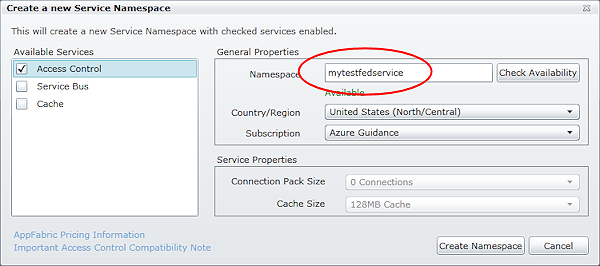
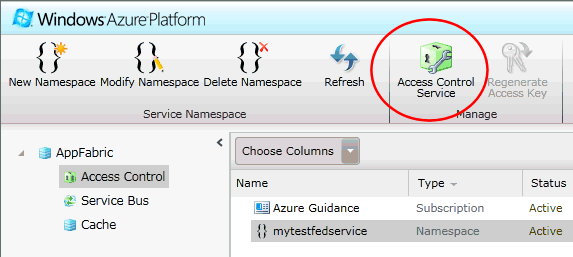
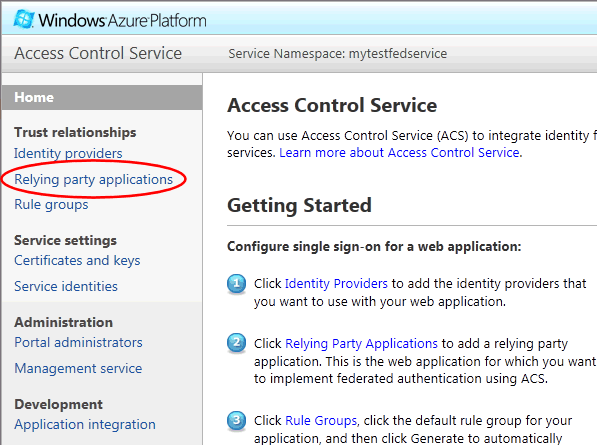
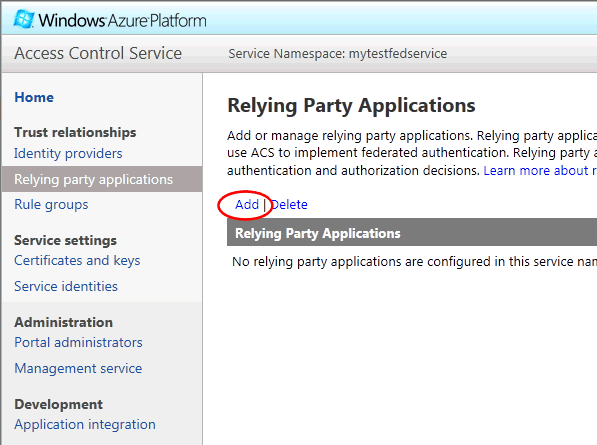
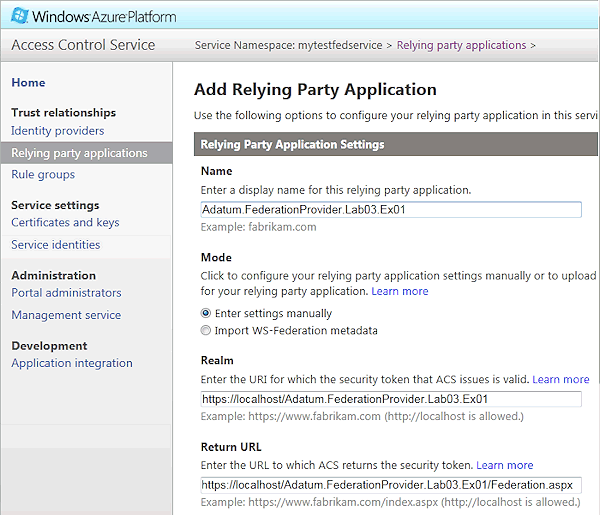
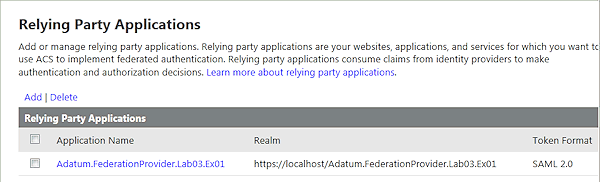
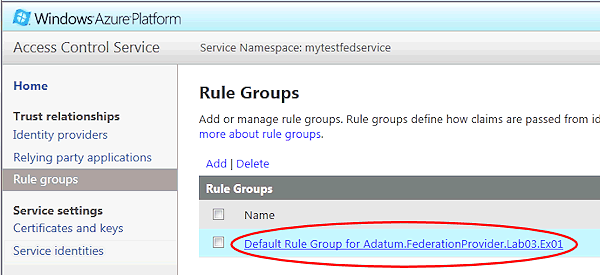
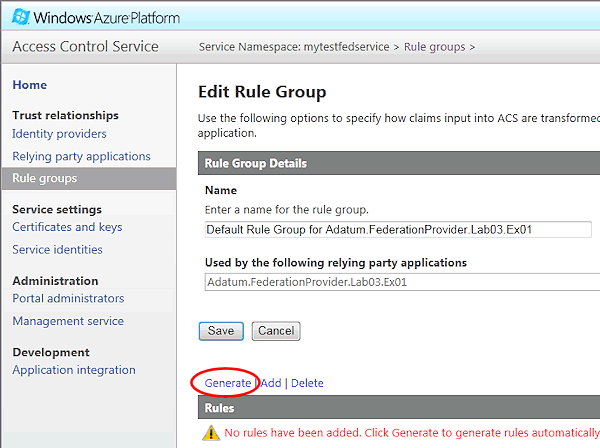
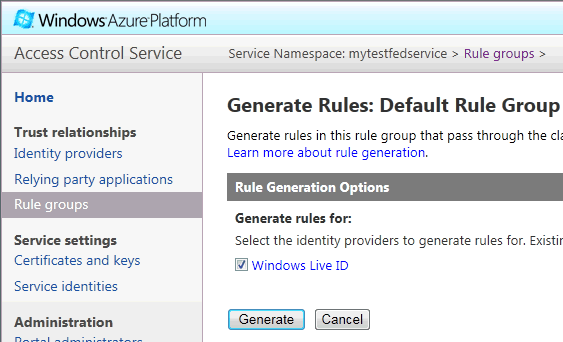
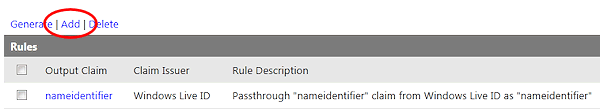
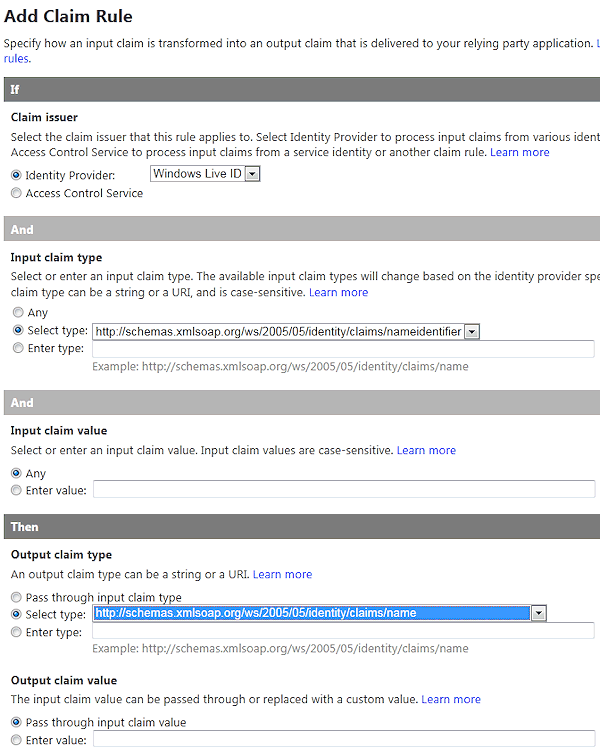
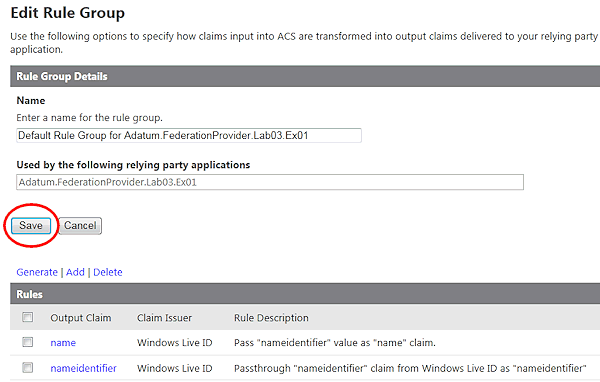
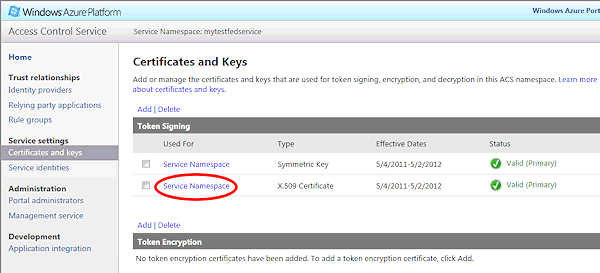
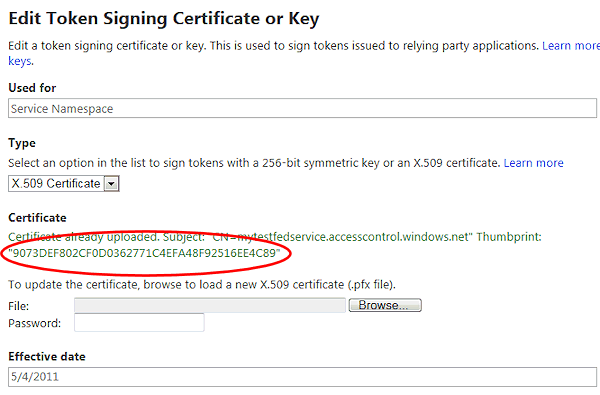
* + [Task 1](#Ex01Task01): Configure ACS for the Adatum federation provider.
  + [Task 2](#Ex01Task02): Modify the Adatum federation provider.

You should be able to complete this exercise in approximately 30 minutes.

## Task 1: Configure ACS for the Adatum Federation Provider

* 1. In this task you will configure ACS so that the Adatum federation provider is a relying party. ACS will use the Windows Live ID identity provider to authenticate users, and you will add rules to ACS that pass the required claims received from Windows Live ID through to the token that ACS will return to the Adatum federation provider.

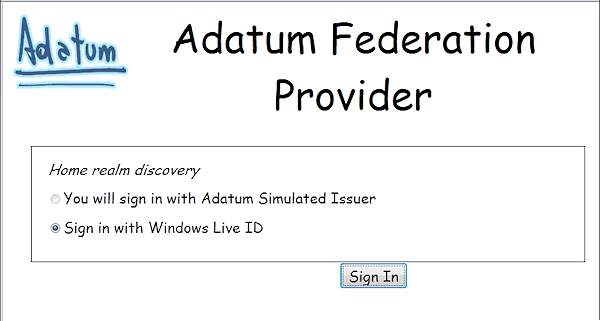
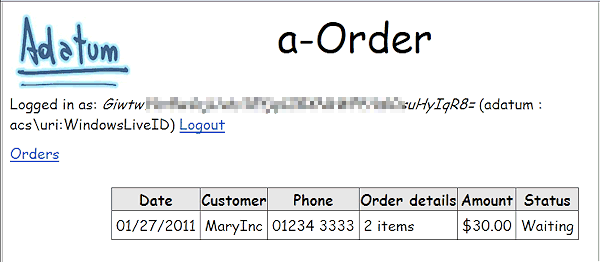
To configure ACS for the Adatum federation provider

* 1. Go to the Windows Azure Management Portal at <https://windows.azure.com/> and sign into your account. Then select **Service Bus, Access Control & Caching** in the lower section of the left pane and click **Access Control** in the list of services in the **AppFabric** tree.
     1. 
  2. Click the **New Namespace** icon at the top of the window and enter the name to use for your ACS namespace. This is the name you will enter into the Web.config files of your applications that use this ACS service (you may need to edit this value if the one you enter is not available). Then select a suitable geographical location from the **Country/Region** drop-down list and click **Create Namespace**.
     1. 
  3. Back in the main page wait until the **Status** column in the list of namespaces shows that the new namespace is active. Then select it and click the Access Control Service icon at the top of the main window to go to the Access Control administration pages.
     1. 
  4. In the Access Control Service page, click **Relying party applications** in the left pane or in the Getting Started section of the main page.
     1. 
  5. In the Relying Party Applications page click the **Add** link.
     1. 
  6. Enter the following values in the Add Relying Party Application page:
     + Name: **Adatum.FederationProvider.Lab03.Ex01**
     + Mode: **Enter settings manually**
     + Realm: **https://localhost/Adatum.FederationProvider.Lab03.Ex01**
     + Return URL: **https://localhost/Adatum.FederationProvider.Lab03.Ex01/Federation.aspx**
     1. 
  7. Leave the remainder of the settings at their default values and click **Save** to create the new relying party entry. You will see it in the list on the Relying Party Applications page.
     1. 
  8. Adding a relying party with the default settings automatically adds the Windows Live ID identity provider to the configuration and creates a new default rule group. Select **Rule Groups** in the left pane of the window and click the new default rule group.
     1. 
  9. In the Edit Rule Group page, click the **Generate** link to generate a default rule for this rule group.
     1. 
  10. This displays the Generate Rules page where you specify which of the configured identity providers you want to create the rules for. Windows Live ID is already selected, so just click **Generate**.
      1. 
  11. The new rule named **nameidentifier** is shown in the list of rules. This is a pass-through rule that copies the value of the **nameidentifier** claim received from the identity provider into the token that ACS will return to the relying party. You need to add another rule, so click the **Add** link above the list of rules.
      1. 
  12. The a-Order application requires a claim that is the name of the authenticated user. Windows Live ID does not disclose the user name, so for this exercise you will use the **nameidentifier** value to populate the **name** claim. Select or enter the following values in the Add Claim Rule page:
      + Identity Provider: **Windows Live ID**
      + Input claim type: **Select Type** and then select the claim type **http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier**
      + Input claim value: **Any**
      + Output claim type: **Select Type** and then select the claim type **http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name**
      + Output claim value: **Pass though the input claim value**
      + Description: **Pass "nameidentifier" value as "name" claim.**
      1. 
  13. Click **Save**. The new rule appears in the list in the Edit Rule Group page. Click the **Save** button above the list to update the rule group with the new rules.
      1. 
      2. You have now completed configuration of ACS, but before you close the portal you must find the thumbprint value of the token signing certificate, which your application requires for checking the signature of the token to ensure it is valid and has not been tampered with.
  14. In the Access Control Service page, click the **Certificates and keys** link in the left of the window to show the list of installed certificates. Click the **Service Namespace** link for the **X.509 Certificate** item.
      1. 
  15. The Edit Token Signing Certificate or Key page opens. It shows information about the certificate that ACS automatically configured for the namespace you added at the start of this task. This is the certificate used to sign the tokens sent from ACS to your relying party application. Make a note of the value of the thumbprint of this certificate. You can select it, copy it to the clipboard, and paste it into a new text document for use when you configure your applications.
      1. 
  16. Click the **Sign Out** link at the top of the portal window.
  17. You have now completed this task. You have configured ACS as a token issuer (STS) that uses Windows Live ID as the identity provider, and configured the Adatum federation provider as a relying party. In the next task you will modify the Adatum federation provider so that it can use the token received from ACS to authenticate users for the a-Order application.

## Task 2: Modify the Adatum Federation Provider

* 1. In this task you will modify the Adatum federation provider so that it trusts ACS. This will allow users to authenticate either through the Adatum simulated issuer, or by selecting a different identity provider that is supported by and exposed through ACS.

To modify the Adatum federation provider

* 1. Start Visual Studio as an administrator and open the solution named **Lab03.Ex01.sln** from the **Lab03-ACS\Source\Ex01\Begin** folder.
  2. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex01** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  3. Add the following highlighted line to the **<trustedIssuers>** element of the Web.config file to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of at the end of the previous task.
     1. XML
     2. <trustedIssuers>
     3. <!-- Adatum issuer certificate -->
     4. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     5. **<add thumbprint="{*your-acs-thumbprint-here*}" name="acs" />**
     6. </trustedIssuers>
  4. Save and close the Web.config file.
  5. In Solution Explorer double-click the file **TrustedIssuers.cs** in the **Adatum.FederationProvider.Lab03.Ex01** project to open it in the Visual Studio code editor. This class defines the token issuers that the Adatum federation provider trusts. Add the following highlighted code so that the federation provider will trust tokens issued by the ACS service namespace you configured in the previous exercise. Insert the name of this namespace into the code as indicated here.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. **public static readonly IssuerInfo Acs = new IssuerInfo("acs",**
     5. **"https://{*your-namespace-name*}.accesscontrol.windows.net/",**
     6. **"https://{*your-namespace-name*}.accesscontrol.windows.net/v2/wsfederation");**
     7. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     8. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex01/");
  6. Find the **GetAll** method (just below the definitions of the issuers) and add the following highlighted code so that the list of issuers returned from this method includes ACS.
     1. C#
     2. public static IEnumerable<IssuerInfo> GetAll()
     3. {
     4. return new IssuerInfo[] { Adatum**, Acs** };
     5. }
  7. Save and close the **TrustedIssuers.cs** file.
  8. In Solution Explorer double-click the file **FederationSecurityTokenService.cs** in the **Adatum.FederationProvider.Lab03.Ex01** project to open it in the Visual Studio code editor. This class contains methods used by the Adatum federation provider to generate claims for the tokens it returns. Add the following function named **CopySocialClaims** to the end of the **FederationSecurityTokenService** class, placing it immediately after the existing method named **CopyClaims**.
     1. C#
     2. private static void CopySocialClaims(IClaimsIdentity input, IClaimsIdentity output)
     3. {
     4. string issuer = input.Claims.Where(c => c.ClaimType.Equals(
     5. "http://schemas.microsoft.com/accesscontrolservice/2010/07/"
     6. + "claims/identityprovider")).FirstOrDefault().Value;
     7. string nameClaimValue;
     8. nameClaimValue = input.Claims.Where(c => c.ClaimType.Equals(
     9. ClaimTypes.Name)).FirstOrDefault().Value;
     10. string originalIssuer = string.Concat("acs\\", issuer);
     11. var claim = new Claim(ClaimTypes.Name, nameClaimValue,
     12. ClaimValueTypes.String, "adatum", originalIssuer);
     13. output.Claims.Add(claim);
     14. }
     15. This method takes the set of claims received from ACS and adds to it a new claim that indicates the original issuer of the token (the STS that created the token after authenticating the user). The value of this claim is a combination of the text "acs" and the name of the identity provider that authenticated the user. The method then adds the new claim to the set of claims that are being passed through from ACS to the a-Order application.
  9. Scroll up in the **FederationSecurityTokenService.cs** file to the **GetOutputClaimsIdentity** method. Add the following highlighted code to it so that the federation provider will call the new **CopySocialClaims** method, and will then add two more claims to the tokens it will issue.
     1. C#
     2. if (issuerInfo == TrustedIssuers.Adatum)
     3. {
     4. output = input;
     5. }
     6. **else if (issuerInfo == TrustedIssuers.Acs)**
     7. **{**
     8. **CopySocialClaims(input, output);**
     9. **output.Claims.Add(new Claim(ClaimTypes.Role, Adatum.Roles.OrderTracker));**
     10. **output.Claims.Add(new Claim(Adatum.ClaimTypes.Organization, "MaryInc"));**
     11. **}**
     12. SingleSignOnManager.RegisterIssuer(issuerInfo.Realm);
     13. The code above adds a claim to the token that indicates the user is a member of the **OrderTracker** group, and another claim that the user is an employee of the organization named **MaryInc**. Both of these are hard-coded into the federation provider, and so will be added to the token irrespective of the user who was authenticated by Windows Live ID. These claims are required by the a-Order application, and so must be added to the token. In a real world application the federation provider would look up these values, and other values such as the user name, from another directory or store based on the authenticated user identifier.
  10. Save and close the **FederationSecurityTokenService.cs** file.
  11. In Solution Explorer double-click the file **HomeRealmDiscovery.aspx** in the **Adatum.FederationProvider.Lab03.Ex01** project to open it in the Visual Studio code editor. This is the ASP.NET page that the user sees when an application accesses the federation provider, and is used to select the identity provider that the user will be directed to for authentication. Add the following highlighted code to the page so that it displays an option for users to select Windows Live ID as the identity provider.
      1. HTML
      2. <div id="issuerSelection">
      3. <span class="issuerSelection-title">
      4. Home realm discovery
      5. </span>
      6. <asp:RadioButton ID="TrustedIssuersRadioButton" runat="server"
      7. Text="You will sign in with Adatum Simulated Issuer"
      8. GroupName="issuer" Checked="true" Enabled="false"
      9. CssClass="box-title" />
      10. **<asp:RadioButton ID="SocialIssuerRadioButton" runat="server"**
      11. **Text="Sign in with Windows Live ID"**
      12. **GroupName="issuer" AutoPostBack="false"**
      13. **CssClass="box-title" />**
      14. </div>
  12. Right-click on the **HomeRealmDiscovery.aspx** page and click **View Code** to open the code-behind file in the Visual Studio editor. Add the following highlighted code to the **ProcessSignIn** method.
      1. C#
      2. public void ProcessSignIn()
      3. {
      4. if (this.TrustedIssuersRadioButton.Checked)
      5. {
      6. this.Response.Redirect("~/Federation.aspx?" + this.Request.QueryString
      7. + "&whr=" + TrustedIssuers.Adatum.Realm, false);
      8. }
      9. **if (this.SocialIssuerRadioButton.Checked)**
      10. **{**
      11. **this.Response.Redirect("~/Federation.aspx?" + this.Request.QueryString**
      12. **+ "&realm=" + TrustedIssuers.Acs.Realm, false);**
      13. **}**
      14. }
      15. This additional code redirects the request to the page **Federation.aspx** within this federation provider with the query string containing an additional name/value pair that specifies the realm to which the requesting application belongs. Notice that ACS requires you to specify the realm of the relying party during authentication using the **realm** name/value pair instead of the **whr** name/value pair more commonly used with WS-Federation requests.
  13. Save and close the **HomeRealmDiscovery.aspx** file. Then, in Solution Explorer, right-click the file **Federation.aspx** in the **Adatum.FederationProvider.Lab03.Ex01** project and click **View Code** to open the code-behind file for the main ASP.NET page of the federation provider in the Visual Studio code editor.
  14. Scroll down to the **HandleSignInRequest** method and edit the first line to specify the correct name for the name/value pair in the query string that denotes the realm, as shown in the following highlighted code.
      1. C#
      2. private void HandleSignInRequest()
      3. {
      4. var homeRealm = this.Request["**realm**"];
      5. string issuerLocation;
      6. ...
  15. Save and close the **Federation.aspx** file. Then, in Solution Explorer, right-click the top-level **Lab03.Ex01** solution item and click **Rebuild solution**. Ensure that it builds with no errors.
  16. In Solution Explorer **right-click the a-Order.OrderTracking.Lab03.Ex01 project item and click View in Browser. The WIF modules in the a-Order application pipeline detect that you are not authenticated, and redirect your browser to the Adatum federation provider.** 
      1. 
  17. **Select Sign in with Windows Live ID and click the Sign In button. The federation provider redirects your browser to ACS, which redirects it to the Windows Live login page.**
      1. 
  18. **Enter your Windows Live ID and password, and click Sign in. After successful authentication your browser is redirected to ACS, and from there back to the a-Order application. You will see the value of your Windows Live nameidentifier claim; which, you'll recall, is used to populate the name claim because Windows Live ID does not expose the user name.**
      1. 

1. You have now completed this task and this exercise. You have seen how you can configure ACS to act as the token issuer (STS), and how you can configure a relying party application. You have also seen how you can modify your application to use the tokens received from ACS to authenticate and authorize users.

## Running the "End" Solution

1. If you did not complete all of the tasks in this exercise, you can run the "end" solution we provide. Before you do you must have created a namespace instance within your own ACS service account. After you have created the namespace, you can use the following procedure to run the setup application we provide that will perform the appropriate configuration of ACS.
   1. You will need the value of the token signing certificate thumbprint to run the end solution. See the last steps of [Task 1 of Exercise 1](#Ex01Task01) in this lab for information on how to find this value.

To run the end solution

* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex01.End** from the folder **Lab03-ACS\Source\Ex01\End**.
  3. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex01** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  4. Scroll down to the **<trustedIssuers>** element and edit it to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of while performing the first task in Exercise 1 of this lab.
     1. XML
     2. <trustedIssuers>
     4. <!-- Adatum issuer certificate -->
     5. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     6. <!-- ACS issuer certificate -->
     7. <add thumbprint="{***your-acs-thumbprint-here*}**" name="acs" />
     9. </trustedIssuers>
  5. Save and close the Web.config file.
  6. In Solution Explorer double-click the file **TrustedIssuers.cs** to open it in the Visual Studio code editor. Modify the URLs of the ACS issuer so that they contain the name of the namespace you have defined in ACS, as shown in the following highlighted code.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     5. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex01.End/");
     6. public static readonly IssuerInfo Acs = new IssuerInfo("acs",
     7. "https://{***your-namespace-name*}**.accesscontrol.windows.net/",
     8. "https://{***your-namespace-name*}**.accesscontrol.windows.net/v2/wsfederation");
  7. Save and close the **TrustedIssuers.cs** file.
  8. Expand the **ACS.Setup.Lab03.Ex01.End** project item and double-click the file **Program.cs** to open it in the Visual Studio editor. Edit the following two highlighted values near the start of the file to specify the management password (key) for your Azure account and the name of the namespace to use for the example.
     1. C#
     2. internal class Program
     3. {
     4. private const string AcsPassword = "{**your-acs-password}**";
     5. private const string AcsServiceNamespace = "{**your-namespace-name}**";
     6. private const string AcsUsername = "ManagementClient";
     7. ...
     8. This code uses the default management account in ACS named **ManagementClient**. For help locating the correct values to use for the password key and namespace, see the section "*Locating Your ACS Password Key and Namespace*" at the start of this lab.
  9. Save and close the Program.cs file, open the **Build** menu, and click **Rebuild Solution**.
  10. Right-click on the **ACS.Setup.Lab03.Ex01.End** project item, point to **Debug**, and click **Start new instance**. This will remove any existing instance of a relying party that has the name **Adatum.FederationProvider.Lab03.Ex01.End**, then add it again and configure the required rules.
  11. In Solution Explorer **right-click the a-Order.OrderTracking.Lab03.Ex01.End project item and click View in Browser. You are redirected to the Adatum federation provider.**
  12. **In the Adatum federation provider page select Sign in with Windows Live ID and click the Sign In button. The federation provider and ACS redirect your browser to the Windows Live login page.**
  13. **Enter your Windows Live ID and password, and click Sign in. After successful authentication your browser is redirected to ACS, and from there back to the a-Order application. You will see the value of your Windows Live nameidentifier claim; which, you'll recall, is used to populate the name claim because Windows Live ID does not expose the user name.**

# Exercise 2: Adding the Facebook Identity Provider and Home Realm Discovery

* 1. In this exercise you will add Facebook as an identity provider to your ACS namespace. This illustrates how, by taking advantage of ACS, you can easily change the options a user has for authentication when using your applications; without requiring any modification of the application or of your own local token issuer or federation provider.
  2. In this example you must make some minor changes to the configuration and code of the federation provider because it must contain a reference to the unique namespace name and certificate thumbprint of your own ACS instance.

This exercise contains the following tasks:

* + [Task 1](#Ex02Task01): Configure Facebook as an identity provider in ACS.
  + [Task 2](#Ex02Task02): Update the federation provider to use your ACS instance.

You should be able to complete this exercise in approximately 20 minutes.

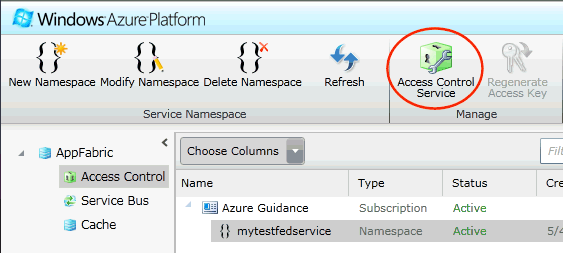
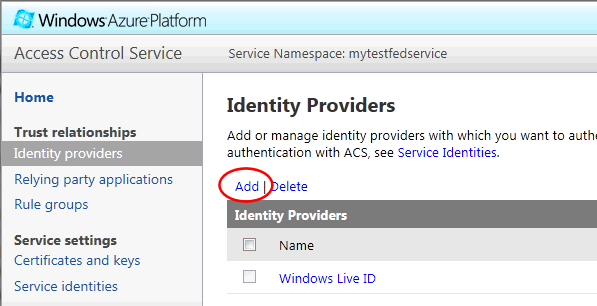
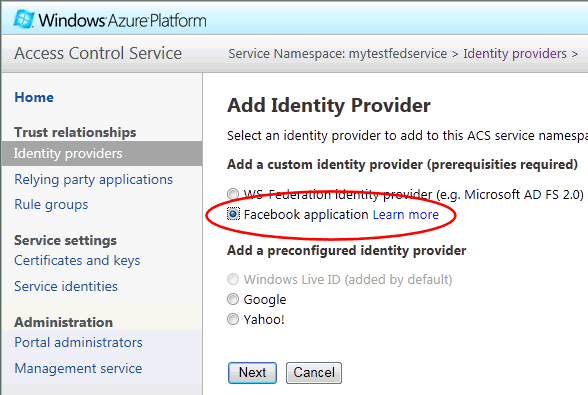
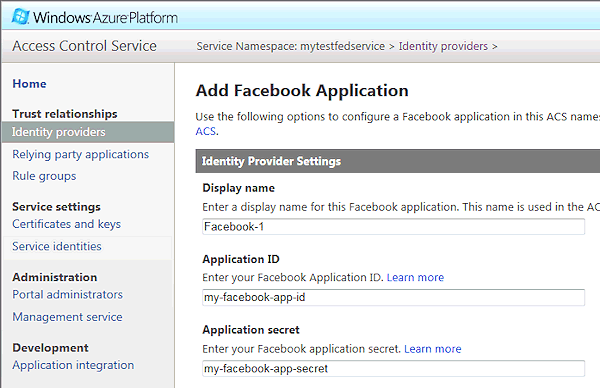
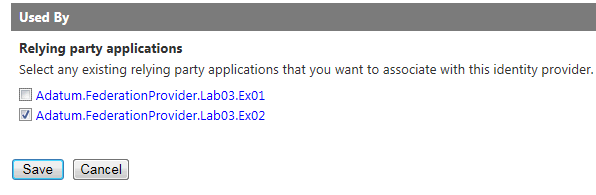
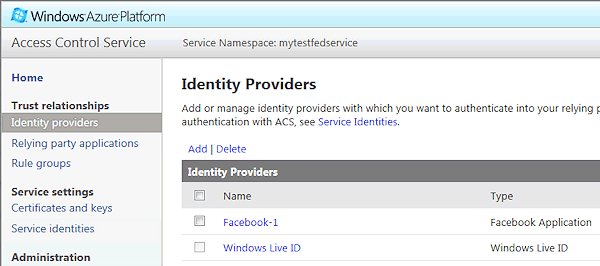
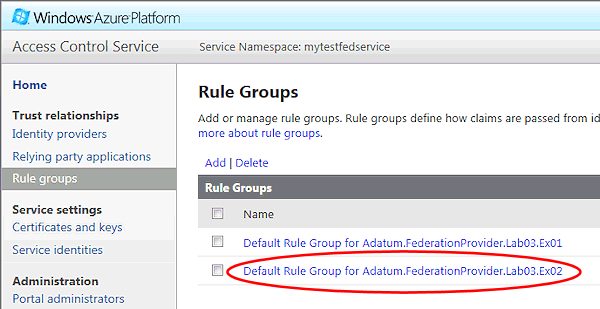
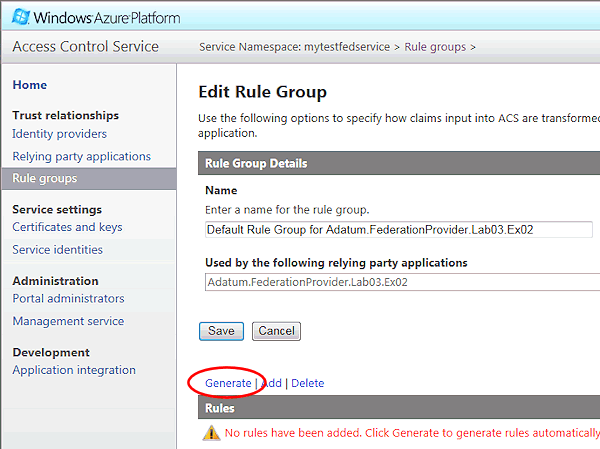
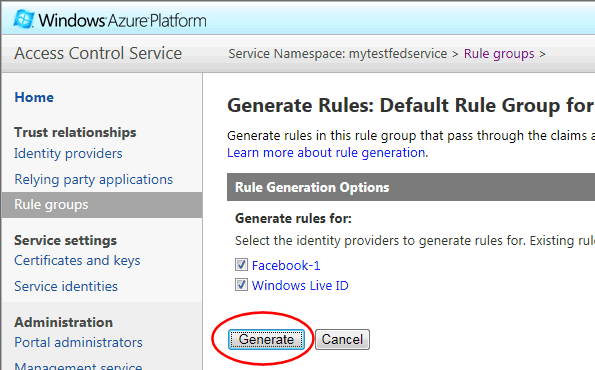
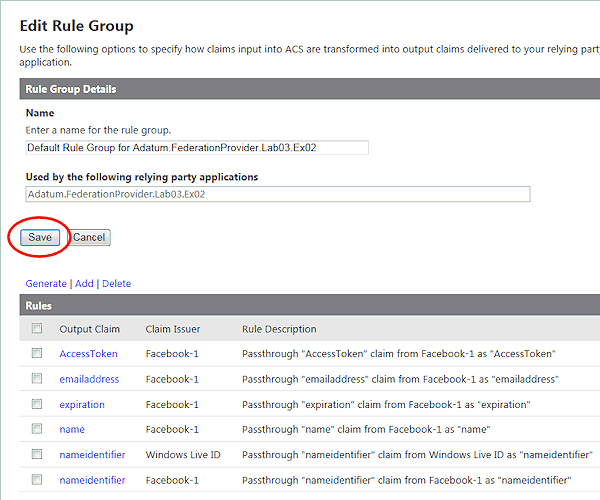
## Obtaining a Facebook Application ID and Facebook Application Secret

* 1. As part of the process of configuring Facebook as an identity provider, you must specify your own Facebook application ID and application secret values. To obtain these, and to set up your account to allow ACS to use Facebook for authentication, you must log into the Facebook developer's portal at <http://www.facebook.com/developers/>. If you do not already have a Facebook account you can create one at <http://facebook.com>.
  2. Once logged into the portal go to the Developers page, click **Create a New Application** and follow the instructions. After the process is complete, open the Edit page for your application and click **Web Site** in the left pane. Make a note of the **Application ID** and **Application Secret** values. Finally, enter the complete URL of your ACS namespace in the format **https://***your-namespace-name***.accesscontrol.windows.net**, and then click **Save Changes**.

## Task 1: Configure Facebook as an Identity Provider in ACS

* 1. In this task you will add Facebook as an identity provider to your ACS namespace, and define the Adatum federation provider as a relying party that uses the Facebook identity provider.

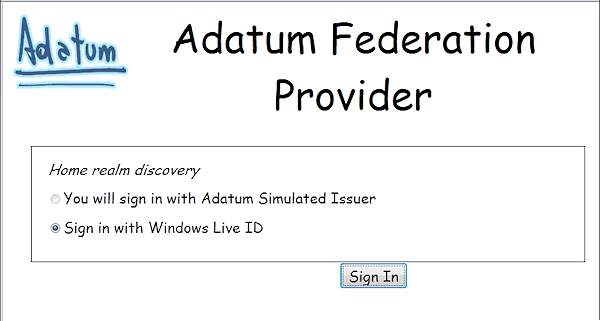
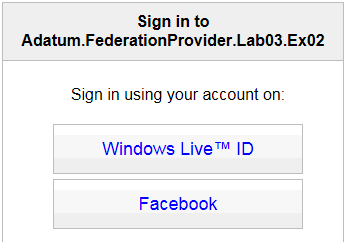
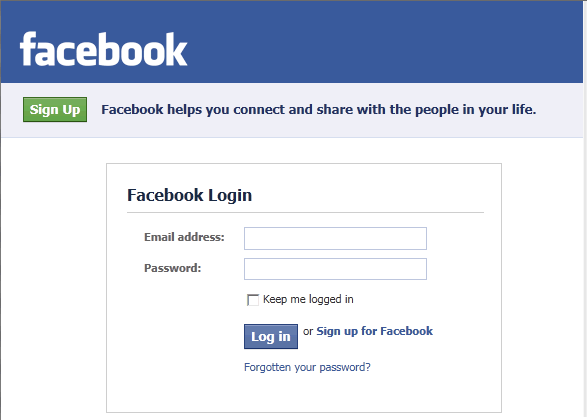
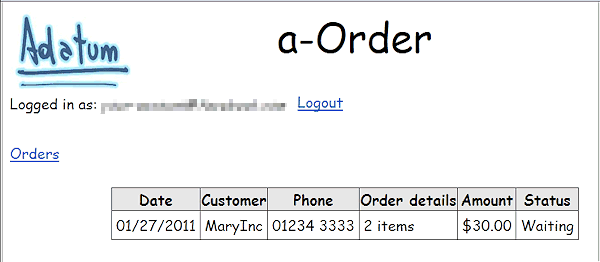
To configure Facebook as an identity provider in ACS

* 1. Go to the Windows Azure Management Portal at <https://windows.azure.com/> and sign into your account. Then select **Service Bus, Access Control & Caching** in the lower section of the left pane and click **Access Control** in the list of services in the **AppFabric** tree.
  2. Select the namespace you created in the previous exercise and click the **Access Control Service** icon at the top of the main window to go to the Access Control administration pages.
     1. 
  3. In the Access Control Service page, click **Relying party applications** in the left pane or in the Getting Started section of the main page.
  4. In the Relying Party Applications page click the **Add** link and enter the following values in the Add Relying Party Application page:
     + Name: **Adatum.FederationProvider.Lab03.Ex02**
     + Mode: **Enter settings manually**
     + Realm: **https://localhost/Adatum.FederationProvider.Lab03.Ex02**
     + Return URL: **https://localhost/Adatum.FederationProvider.Lab03.Ex02/Federation.aspx**
  5. Leave the remainder of the settings at their default values and click **Save** to create the new relying party entry. You will see it in the list on the Relying Party Applications page.
     1. Adding a relying party with the default settings automatically adds the Windows Live ID identity provider to the configuration and creates a new default rule group. However, in this exercise, you want to use the Facebook identity provider.
  6. Click **Identity providers** in the left pane. In the Identity Providers page notice that, as in the previous exercise, only Windows Live ID is configured by default. Click the **Add** link above the list of identity providers.
     1. 
  7. In the Add Identity Provider page select **Facebook application** and then click **Next**.
     1. 
  8. In the Add Facebook Application page enter the name to display for the application, your Facebook application ID, and your Facebook application secret.
     1. 
     2. For information about how and where you get the values for your Facebook application ID and Facebook application secret, see the section "*Obtaining a Facebook Application ID and Facebook Application Secret*" at the start of this task or visit <http://msdn.microsoft.com/en-us/library/gg185919.aspx> on MSDN.
  9. Scroll to the end of the Add Facebook Application page. You will see that, by default, the Facebook provider will be configured for all of the relying parties you have configured for this namespace. Un-tick the relying party for the previous exercise (Adatum.Federation.Provider.Lab03.Ex01), but ensure that the **Adatum.Federation.Provider.Lab03.Ex02** option is selected.
     1. 
  10. Click the **Save** button. You will see the new Facebook Application identity provider in the list on the Identity Providers page.
      1. 
  11. Click **Rule Groups** in the left pane of the window. In the Rule Groups page click the new default rule group that was created for the **Adatum.Federation.Provider.Lab03.Ex02** relying party.
      1. 
  12. In the Edit Rule Group page, click the **Generate** link to generate a default rule for this rule group.
      1. 
  13. This displays the Generate Rules page where you specify which of the configured identity providers you want to create the rules for. Both the Facebook and Windows Live identity providers are already selected, so just click **Generate** to generate the default rules for both.
      1. 
  14. The Edit Rule Group page reappears showing the rules that were generated. As in the previous exercise, there is a rule named **nameidentifier** for the Windows Live ID claims issuer (identity provider). There is also a rule named **nameidentifier** for the Facebook identity provider. In addition, ACS has generated several other rules that pass the values of the other claims that Facebook places in tokens it issues.
      1. 
      2. The a-Order application requires a claim that is the name of the authenticated user. Windows Live ID does not disclose the user name, so in the previous exercise you created a rule to pass the **nameidentifier** value as the **name** claim. Facebook exposes the name claim, which you are passing through to the application, so you do not require any additional rules.
      3. You can specify which claims Facebook will include in the tokens it issues when you define the settings for the Facebook identity provider. In the previous steps of this task you accepted the default setting, which causes only the email address of the authenticated user (together with the four standard claims **AccessToken**, **expiration**, **name**, and **nameidentifier**) to be included in the token.
      4. Note that, when only the rules listed above are configured, you will not be able to use the Windows Live ID identity provider to authenticate because it does not return a **name** claim by default. If you want to be able to use both identity providers, you must also carry out steps 11, 12, and 13 of Task 1 in Exercise 1.
  15. Click the **Save** button above the list to update the rule group with the new rules.
  16. Click the **Sign Out** link at the top of the portal window.
  17. You have now completed this task. You have configured ACS to act as token issuer for the a-Order application by configuring theAdatum federation provider for this exercise as a relying party, and configured ACS to use Facebook as an identity provider in addition to Windows Live ID.

## Task 2: Update the Federation Provider to Use Your ACS Instance

* 1. In this task you will make some minor changes to the configuration and code of the federation provider because it must contain a reference to the unique namespace name and certificate thumbprint of your own ACS instance. Then you will run the application and see the results of using the Facebook identity provider.

To update the federation provider to use your ACS instance

* 1. Start Visual Studio as an administrator and open the solution named **Lab03.Ex02.sln** from the **Lab03-ACS\Source\Ex02\Begin** folder.
  2. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex02** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  3. Scroll down to the **<trustedIssuers>** element and edit it to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of while performing the first task in Exercise 1 of this lab.
     1. XML
     2. <trustedIssuers>
     4. <!-- Adatum issuer certificate -->
     5. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     6. <!-- ACS issuer certificate -->
     7. <add thumbprint="{***your-acs-thumbprint-here*}**" name="acs" />
     9. </trustedIssuers>
  4. Save and close the Web.config file.
  5. In Solution Explorer double-click the file **TrustedIssuers.cs** in the **Adatum.FederationProvider.Lab03.Ex02** project to open it in the Visual Studio code editor. Modify the URLs of the ACS issuer so that they contain the name of the namespace you have defined in ACS, as shown in the following highlighted code.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     5. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex02/");
     6. public static readonly IssuerInfo Acs = new IssuerInfo("acs",
     7. "https://{***your-namespace-name*}**.accesscontrol.windows.net/",
     8. "https://{***your-namespace-name*}**.accesscontrol.windows.net/v2/wsfederation");
  6. Save and close the **TrustedIssuers.cs** file.
  7. In Solution Explorer right-click the top-level **Lab03.Ex02** solution item and click **Rebuild solution**. Ensure that it builds with no errors.
  8. In Solution Explorer **right-click the a-Order.OrderTracking.Lab03.Ex02 project item and click View in Browser. The WIF modules in the a-Order application pipeline detect that you are not authenticated, and redirect your browser to the Adatum federation provider.** 
     1. 
  9. **Select Sign in with Windows Live ID (this text reflects the modifications you made in the previous exercise; you will actually be logging in using Facebook not Windows Live ID). Then click the Sign In button. The federation provider redirects your browser to ACS. Because there is more than one identity provider configured, ACS displays the Home Realm Discovery page where users can select the identity provider to use.**
     1. 
     2. **The Home Realm Discovery page allows users to specify which realm (identity provider) holds their account. It is only displayed when more than one identity provider is configured for the relying party application. In this case it is displayed because you configured Facebook as an identity provider in addition to the default Windows Live ID identity provider. However, unlike in Exercise 1 of this lab, you did *not* configure the additional rule in ACS to populate the name claim required by the a-Order application for the Windows Live ID identity provider. This means that you cannot use the Widows Live ID identity provider to authenticate in this example because the a-Order application requires a name claim and ACS will not send one for the Windows Live ID identity provider. If you want to enable both identity providers, you must also carry out steps 11, 12, and 13 in Task 1 of Exercise 1.**
  10. **Click the Facebook button. ACS redirects your browser to the Facebook login page.** 
      1. 
  11. **Enter your credentials and click Log in. After successful authentication you see a page that asks for permission to send the token to ACS. Click Allow. Your browser is redirected to ACS, and from there back to the a-Order application.** 
      1. 

1. You have now completed this task and this exercise. You have seen how you can configure ACS to use more than one identity provider, and how it automatically displays a Home Realm Discovery page when there is more than one identity provider configured for a relying party application.

## Running the "End" Solution

1. If you did not complete all of the tasks in this exercise, you can run the "end" solution we provide. Before you do you must have created a namespace instance within your own ACS service account. After you have created the namespace, you can use the following procedure to run the setup application we provide that will perform the appropriate configuration of ACS.
   1. You will need the value of the token signing certificate thumbprint to run the end solution. See the last steps of [Task 1 of Exercise 1](#Ex01Task01) in this lab for information on how to find this value.
   2. You must also have obtained the application ID and secret for your own Facebook application to be able to run the End solution. See the section "[Obtaining a Facebook Application ID and Facebook Application Secret](#_Obtaining_a_Facebook)" at the start of this exercise for more information.

To run the end solution

* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex02.End** from the folder **Lab03-ACS\Source\Ex02\End**.
  3. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex02** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  4. Scroll down to the **<trustedIssuers>** element and edit it to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of while performing the first task in Exercise 1 of this lab.
     1. XML
     2. <trustedIssuers>
     4. <!-- Adatum issuer certificate -->
     5. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     6. <!-- ACS issuer certificate -->
     7. <add thumbprint="{***your-acs-thumbprint-here*}**" name="acs" />
     9. </trustedIssuers>
  5. Save and close the Web.config file.
  6. In Solution Explorer double-click the file **TrustedIssuers.cs** to open it in the Visual Studio code editor. Modify the URLs of the ACS issuer so that they contain the name of the namespace you have defined in ACS, as shown in the following highlighted code.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     5. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex02.End/");
     6. public static readonly IssuerInfo Acs = new IssuerInfo("acs",
     7. "https://{***your-namespace-name*}**.accesscontrol.windows.net/",
     8. "https://{***your-namespace-name*}**.accesscontrol.windows.net/v2/wsfederation");
  7. Save and close the **TrustedIssuers.cs** file.
  8. Expand the **ACS.Setup.Lab03.Ex02.End** project item and double-click the file **Program.cs** to open it in the Visual Studio editor. Edit the following two highlighted values near the start of the file to specify the management password key for your Azure account and the name of the namespace to use for the example.
     1. C#
     2. internal class Program
     3. {
     4. private const string AcsPassword = "{***your-acs-password*}**";
     5. private const string AcsServiceNamespace = "{***your-namespace-name*}**";
     6. private const string AcsUsername = "ManagementClient";
     7. ...
     8. This code uses the default management account in ACS named **ManagementClient**. For help locating the correct values to use for the password key and namespace, see the section "*Locating Your ACS Password Key and Namespace*" at the start of this lab.
  9. Locate the following line of code in the **Main** routine of the Program.cs file and replace the two highlighted values with the application ID and application secret for your own Facebook application.
     1. C#
     2. acs.AddFacebookIdentityProvider("Facebook",
     3. "**{*your-facebook-application-id*}**", "**{*your-facebook-application-secret*}**");
  10. Save and close the Program.cs file, open the **Build** menu, and click **Rebuild Solution**.
  11. Right-click on the **ACS.Setup.Lab03.Ex02.End** project, point to **Debug**, and click **Start new instance**. This will remove any existing instance of a relying party that has the name **Adatum.FederationProvider.Lab03.Ex02.End**, then add it again and configure the required rules.
  12. In Solution Explorer **right-click the a-Order.OrderTracking.Lab03.Ex02.End project item and click View in Browser. You are redirected to the Adatum federation provider.**
  13. **In the Adatum federation provider page select Sign in with a Social Identity and click the Sign In button. The federation provider redirects your browser to the ACS Home Realm Discovery page.**
  14. **Click Facebook, and ACS redirects your browser to the Facebook login page.**
  15. **Enter your Facebook credentials, and click Log in. After successful authentication your browser is redirected to ACS, and from there back to the a-Order application.**

# Exercise 3: Adding a Custom OpenID Identity Provider

* 1. In this exercise you will add OpenID as an identity provider to ACS. The ACS web portal you used in the previous exercises does not support OpenID as a configurable identity provider, so you must add it as a custom identity provider using the Management API exposed by ACS. We provide a wrapper class for you to use in this exercise that simplifies access to the ACS Management API.

This exercise contains the following tasks:

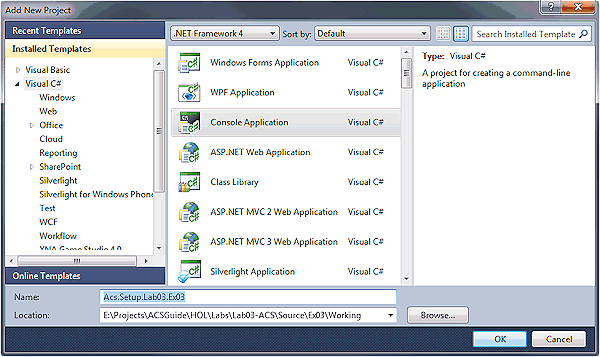
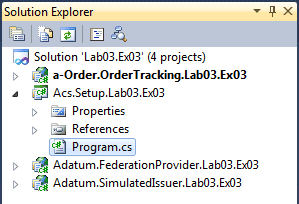
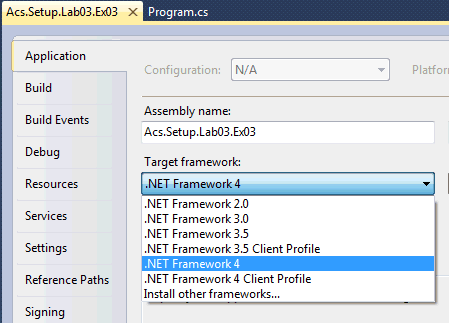
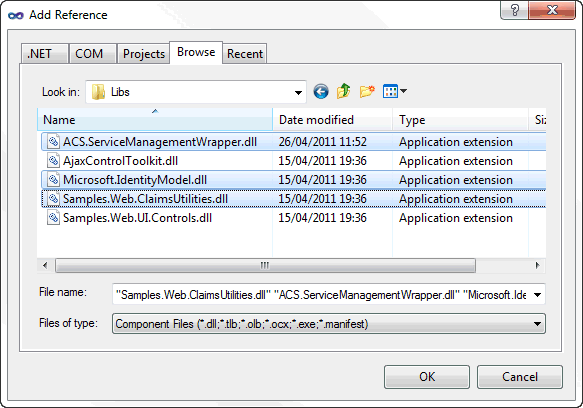
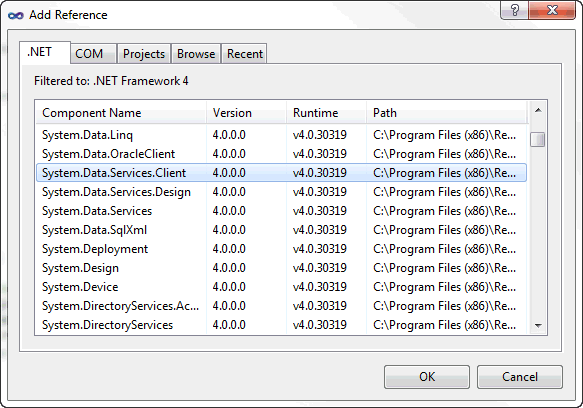
* + [Task 1](#Ex03Task01): Create the setup program that uses the ACS Management API
  + [Task 2](#Ex03Task02): Run the setup program and examine the results

You should be able to complete this exercise in approximately 20 minutes.

## Task 1: Create the Setup Program that Uses the ACS Management API

In this task you will create the setup program as a Console application. This program will use a custom wrapper class that we provide to access the ACS Management API.

To create the setup program that uses the ACS Management API

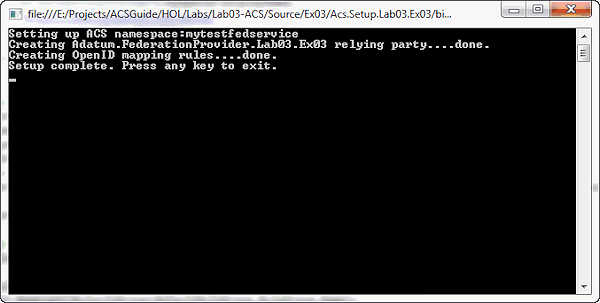
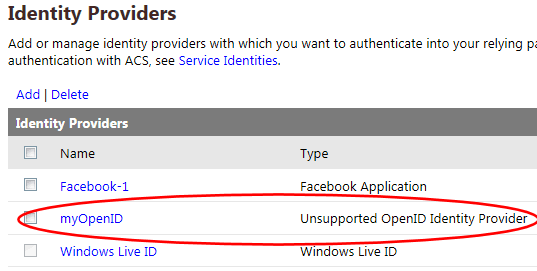
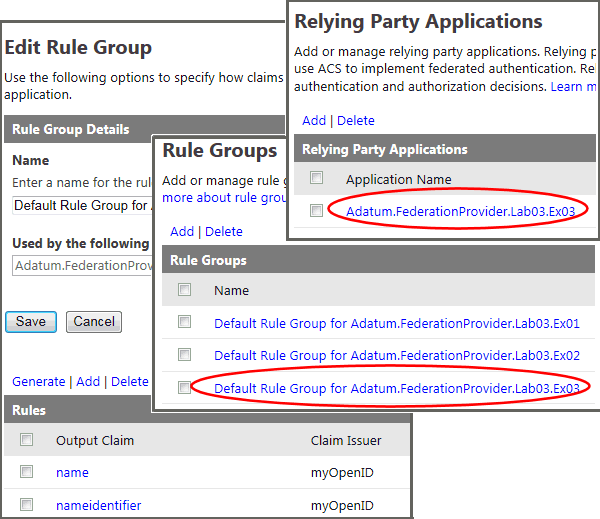
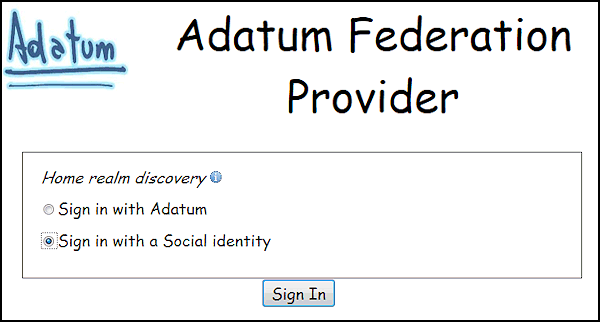
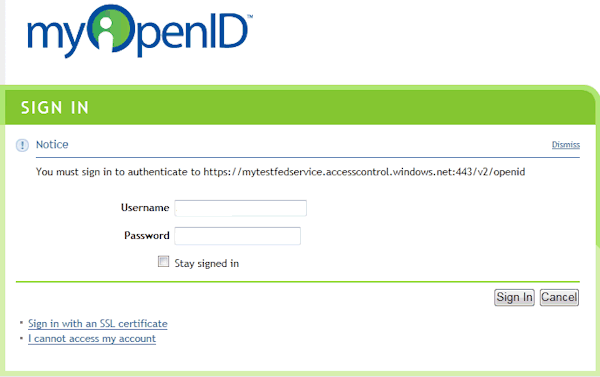
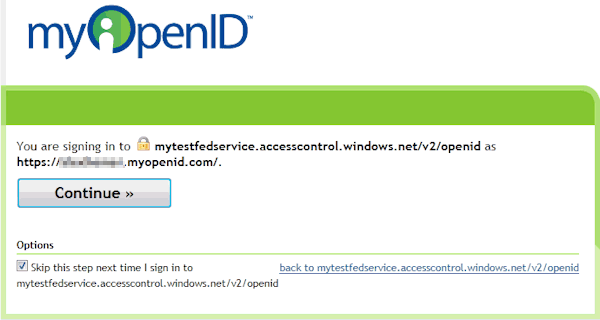
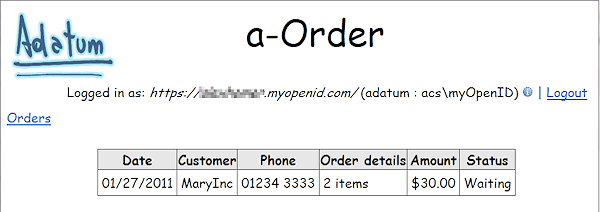
* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex03** from the folder **Lab03-ACS\Source\Ex03\Begin**.
  3. In Solution Explorer right-click the top-level solution item, point to **Add**, and click **New Project**. In the New project dialog select the **Visual C# Console Application** and change the name to **Acs.Setup.Lab03.Ex03**.
     1. 
  4. Click **OK**. The new project appears in Solution Explorer and the file **Program.cs** opens in the Visual Studio code editor.
     1. 
  5. In Solution Explorer right-click the **Acs.Setup.Lab03.Ex03** project item and click **Properties**. Ensure that the Target framework is set to **.NET Framework 4** (*not* .NET Framework 4 Client Profile). If you need to change the setting, you must close and then re-open the solution afterwards.
     1. 
  6. In Solution Explorer right-click the **Acs.Setup.Lab03.Ex03** project item and click **Add Reference**. In the Add Reference dialog open the **Browse** tab and navigate to the folder **Lab03-ACS//Source/Shared Code/Libs**. Select the three assemblies named **ACS.ServiceManagementWrapper**, **Samples.Web.ClaimsUtilities**, and **Microsoft.IdentityModel**.
     1. 
  7. Click **OK** to add the assembly references to your project. Then right-click the **Acs.Setup.Lab03.Ex03** project item and click **Add Reference** again. In the Add Reference dialog open the **.NET** tab and select the assembly named **System.Data.Services.Client**.
     1. 
  8. Click **OK** to add the assembly reference to your project.
  9. In the **Program.cs** file delete all of the existing content and replace it with the following code that declares the namespace, references the types it will use, and makes the class internal.
     1. C#
     2. namespace Acs.Setup
     3. {
     4. using System;
     5. using System.Linq;
     6. using ACS.ServiceManagementWrapper;
     7. using ACS.ServiceManagementWrapper.ACS.Management;
     8. using Microsoft.IdentityModel.Claims;
     9. internal class Program
     10. {
     11. static void Main()
     12. {
     13. }
     14. }
     15. }
  10. Add the following highlighted code to the class to declare the password (key), namespace, and user name that the program will use to access the ACS Management API.
      1. C#
      2. internal class Program
      3. {
      4. private const string MyAcsPassword = "***{your-acs-password}***";
      5. private const string MyAcsServiceNamespace = "***{your-acs-namespace}***";
      6. **private const string MyAcsUsername = "ManagementClient";**
      7. static void Main()
      8. {
      9. This code uses the default management account in ACS named **ManagementClient**. For help locating the correct values to use for the password key and namespace, see the section "*Locating Your ACS Password Key and Namespace*" at the start of this lab.
  11. Now you can create the code that will call the Management API. Add the following highlighted code to the **Main** method to create a new instance of the service management wrapper class, and add the OpenID identity provider to your namespace in ACS.
      1. C#
      2. static void Main()
      3. {
      4. **var acsWrapper = new ServiceManagementWrapper(MyAcsServiceNamespace,**
      5. **MyAcsUsername, MyAcsPassword);**
      6. **Console.WriteLine("Setting up ACS namespace:" + MyAcsServiceNamespace);**
      7. **try**
      8. **{**
      9. // Add identity provider for OpenID
      10. **acsWrapper.AddIdentityProviderManually(**
      11. **SocialIdentityProviders.MyOpenID.DisplayName,**
      12. **"https://www.myopenid.com/server",**
      13. **WebSSOProtocolType.OpenId);**
      14. ... code to create relying party goes here ...
      15. ... code to add rules goes here ...
      16. **}**
      17. **catch (Exception ex)**
      18. **{**
      19. **Console.WriteLine(ex.Message);**
      20. **}**
      21. **Console.WriteLine("Setup complete. Press any key to exit.");**
      22. **Console.ReadKey();**
      23. }
      24. If you are creating a program that you need to execute more than once, you should consider including code that removes any existing identity providers, relying parties, and rules that may conflict if you try to add them again. If you decide to perform this task again, you should manually remove the relevant identity providers, relying parties, and rules from your ACS namespace before re-running the setup program.
  12. Next you must add the relying party for this exercise. Add the following highlighted code to specify the name for the relying party, define the realm and reply addresses, specify the identity providers to use for this relying party, and then add the relying party to the namespace in ACS.
      1. C#
      2. ...
      3. WebSSOProtocolType.OpenId);
      4. // Specify the relying party name
      5. **var relyingPartyName = "Adatum.FederationProvider.Lab03.Ex03";**
      6. // Create the relying party in ACS
      7. **Console.Write(string.Format("Creating {0} relying party....", relyingPartyName));**
      8. **var realmAddress**
      9. **= "https://localhost/Adatum.FederationProvider.Lab03.Ex03/";**
      10. **var replyAddress**
      11. **= "https://localhost/Adatum.FederationProvider.Lab03.Ex03/Federation.aspx";**
      12. **var ruleGroup = string.Format("Default Rule Group for {0}", relyingPartyName);**
      13. **var ips = new[] { SocialIdentityProviders.MyOpenID.DisplayName };**
      14. **acsWrapper.AddRelyingParty(relyingPartyName, realmAddress, replyAddress,**
      15. **null, null, null, ruleGroup, ips);**
      16. **Console.WriteLine("done.");**
      17. ... code to add rules goes here ...
  13. Now add the following highlighted code to the **Main** method to remove any existing rules for this relying party and add two rules that pass through the display name claim returned by OpenID as the **nameidentifier** and **name** claims (which, you'll recall, are required by the a-Order application).
      1. C#
      2. ...
      3. Console.WriteLine("done.");
      4. // Remove any existing rules from rule group for this relying party
      5. **var relyingParty = acsWrapper.RetrieveRelyingParties().Single(**
      6. **rp => rp.Name == relyingPartyName);**
      7. **var defaultRuleGroup = relyingParty.RelyingPartyRuleGroups.FirstOrDefault();**
      8. **acsWrapper.RemoveAllRulesInGroup(defaultRuleGroup.RuleGroup.Name);**
      9. **Console.Write("Creating Open ID mapping rules....");**
      10. **var name = SocialIdentityProviders.MyOpenID.DisplayName;**
      11. // Add rule to pass though nameidentifier claim
      12. **acsWrapper.AddPassThroughRuleToRuleGroup(defaultRuleGroup.RuleGroup.Name,**
      13. **name, ClaimTypes.NameIdentifier);**
      14. // Add rule to pass though name claim
      15. **acsWrapper.AddPassThroughRuleToRuleGroup(defaultRuleGroup.RuleGroup.Name,**
      16. **name, ClaimTypes.NameIdentifier, ClaimTypes.Name);**
      17. **Console.WriteLine("done.");**
      18. }
      19. ...
  14. Open the **Build** menu and click **Rebuild Solution**. Ensure that the solution builds without errors.

You have now completed this task. In the next task you will use the setup program you have created to programmatically configure ACS with the OpenID provider, a relying party, and the required rules for this exercise.

## Task 2: Run the Setup Program and Examine the Results

In this task you will use the setup program you have created to programmatically configure ACS with the OpenID provider, a relying party, and the required rules; and then examine the results in the ACS web portal and by executing the a-Order application.

To run the setup program and examine the results

* 1. Continue with the **Lab03.Ex03** solution you used in the previous task.
  2. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex03** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  3. Scroll down to the **<trustedIssuers>** element and edit it to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of while performing the first task in Exercise 1 of this lab.
     1. XML
     2. <trustedIssuers>
     4. <!-- Adatum issuer certificate -->
     5. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     6. <!-- ACS issuer certificate -->
     7. <add thumbprint="**{*your-acs-thumbprint-here*}**" name="acs" />
     9. </trustedIssuers>
  4. Save and close the Web.config file.
  5. In Solution Explorer double-click the file **TrustedIssuers.cs** to open it in the Visual Studio code editor. Modify the URLs of the ACS issuer so that they contain the name of the namespace you have defined in ACS, as shown in the following highlighted code.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     5. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex03/");
     6. public static readonly IssuerInfo Acs = new IssuerInfo("acs",
     7. "https://{***your-namespace-name*}**.accesscontrol.windows.net/",
     8. "https://{***your-namespace-name*}**.accesscontrol.windows.net/v2/wsfederation");
  6. Save and close the **TrustedIssuers.cs** file.
  7. In Solution Explorer right-click the **Acs.Setup.Lab03.Ex03** project item, point to **Debug**, and click **Start new instance**. You will see the results as the program configures ACS in the command window.
     1. 
  8. Press any key to close the window, and then open the ACS web portal in your browser. Sign in, go to the **Access Control** pages, and click the **Identity Providers** link. You will see the OpenID identity provider in the list.
     1. 
     2. Notice that it is shown as "Unsupported", and clicking the link does nothing. There are no configurable properties for this identity provider.
  9. In the ACS web portal look at the remaining settings that the setup program has configured for the **Adatum.FederationProvider.Lab03.Ex03** relaying party application. You will see there is a rule group containing the two pass-through rules for the **name** and **nameidentifier** claims required by the a-Order application.
     1. 
  10. In Visual Studio Solution Explorer right-click the **a-Order.OrderTracking.Lab03.Ex03** project item and click **View in Browser**. Because you are not currently authenticated, the WIF modules in the a-Order application pipeline redirect your browser to the Adatum federation provider. When the **Adatum Federation Provider** page appears, select **Sign in with a Social identity** and click the **Sign In** button.
      1. 
  11. The federation provider redirects your browser to ACS. As there is only the OpenID identity provider configured, ACS immediately redirects your browser to the login page URL of the OpenID provider you specified in the setup program (in this example, <https://www.myopenid.com/server>).
      1. 
  12. Enter your OpenID credentials and click the **Sign In** button. After successful authentication the myOpenID identity provider displays a confirmation including the URL of the site that requested authentication. Notice that users can specify this confirmation page is not shown again.
      1. 
  13. Click **Continue**. Your browser is redirected back to ACS, which redirects it back to the a-Order application. You can see the values of the **name** and **nameidentifier** claim in the page.
      1. Insert Caption
      2. 

You have now completed this task and this exercise. You have seen how ACS supports configuration using the Management API, and how you can use it to add identity providers, relying parties, and claims rules. You have also seen how you can use a non-supported identity provider such as OpenID with ACS. In the next exercise you will replace the Adatum federation provider you have been using in this lab with ADFS.

## Running the "End" Solution

1. If you did not complete all of the tasks in this exercise, you can run the "end" solution we provide. Before you do you must have created a namespace instance within your own ACS service account. After you have created the namespace, you can use the following procedure to run the setup application we provide that will perform the appropriate configuration of ACS.
   1. You will need the value of the token signing certificate thumbprint to run the end solution. See the last steps of [Task 1 of Exercise 1](#Ex01Task01) in this lab for information on how to find this value.

To run the end solution

* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex03.End** from the folder **Lab03-ACS\Source\Ex03\End**.
  3. In Solution Explorer expand the **Adatum.FederationProvider.Lab03.Ex03.End** project item and double-click the file **Web.config** to open it in the Visual Studio editor.
  4. Scroll down to the **<trustedIssuers>** element and edit it to specify the thumbprint of the certificate the application will use to validate tokens received from ACS. Insert the value of the thumbprint you made a note of while performing the first task in Exercise 1 of this lab.
     1. XML
     2. <trustedIssuers>
     4. <!-- Adatum issuer certificate -->
     5. <add thumbprint="f260042d59e14817984c6183fbc6bfc71baf5462" name="adatum" />
     6. <!-- ACS issuer certificate -->
     7. <add thumbprint="**{*your-acs-thumbprint-here*}**" name="acs" />
     9. </trustedIssuers>
  5. Save and close the Web.config file.
  6. In Solution Explorer double-click the file **TrustedIssuers.cs** to open it in the Visual Studio code editor. Modify the URLs of the ACS issuer so that they contain the name of the namespace you have defined in ACS, as shown in the following highlighted code.
     1. C#
     2. public class TrustedIssuers
     3. {
     4. public static readonly IssuerInfo Adatum = new IssuerInfo("adatum",
     5. "adatum.com", "https://localhost/Adatum.SimulatedIssuer.Lab03.Ex03.End/");
     6. public static readonly IssuerInfo Acs = new IssuerInfo("acs",
     7. "https://{***your-namespace-name*}**.accesscontrol.windows.net/",
     8. "https://{***your-namespace-name*}**.accesscontrol.windows.net/v2/wsfederation");
  7. Save and close the **TrustedIssuers.cs** file.
  8. Expand the **ACS.Setup.Lab03.Ex03.End** project item and double-click the file **Program.cs** to open it in the Visual Studio editor. Edit the following two highlighted values near the start of the file to specify the password for your Azure account and the name of the namespace to use for the example.
     1. C#
     2. internal class Program
     3. {
     4. private const string AcsPassword = "{**your-acs-password}**";
     5. private const string AcsServiceNamespace = "{**your-acs-namespace}**";
     6. ...
     7. This code uses the default management account in ACS named **ManagementClient**. For help locating the correct values to use for the password key and namespace, see the section "*Locating Your ACS Password Key and Namespace*" at the start of this lab.
  9. Save and close the Program.cs file, open the **Build** menu, and click **Rebuild Solution**.
  10. Right-click on the **ACS.Setup.Lab03.Ex03.End** project, point to **Debug**, and click **Start new instance**. This will remove any existing instance of a relying party that has the name **Adatum.FederationProvider.Lab03.Ex03.End**, then add it again and configure the required rules.
  11. In Solution Explorer **right-click the a-Order.OrderTracking.Lab03.Ex03 project item and click View in Browser. You are redirected to the Adatum federation provider.**
  12. **In the Adatum federation provider page select Sign in with a Social Identity and click the Sign In button. The federation provider redirects your browser to ACS for authentication.**
  13. **As only the OpenID identity provider is configured, ACS redirects your browser to the OpenID login page.**
  14. **Enter your OpenID credentials and click Log in. After successful authentication your browser is redirected to ACS, and from there back to the a-Order application.**

# Exercise 4: Replacing the Adatum Federation Provider with ADFS

* 1. In this final exercise of Lab 1, you will replace the existing Adatum federation provider with an ADFS instance, and configure this to use ACS as a token issuer and identity provider.
  2. If you want to run the **Begin** solution and be able to sign in using a social identity before you start this exercise, you will need to make some changes to the code: in the **Adatum.FederationProvider.Lab03.Ex04** project, in the **TrustedIssuers.cs** file, you must modify the **Acs** **IssuerInfo** object to reference your ACS service namespace. In addition, you won't be able to sign in with a social identity unless you configure **Adatum.FederationProvider.Lab03.Ex04** as a relying party in your ACS namespace. See exercises 1, 2, and 3 in this lab for details of how to configure ACS.
  3. This exercise is optional, and is only possible if you have access to an ADFS 2.0 instance that you can configure for this exercise. For information about installing ADFS 2.0 for these labs and obtaining the necessary information for configuring the example, see the section "*Integrating with Active Directory Federation Services*" in the *Introduction* document for these labs.

This exercise contains the following tasks:

* + [Task 1](#_Task_1:_Modify): Modify the Order Tracking application to trust ADFS
  + [Task 2](#_Task_2:_Add): Add the Adatum claim rules to ADFS
  + [Task 3](#_Task_3:_Add): Add the Order Tracking application as a relying party in ADFS
  + [Task 4](#_Task_4:_Add): Add the Order Tracking application claim rules to ADFS
  + [Task 5](#_Task_5:_Add): Add your ACS namespace service as a claims provider trust to ADFS
  + [Task 6](#_Task_6:_Add): Add claims rules to the ACS claims provider trust in ADFS
  + [Task 7](#_Task_7:_Add): Add your ADFS instance as a relying party in your ACS service namespace
  + [Task 8](#_Task_8:_Verify): Verify the configuration changes

You should be able to complete this exercise in approximately 30 minutes.

## Task 1: Modify the Order Tracking Application to Trust ADFS

* 1. In this task, you will modify the Order Tracking application to trust your ADFS instance. You will first remove the existing Adatum simulated issuer and Adatum simulated federation provider from the solution.

To modify the Order Tracking application to trust ADFS

* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex04** from the folder **Lab03-ACS\Source\Ex04\Begin**.
  3. In Solution Explorer, right-click on the **Adatum.FederationProvider.Lab03.Ex04** project node, and select **Remove**. Then click **OK**. ADFS will take on the role of the simulated federation provider ADFS in this exercise.
  4. In Solution Explorer, right-click on the **Adatum.SimulatedIssuer.Lab03.Ex04** project node, and select **Remove**. Then click **OK**. ADFS will take on the role of the role of the simulated issuer ADFS in this exercise.
  5. In Solution Explorer expand the **a-Order.OrderTracking.Lab03.Ex04** project, and double-click on the file **Web.config** to open it in the code editor.
  6. Using the values for the name and thumbprint of your ADFS issuer that you recorded when you read the section "Integrating With Active Directory Federation Services" in the Introduction to these Lab exercises, modify the highlighted values in the **trustedIssuers** section as shown in the following snippet:
     1. XML
     2. <trustedIssuers>
     3. <add thumbprint="**{your-thumbprint}**"
     4. name="{**your-name-identifier}**" />
     5. </trustedIssuers>
  7. Using the value for the issuer address in your ADFS instance that you recorded when you read the section "Integrating With Active Directory Federation Services" in the Introduction to these Lab exercises, modify the **issuer** attribute of the **wsFederation** element as shown in the following snippet:
     1. XML
     2. <federatedAuthentication>

<wsFederation passiveRedirectEnabled="true"

issuer="**https://{issuer-url}/adfs/ls/**"

realm="https://localhost/a-Order.OrderTracking.Lab03.Ex04/"

requireHttps="true" />

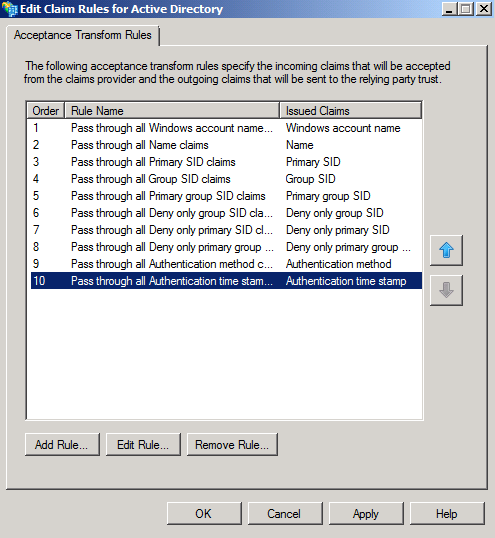
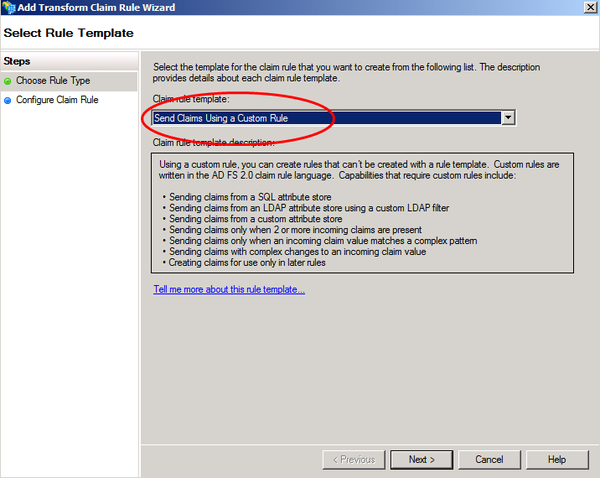
* + 1. <cookieHandler requireSsl="true" path="/a-Order.OrderTracking.Lab03.Ex04/" />
    2. </federatedAuthentication>
  1. Close all of the files open in the Visual Studio editor, making sure to save your changes, but keep the solution open so that you are ready to use it in the next task.

You have now completed this task to configure the a-Order application to trust your ADFS instance to issue claims instead of using the mock federation provider.

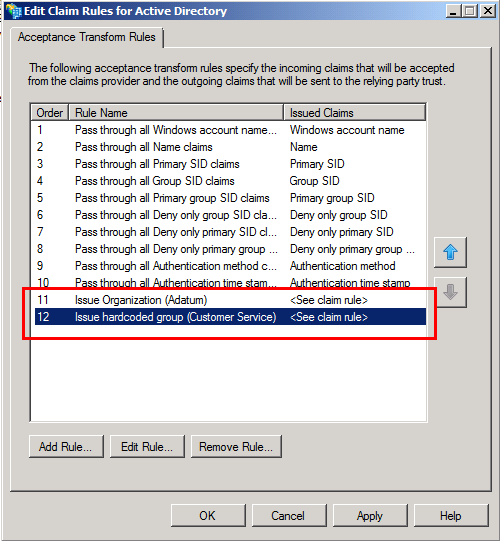
## Task 2: Add the Adatum Claim Rules to ADFS

* 1. In this task you will modify the Active Directory claims provider trust in ADFS to issue some sample claims to work with the a-Order application. You will use the AD FS 2.0 Management console to add the rules that you can use to test the a-Order application. You will also add a test user, johndoe, to your Active Directory.
  2. If you have completed either Lab 1 or Lab 2, you will have already added these rules and this user. If this is the case, you can skip this task.

To add the Adatum claims rules to ADFS

* 1. On the machine where you have installed ADFS, start the AD FS 2.0 Management tool.
  2. Expand **Trust Relationships**, then click on **Claims Provider Trusts**, then right-click on **Active Directory**, and then select **Edit Claim Rules**.
     1. 
  3. Now you can add the additional rules that you need to test the scenario. In the **Edit Claim Rules for Active Directory** dialog, click **Add Rule.** Then, in **Add Transform Claim Rule Wizard** dialog, in the **Claim rule template** dropdown list, select **Send Claims Using a Custom Rule**, and then click **Next**.
     1. 
  4. On the **Configure Rule** page of the wizard, in the **Claim rule name** textbox, type **Issue Organization (Adatum)**. In the **Custom rule** textbox, add the following rule:
     1. => issue(Type = "http://schemas.adatum.com/claims/2009/08/organization", Value = "Adatum");
  5. On the **Configure Rule** page of the wizard, click **Finish**.
  6. Using the information in the following table, repeat steps 3, 4 and 5 to add the Group rule to ADFS.

|  |  |
| --- | --- |
| Claim rule name | Custom rule |
| Issue hardcoded group (Customer Service) | => issue(Type = "http://schemas.xmlsoap.org/claims/group", Value = "Customer Service"); |

* 1. When you have finished, the **Edit Claim Rules for Active Directory** dialog should look like the following screenshot. Click **OK** to close the dialog.
     1. 
  2. On the machine where you have installed Active Directory for use in this Lab, start **Active Directory Users and Computers**.
  3. Add a new user in your domain with the attributes listed in the following table.

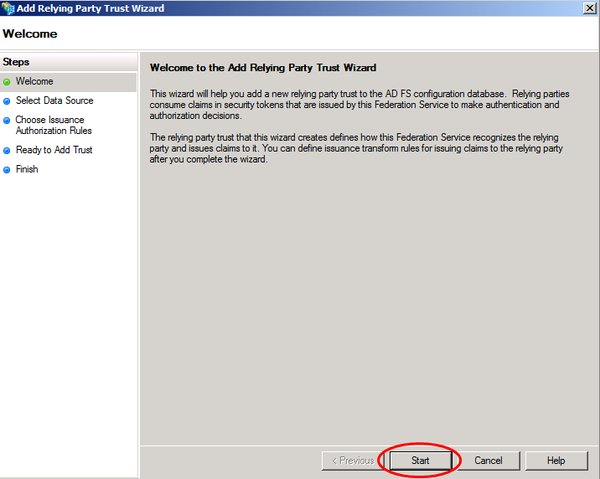
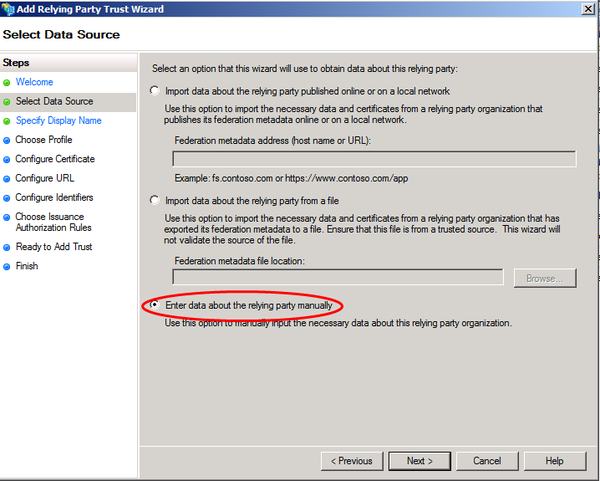
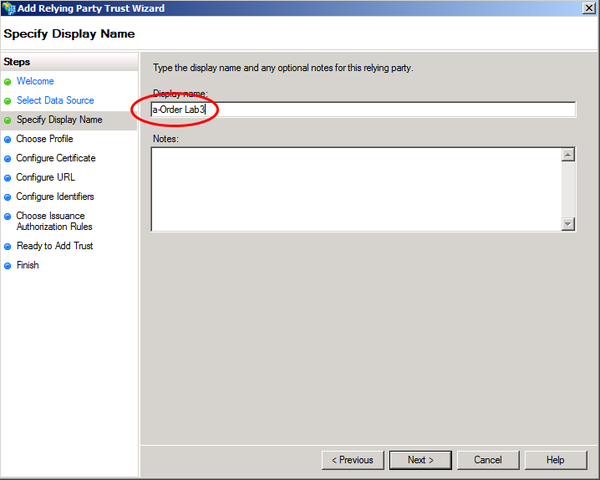
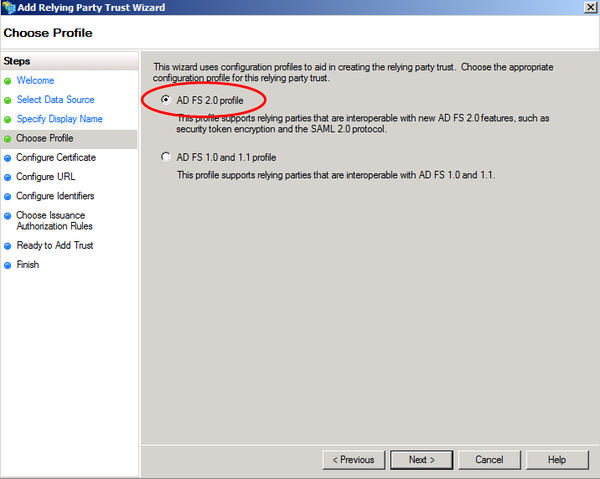
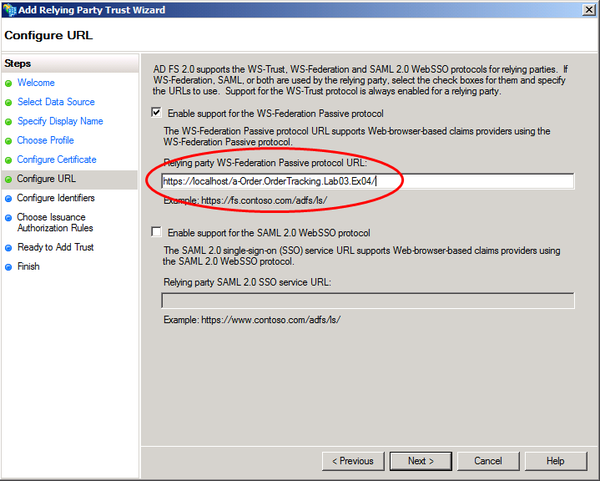
|  |  |
| --- | --- |
| Attribute | Value |
| Display name | johndoe |
| User logon name | johndoe |
| Password | Pa$$w0rd |
| User must change password at next logon | False |

You have now completed this task to add the claims rules to the Active Directory claims provider trust in ADFS that you can use to test the a-Order application.

## Task 3: Add the Order Tracking Application as a Relying Party in ADFS

* 1. In this task, you will modify the ADFS configuration to define the a-Order application as a relying party.

To add the a-Order application as a Relying Party in ADFS

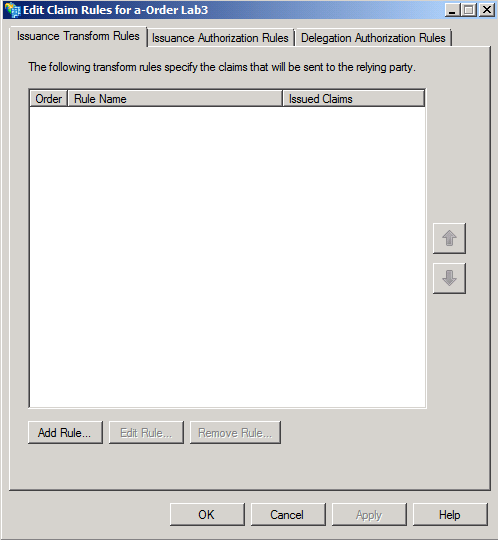
* 1. On the machine where you have installed ADFS, start the AD FS 2.0 Management tool.
  2. Expand **Trust Relationships**, then right-click on **Relying Party Trusts**, and then select **Add Relying Party Trust**.
  3. In the **Add Relying Party Trust Wizard** click **Start**.
     1. 
  4. On the **Select Data Source** page of the wizard, select **Enter data about the relying party manually** and click **Next**.
     1. 
  5. On the **Specify Display Name** page of the wizard, type **a-Order Lab3** in the **Display name** textbox, then click **Next**.
     1. 
  6. On the **Choose Profile** page of the wizard, ensure that the **AD FS 2.0 profile** option is selected, and then click **Next**.
     1. 
  7. On the **Configure Certificate** page of the wizard, click **Next**.
  8. On the **Configure URL** page of the wizard, ensure that **Enable support for the WS-Federation Passive protocol** is checked, in the **Relying party WS-Federation Passive protocol URL** text box type **https://localhost/a-Order.OrderTracking.Lab03.Ex04/**, and then click **Next**.
     1. 
  9. On the **Configure Identifiers** page of the wizard, click **Next**.
  10. On the **Choose Issuance Authorization Rules** page of the wizard, click **Next**.
  11. On the **Ready to Add Trust** page of the wizard, click **Next**.
  12. On the **Finish** page of the wizard, click **Close**. The **Edit Claim Rules for a-Order Lab2** dialog will open ready for you to complete the next task.

You have now completed this task to define the a-Order application as a relying party in ADFS.

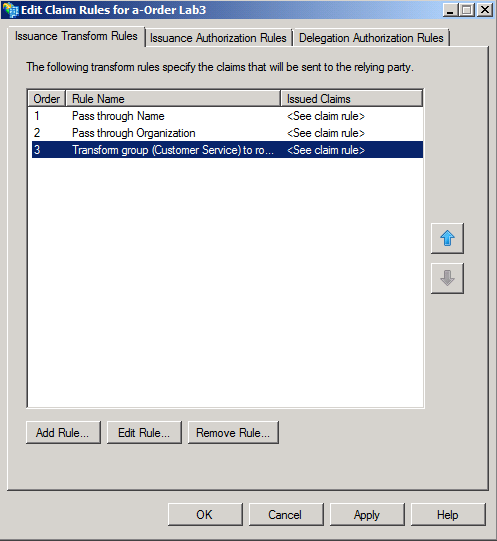
## Task 4: Add the Order Tracking Application Claim Rules to ADFS

* 1. In this task, you will add the claims transformation rules to the **a-Order Lab3** relying party trust in ADFS. These rules will map the claims from the **Active Directory** claims provider trust to a set of claims that the a-Order application will recognize.

To add the Order Tracking application claim rules to ADFS

* 1. If you do not have the **Edit Claim Rules for a-Order Lab3** dialog open from the previous task, then on the machine where you have installed ADFS, start the AD FS 2.0 Management tool. Expand **Trust Relationships**, then click on **Relying Party Trusts**, then right-click on **a-Order Lab3**, and then select **Edit Claim Rules**. You should be able to see the dialog shown in the following screenshot:
     1. 
  2. Now you can add the claim transformation rules you need to test the scenario. In the **Edit Claim Rules for a-Order Lab3** dialog, click **Add Rule.** Then, in **Add Transform Claim Rule Wizard** dialog, in the **Claim rule template** dropdown list, select **Send Claims Using a Custom Rule**, and then click **Next**.
  3. On the **Configure Rule** page of the wizard, in the **Claim rule name** textbox, type **Pass through Name**. In the **Custom rule** textbox, add the following rule:
     1. c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"] => issue(claim = c);
  4. On the **Configure Rule** page of the wizard, click **Finish**.
  5. Using the information in the following table, repeat steps 2, 3, and 4 to add the remaining rules to ADFS.

|  |  |
| --- | --- |
| Claim rule name | Custom rule |
| Pass through Organization | c:[Type == "http://schemas.adatum.com/claims/2009/08/organization"] => issue (claim = c); |
| Transform group (Customer Service) to role (Order Tracker) | c:[Type == "http://schemas.xmlsoap.org/claims/group", Value =~ "^(?i)Customer\ Service$"]  => issue(Type = "http://schemas.microsoft.com/ws/2008/06/identity/claims/role", Issuer = c.Issuer, OriginalIssuer = c.OriginalIssuer, Value = "Order Tracker", ValueType = c.ValueType); |

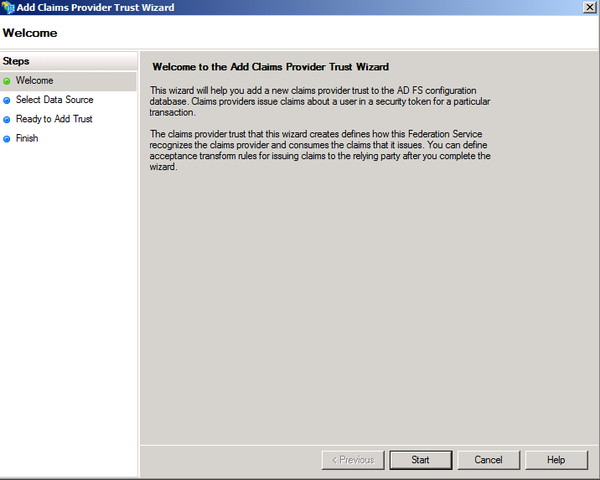
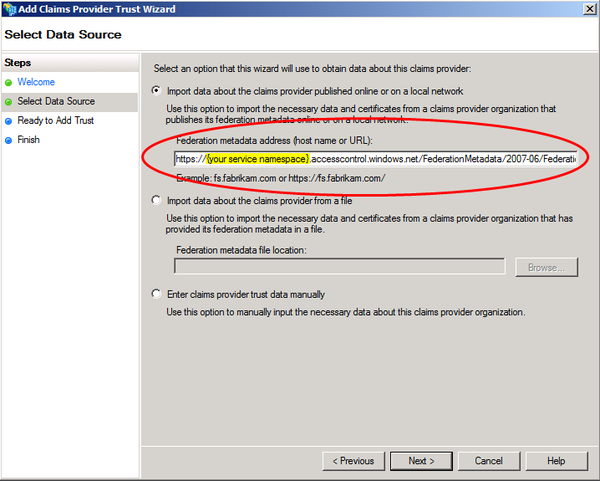
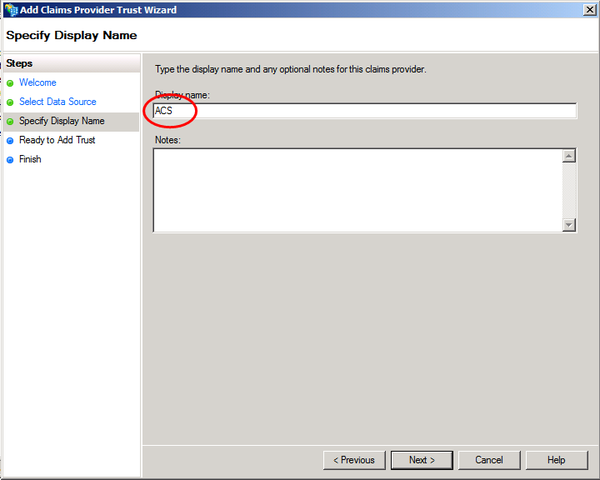
* 1. When you have finished, the **Edit Claim Rules for a-Order Lab3** dialog should look like the following screenshot. Click **OK** to close the dialog.
     1. 

You have now completed this task and added a set of claim transformation rules for the a-Order relying party application in ADFS.

## Task 5: Add your ACS Namespace Service as a Claims Provider Trust to ADFS

* 1. In this task, you will add a claims provider trust for ACS to the ADFS configuration. This will configure ADFS to trust claims issued by your ACS service namespace.

To add your ACS namespace service as a claims provider trust to ADFS

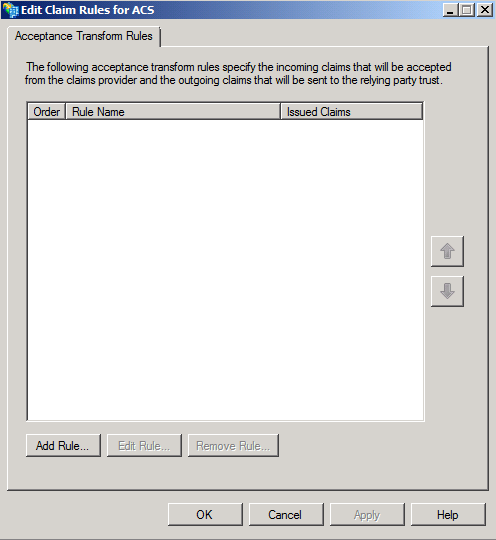
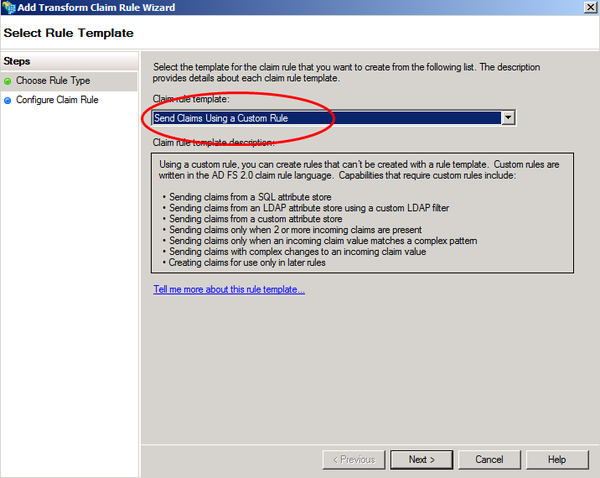
* 1. On the machine where you have installed ADFS, start the **AD FS 2.0 Management** console. Expand **Trust Relationships**, then right-click on **Claims Provider Trusts**, and then select **Add Claims Provider Trust**. The **Add Claims Provider Trust Wizard** will launch. Click **Start**.
     1. 
  2. On the **Select Data Source** page of the wizard, click **Import data about the claims provider published online or on a local network**, and enter your ACS namespace service metadata URL: **https://{your service namespace}.accesscontrol.windows.net/FederationMetadata/2007-06/FederationMetadata.xml**. Then click **Next**.
     1. You can use the ACS service namespace that you created in exercise 1 of this lab.
     2. 
  3. On the **Specify Display Name** page, in **Display name** type **ACS**. Then click **Next**.
     1. 
  4. On the **Ready to Add Trust** page, click **Next** to save your claims provider trust information.
  5. On the **Finish** page, click **Close**. This action automatically displays the **Edit Claim Rules** dialog box that you will use in the next task.

You have now completed this task to configure your ACS service namespace as a claims provider in ADFS.

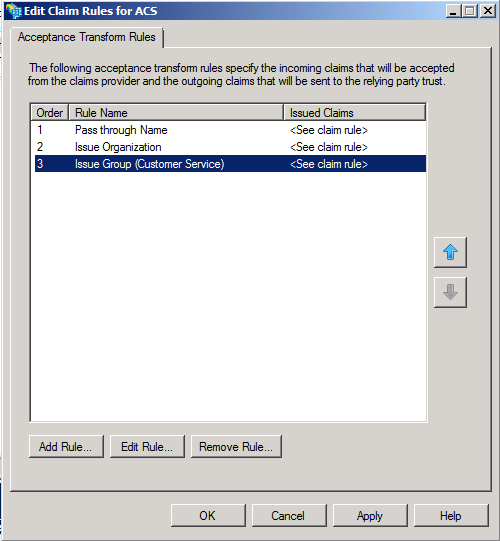
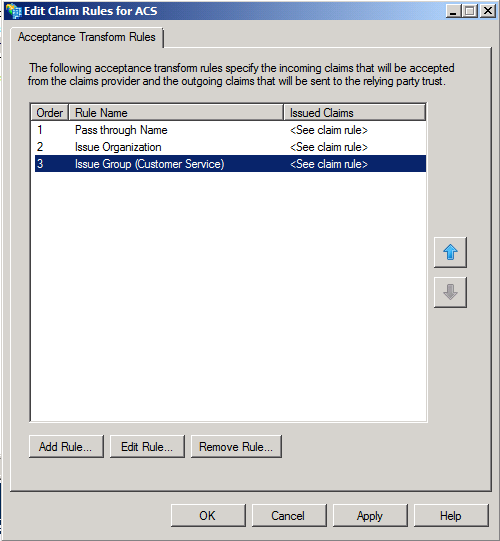
## Task 6: Add Claims Rules to the ACS Claims Provider Trust in ADFS

* 1. In this task, you will modify the ACS claims provider trust in ADFS to accept claims from ACS. You will use the AD FS 2.0 Management console to add the rules that you can use to test the a-Order application.

To Add claims rules to the ACS claims provider trust in ADFS

* 1. If you do not have the **Edit Claim Rules for ACS** dialog open from the previous task, then on the machine where you have installed ADFS, start the AD FS 2.0 Management console. Expand **Trust Relationships**, then click on **Claims Provider Trusts**, then right-click on **ACS**, and then select **Edit Claim Rules**. You should be able to see the dialog shown in the following screenshot.
     1. 
  2. Now you can add the rules you need to test the scenario. In the **Edit Claim Rules for ACS** dialog, click **Add Rule.** Then, in **Add Transform Claim Rule Wizard** dialog, in the **Claim rule template** dropdown list, select **Send Claims Using a Custom Rule**, and then click **Next**.
     1. 
  3. On the **Configure Rule** page of the wizard, in the **Claim rule name** textbox, type **Pass through Name**. In the **Custom rule** textbox, add the following rule:
     1. c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"] => issue (claim = c);
  4. On the **Configure Rule** page of the wizard, click **Finish**.
  5. Using the information in the following table, repeat steps 2, 3 and 4 to add the remaining rules to ADFS.

|  |  |
| --- | --- |
| Claim rule name | Custom rule |
| Issue Organization | => issue(Type = "http://schemas.adatum.com/claims/2009/08/organization", Value = "MaryInc"); |
| Issue Group (Customer Service) | => issue(Type = "http://schemas.xmlsoap.org/claims/group", Value = "Customer Service"); |

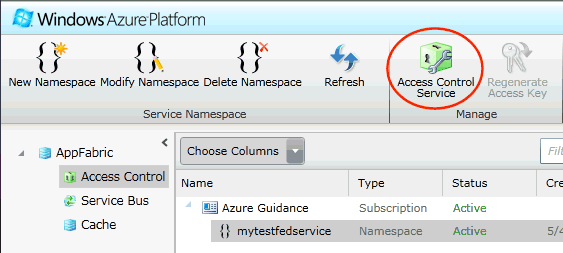
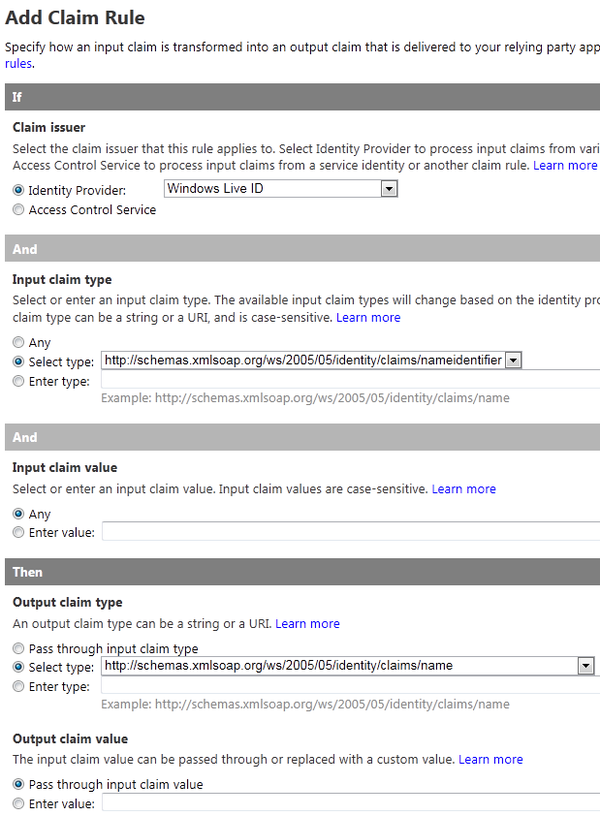
* 1. When you have finished, the **Edit Claim Rules for ACS** dialog should look like the following screenshot. Click **OK** to close the dialog.
     1. 

You have now completed this task to add the ACS claims rules to the ACS claims provider trust in ADFS that you can use to test the a-Order application.

## Task 7: Add your ADFS Instance as a Relying Party in your ACS Service Namespace

* 1. In this task, you will add your ADFS instance as a relying party in your ACS service namespace. This will enable your ACS service namespace to issue claims to your ADFS instance.

To add your ADFS instance as a relying party in your ACS service namespace

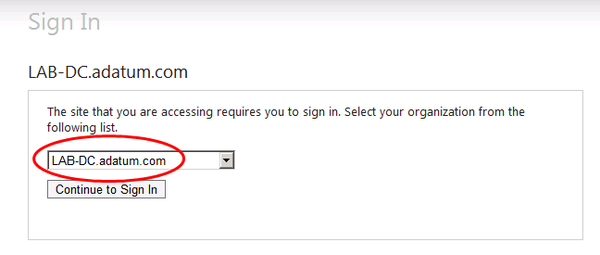
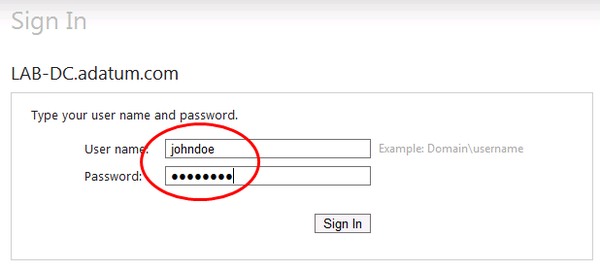
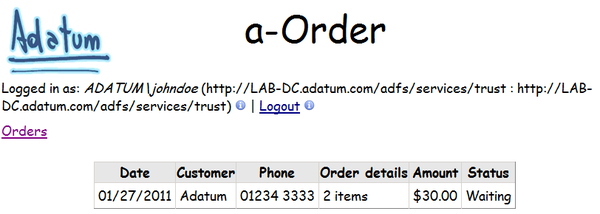
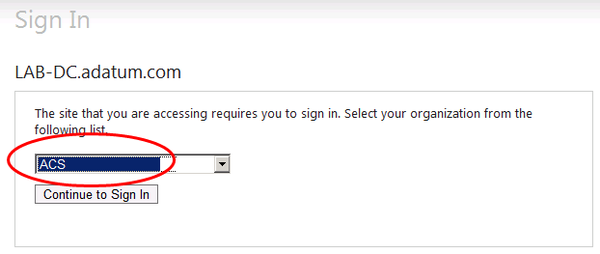
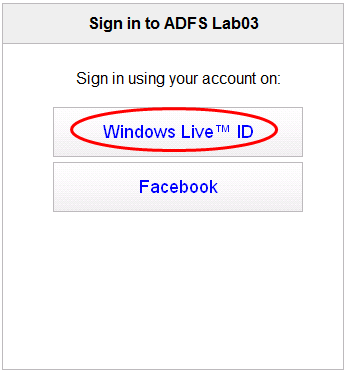
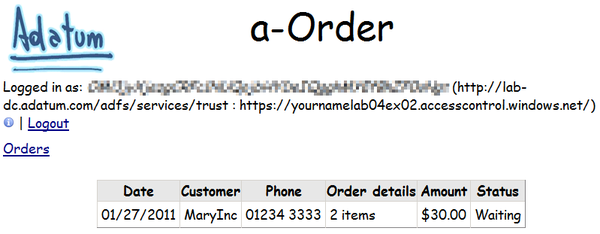
* 1. Go to the Windows Azure Management Portal at <https://windows.azure.com/> and sign into your account. Then select **Service Bus, Access Control & Caching** in the lower section of the left pane and click **Access Control** in the list of services in the **AppFabric** tree.
  2. Select the namespace you created in the previous exercise and click the **Access Control Service** icon at the top of the main window to go to the Access Control administration pages.
     1. 
  3. In the Access Control Service page, click **Relying party applications** in the left pane or in the Getting Started section of the main page.
  4. In the Relying Party Applications page click the **Add** link and enter the following values in the Add Relying Party Application page:
     + Name: **ADFS Lab03**
     + Mode: **Enter settings manually**
     + Realm: **http:// {Your ADFS hostname}/adfs/services/trust**
     + Return URL: **https:// {Your ADFS hostname}/adfs/ls/**
  5. Make sure that you select **Facebook** and **Windows LiveID** as **Identity providers**.
  6. Leave the remainder of the settings at their default values and click **Save** to create the new relying party entry. You will see it in the list on the Relying Party Applications page.
     1. Adding a relying party with the default settings automatically adds the Windows Live ID identity provider to the configuration and creates a new default rule group.
  7. In the Access Control Service page, click **Rule Groups** in the left pane or in the Getting Started section of the main page.
  8. In the Rule Groups page, click the **Default Rule Group for ADFS Lab03** link, then click the **Generate** link. On the Generate Rules page, make sure that you select **Facebook** and **Windows LiveID** in the **Rule Generation Options** section, and then click **Generate**.
  9. You will see the generated rules on the **Edit Rule Group page**: these rules include passthrough rules for the Facebook and Windows Live ID issuers. Click the **Add** link to add additional rules.
  10. The a-Order application requires a claim that is the name of the authenticated user. Windows Live ID does not disclose the user name, so for this exercise you will use the **nameidentifier** value to populate the **name** claim. Select or enter the following values in the Add Claim Rule page:
      + Identity Provider: **Windows Live ID**
      + Input claim type: **Select Type** and then select the claim type **http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier**
      + Input claim value: **Any**
      + Output claim type: **Select Type** and then select the claim type **http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name**
      + Output claim value: **Pass though the input claim value**
      + Description: **Pass "nameidentifier" value as "name" claim.**
      1. 
  11. Click **Save**. The new rule appears in the list in the Edit Rule Group page. Click the **Save** button above the list to update the rule group with the new rules.
      1. If you want to test the solution using your OpenID provider, you should add a rule to pass through a name claim from the issuer. See exercise 3 in this lab for an example.

You have now completed this task. In this task, you have seen how to configure your ADFS instance as a relying party in ACS.

## Task 8: Verify the Configuration Changes

* 1. In this task, you will verify the configuration changes you have made: you will sign-in to the a-Order application as **johndoe** from Adatum and using a Windows Live ID. The a-Order application is now configured to trust ADFS as its federation provider.
  2. ADFS uses a cookie to record your preferred identity provider: to avoid being "locked" to one identity provider during testing you should use your browser's **InPrivate** browsing feature that ignores all persistent cookies. In Internet Explorer 9, you can activate **InPrivate** browsing by clicking the **Tools** icon, then pointing to **Safety**, and then click **InPrivate Browsing**. In Internet Explorer 8, click **Safety** on the toolbar, and then click **InPrivate Browsing**.
  3. If you are running this Lab on the same machine as the one where you have installed ADFS, then you will receive a warning message "The security certificate presented by this website was issued for a different website's address" when your browser is redirected to ADFS. You should click **Continue to this website** to continue.

To verify the configuration changes

* 1. Open your web browser, activate **InPrivate** browsing, and navigate to the URL https://localhost/a-Order.OrderTracking.Lab03.Ex04/ to run the claims-aware a-Order application. Because you have not authenticated, you will be immediately redirected to ADFS.
  2. If you see a **Certificate Error: Navigation Blocked** page, click **Continue to this website (not recommended)**. You will see the ADFS home realm discovery page. From the entries in the drop-down listbox, select the one that matches the name of your Active Directory domain (it may not match exactly the name shown in the following screenshot.) Then click **Continue to Sign In**.
     1. 
  3. On the Sign In page enter your domain name and **johndoe** as the user name, and **Pa$$w0rd** as the password. Then click **Sign In**.
     1. 
     2. If you don't see the ADFS sign in page, you should check your ADFS configuration. See the section "*Integrating with Active Directory Federation Services*" in the *Introduction* document for these labs for details.
  4. You will see the a-Order page with a list of Adatum items and your login details.
     1. 
  5. Close your **InPrivate** browser window, then open a new **InPrivate** browsing window, and navigate to the URL <https://localhost/a-Order.OrderTracking.Lab03.Ex04/> to run the claims-aware a-Order application. Because you have not authenticated, you will be immediately redirected to ADFS.
  6. If you see a **Certificate Error: Navigation Blocked** page, click **Continue to this website (not recommended)**. You will see the ADFS home realm discovery page. From the entries in the drop-down listbox, select **ACS.** Then click **Continue to Sign In**.
     1. 
     2. If you are not using **InPrivate** browsing you will be redirected immediately to the Adatum sign-in page, by-passing this page. This is because ADFS has saved your preferred identity provider in a cookie. To see the home realm discovery page you will need to clear your browser's cookie cache.
  7. On the ACS home realm discovery page, click **Windows LiveTM ID** to log in with your Windows Live ID.
     1. 
  8. On the **Welcome to Windows Live** page, enter the Windows Live credentials that you are using to test this scenario, and then click **Sign in**.
  9. You will see the a-Order page with a list of MaryInc items and your login details. You will see your Windows Live ID nameidentifier displayed as the login id.
     1. 

You have now completed this task to verify that ADFS functions as a federation provider for the a-Order application.

You can use ADFS as part of your federated identity infrastructure. Adatum users can sign-in with their AD credentials, and users with social identities such as Windows Live IDs and Facebook IDs can sign with their social identity. Both sets of users get access to their data in the a-Order application.

* 1. You have now completed this task and this final exercise. In this exercise, you have seen how you can replace the simulated federation provider from the previous exercises with ADFS to configure a more "real-world" scenario. ADFS functions as a federation provider and enables the user to select between identity providers (in this case either Active Directory or ACS). ADFS also transforms the tokens it receives from Active Directory and ACS into tokens that the a-Order application understands.

## Running the "End" Solution

1. If you did not complete all of the tasks in this exercise, you can run the provided "end" solution. Before you do, you must have created a namespace instance within your own ACS service account. After you have created the namespace, you can use the following procedure to run the setup application in the "end" solution that will perform the appropriate configuration of ACS.
   1. You will need the value of the token signing certificate thumbprint to run the end solution. See the last steps of [Task 1 of Exercise 1](#Ex01Task01) in this lab for information on how to find this value.

To run the end solution

* 1. Start Visual Studio 2010 as an administrator.
  2. Open the solution named **Lab03.Ex04.End** from the folder **Lab03-ACS\Source\Ex04\End**.
  3. In the **Lab03.Ex04.End** solution, you complete steps 5 to 8 from task 1 in this exercise to configure the a-Order application to trust your ADFS instance.
  4. You must add the required Adatum claim rules to your ADFS instance. To do this, you should run the PowerShell script, **Add-Adatum-Rules.ps1** in the Lab03-ACS\Source\Ex04\Assets folder.
     1. Before your run this script, you should check whether you added these rules in a previous exercise — if the claim rules for Active Directory already includes rules named **Issue Organization (Adatum)** and **Issue hardcoded group (Customer Service)** then you have the rules that you need to run this scenario.
  5. You must configure the a-Order application as a relying party in ADFS. To do this, you should run the PowerShell script, **Add-aOrder-RelyingParty.ps1** in the Lab03-ACS\Source\Ex04\Assets folder.
  6. You must add the claims transformation rules to the a-Order relying party in ADFS. To do this, you should run the PowerShell script, **Add-aOrder-Rules.ps1** in the Lab03-ACS\Source\Ex04\Assets folder.
  7. You must add ACS as a claims provider trust to ADFS. To do this, you should first edit the PowerShell script, **Add-ACS-ClaimsProviderTrust.ps1** in the Lab03-ACS\Source\Ex04\Assets folder to replace the value {your-namespace-name} with your ACS namespace service name, then run the script.
  8. You must add the claims transformation rules to the ACS claim provider trust in ADFS. To do this, you should run the PowerShell script, **Add-ACS-Rules.ps1** in the Lab03-ACS\Source\Ex04\Assets folder.
  9. You must configure your ADFS instance as a relying party in your ACS service namespace. In Visual Studio, in the solution named **Lab03.Ex04.End**, expand the **ACS.Setup.Lab03.Ex04.End** project item and double-click the file **Program.cs** to open it in the Visual Studio editor. Edit the following three highlighted values near the start of the file to specify the ACS management password key for your Azure account, the name of the ACS service namespace to use for the example, and the hostname of your ADFS server.
     1. C#
     2. internal class Program
     3. {
     4. private const string AcsPassword = "**{your-acs-password}**";
     5. private const string AcsServiceNamespace = "{**your-namespace-name}**";
     6. private const string AcsUsername = "ManagementClient";
     7. private const string AdfsHostName = "**{issuer-url}**";
     8. ...
     9. This code uses the default management account in ACS named **ManagementClient**. For help locating the correct values to use for the password key and namespace, see the section "*Locating Your ACS Password Key and Namespace*" at the start of this lab.
  10. Save and close the Program.cs file, open the **Build** menu, and click **Rebuild Solution**.
  11. Right-click on the **ACS.Setup.Lab03.Ex04.End** project, point to **Debug**, and click **Start new instance**. This will remove any existing instance of a relying party that has the name **ADFS Lab03**, then add it again and configure the required rules.
  12. Complete task 8 in this exercise to verify the configuration changes. You should use the **https://localhost/a-Order.OrderTracking.Lab03.Ex04.End/** URL instead of **https://localhost/a-Order.OrderTracking.Lab03.Ex04/**.