WS-Trust STS and Microsoft Integration Guide

Contents

WS-Trust STS and Microsoft Integration Guide

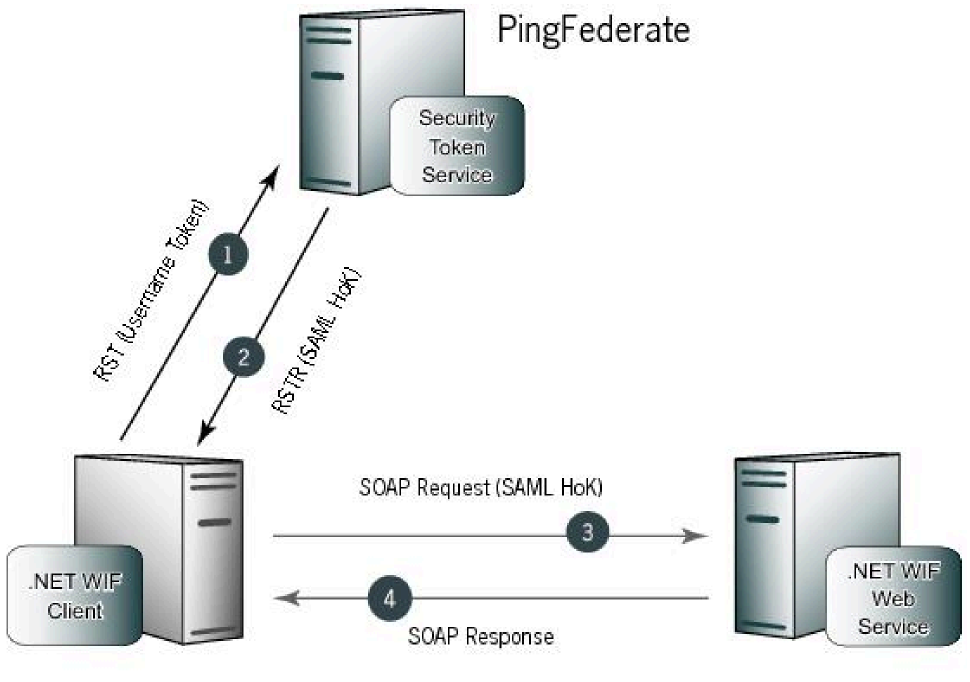
The Microsoft suite of federation technologies provides several integration points with the PingFederate WS-Trust Security Token Service (STS). This document addresses common scenarios encountered when using the Windows Identity Foundation (WIF) and Windows Communication Foundation (WCF). Currently, the PingFederate STS fulfills active use cases and may act as either an Identity Provider STS (IP-STS) or Relying Party STS (RP-STS). (For the purpose of this document, Microsoft Web Services federation terminology is used.)

Overview

The Microsoft suite of federation technologies provides several integration points with the PingFederate WS-Trust Security Token Service (STS). This document addresses common scenarios encountered when using the Windows Identity Foundation (WIF) and Windows Communication Foundation (WCF). Currently, the PingFederate STS fulfills active use cases and may act as either an Identity Provider STS (IP-STS) or Relying Party STS (RP-STS). (For the purpose of this document, Microsoft Web Services federation terminology is used.)

Note: This document provides a sample configuration and applications for the IP-STS use case.

Increasingly, Ping Identity customers are deploying PingFederate in conjunction with Microsoft federation technologies. To ensure successful deployment of these technologies, Ping Identity is providing this document to demonstrate the most common interoperability scenario (IP-STS) via configuration examples and sample applications. Most scenarios resemble the diagram below but are becoming more complex as federation technologies mature.



Sequence

1. The .NET WIF client sends a request-security-token (RST) message to the PingFederate STS containing a WS-Security Username Token.

2. PingFederate STS validates the WS-Security Username Token using a token processor instance and issues a SAML token with holder-of-key (HoK) confirmation method contained in a request-security-token-response (RSTR) message.
3. The .NET WIF client sends a SOAP request to the .NET WIF Web Service with the issued SAML token to be used for authentication by the service.
4. The .NET WIF Web Service validates the issued SAML token and fulfills the Web Service request.

Preparation

These sections provide detailed requirements for PingFederate and Windows components needed for integration, as well as information needed to run the sample applications.

PingFederate server prerequisites

The samples used in this document require a PingFederate server with additional components and configuration applied, as listed below.

ⓘ **Note:** Some of these settings will render the PingFederate server temporarily unreachable for other purposes and thus should not be carried out on a production or other in-use instance of the server. Be sure to restore previous settings after completing this demonstration, or create a configuration archive in advance if needed to restore all settings.

- PingFederate 6.5-PREVIEW or higher
- Username Token Translator version 1.1 or higher

  ⓘ **Note:** For PingFederate 7.2 or higher, Username Token Processor is part of the product and does not require a separate download or installation.
- Imported Key Pair and Certificate for digital signature and digital signature verification.
- Use of an SSL Certificate for the PingFederate runtime port that is trusted by the Microsoft Windows Workstation. For purposes of this document, DN of the SSL Certificate should be `CN=pfsts.contoso.com`.
- PingFederate must be configured on **Server Configuration** > **Server Settings** > **Federation Info** screen with a Base URL of `http://pfsts.contoso.com:9031`, which matches the DN of the SSL certificate for the purpose of this document.

Microsoft Windows Workstation prerequisites

The samples used in this document require a Microsoft Windows workstation with the following software installed and configured.

- Microsoft Windows 7 Professional or higher
- Microsoft Visual Studio 2010 Professional or higher
- Microsoft Windows Identity Foundation runtime (release 3/22/2010 KB974405)
- Microsoft Windows Identity Foundation SDK (release 12/15/2010 for Microsoft .NET Framework 4.0)
- Internet Information Services (IIS) 6.1 or higher with DefaultAppPool available
  - Dynamic XSD generation by the WCF Service Application requires write access to `%SYSTEMROOT%\Temp`. Grant sufficient privileges to this directory to the `<IIS AppPool\DefaultAppPool>` IIS user.

- Create a record for the PingFederate server in `%SYSTEMROOT%\System32\drivers\etc\hosts`. For the purpose of this document, designate the PingFederate server hostname as `pfsts.contoso.com`.

Sample Application Prerequisites

About this task

A self-signed key pair with DN CN=localhost is required to use the sample application.

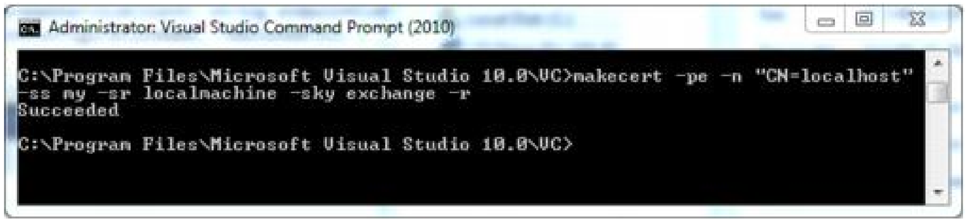
> **Note:** If such a key pair already exists on the Microsoft Windows development workstation, skip this section.

To create the key pair:

Steps

1. Use the following command from the Visual Studio 2010 Command Prompt.

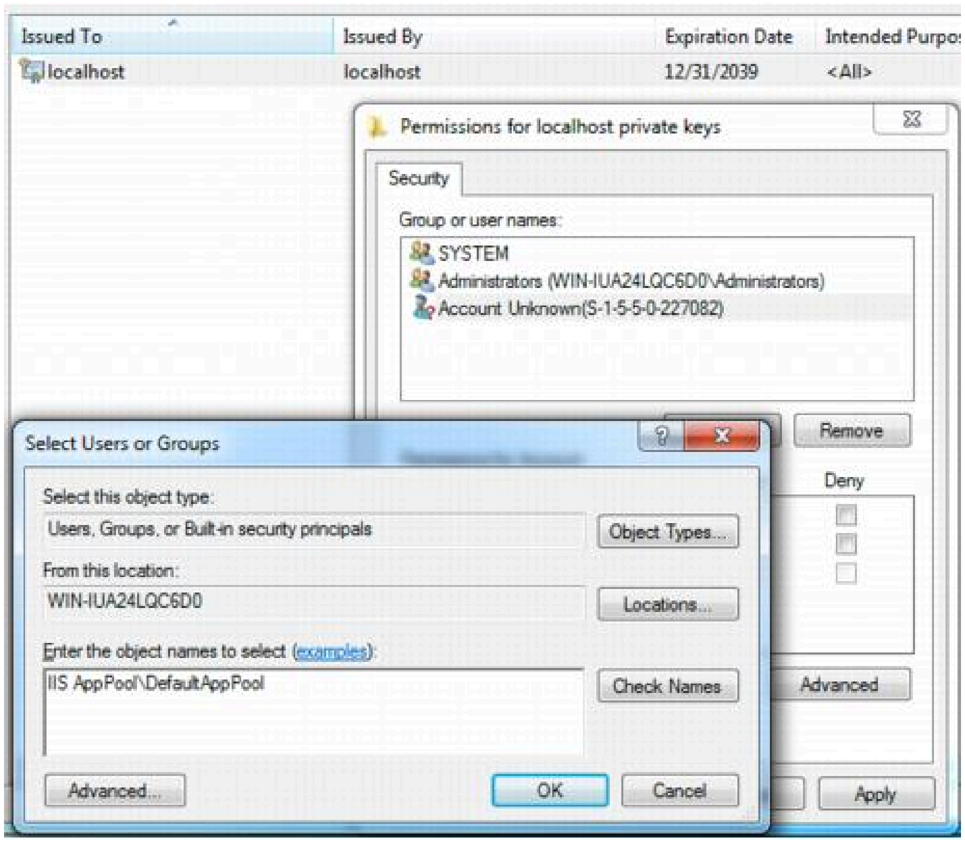
```
makecert -pe -n "CN=localhost" -ss my -sr localmachine -sky exchange -r
```



```
C:\Program Files\Microsoft Visual Studio 10.0\VC>makecert -pe -n "CN=localhost"
-ss my -sr localmachine -sky exchange -r
Succeeded

C:\Program Files\Microsoft Visual Studio 10.0\VC>
```

> **Note:** The `makecert.exe` example imports the created certificate into the LocalMachine\Personal certificate store. Since this is a privileged operation, run the Visual Studio 2010 Command Prompt as Administrator and disable User Account Control (UAC) to prevent interference with privileged operations run from a command line.

2. Grant Full Control of the `CN=localhost` private key to the `<IIS AppPool\DefaultAppPool>` via the **Certificates Snap-In for Microsoft Management Console** (MMC).



Issued To	Issued By	Expiration Date	Intended Purpo
localhost	localhost	12/31/2039	<All>

Permissions for localhost private keys

Security

Group or user names:

SYSTEM
Administrators (WIN-IUA24LQC6D0\Administrators)
Account Unknown(S-1-5-5-0-227082)

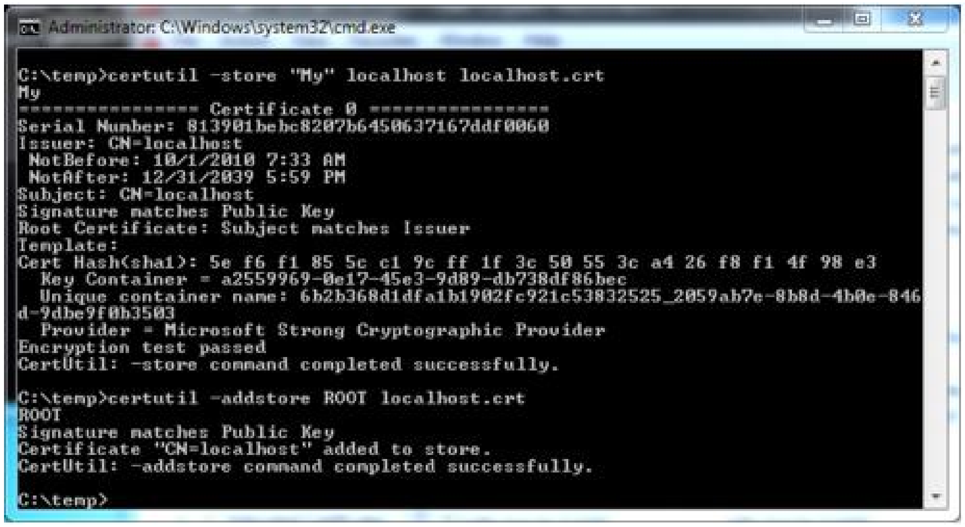
Select Users or Groups

Select this object type:
Users, Groups, or Built-in security principals

From this location:
WIN-IUA24LQC6D0

Enter the object names to select (examples):
IIS AppPool\DefaultAppPool

3. Import the public certificate into the **Trusted Root Certification Authorities**. The following command window shows both commands:



Administrator: C:\Windows\system32\cmd.exe

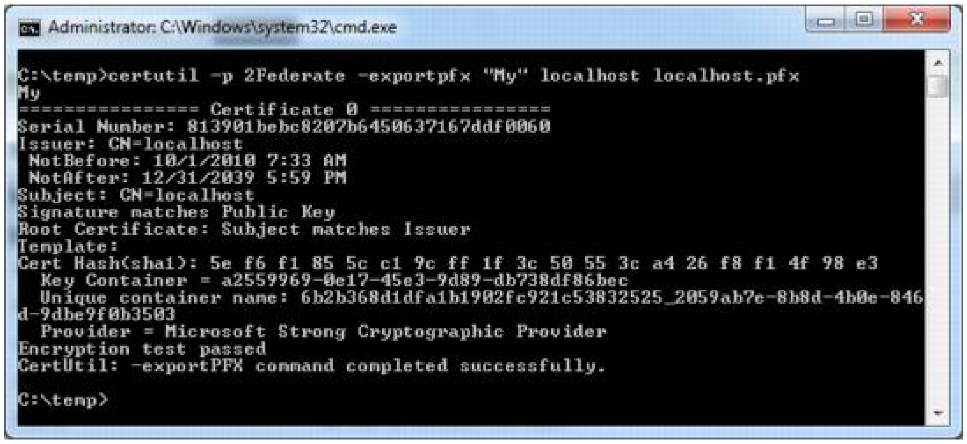
```
C:\temp>certutil -store "My" localhost localhost.crt
My
================ Certificate 0 ================
Serial Number: 813901bebc8207b6450637167ddf0060
Issuer: CN=localhost
 NotBefore: 10/1/2010 7:33 AM
 NotAfter: 12/31/2039 5:59 PM
Subject: CN=localhost
Signature matches Public Key
Root Certificate: Subject matches Issuer
Template:
Cert Hash(sha1): 5e f6 f1 85 5c c1 9c ff 1f 3c 50 55 3c a4 26 f8 f1 4f 98 e3
  Key Container = a2559969-0e17-45e3-9d89-db738df86bec
  Unique container name: 6b2b368d1dfa1b1902fc921c53832525_2059ab7e-8b8d-4b0e-846
d-9dbe9f0b3503
  Provider = Microsoft Strong Cryptographic Provider
Encryption test passed
CertUtil: -store command completed successfully.

C:\temp>certutil -addstore ROOT localhost.crt
ROOT
Signature matches Public Key
Certificate "CN=localhost" added to store.
CertUtil: -addstore command completed successfully.

C:\temp>
```

- The first command, `fertutil -store`, displays the key pair details and saves the public certificate to file `localhost.crt` in the current directory.

  Note: Ensure that the file is accessible to the PingFederate server for importing later.

- The second command, `certutil -addstore`, imports the certificate into the necessary store to create a valid certificate path for the newly created self-signed key pair. The public certificate is also used later when configuring a connection via the PingFederate administrative console.

4. Use `certutil` to export the public/private key pair to a file named `localhost.pfx` for later import into PingFederate, as shown here:



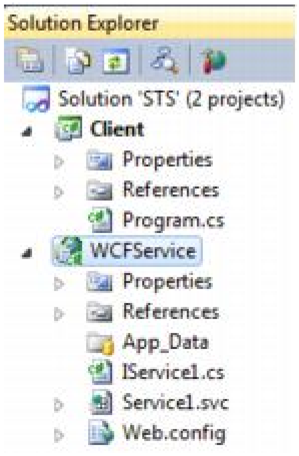
```
C:\temp>certutil -p 2Federate -exportpfx "My" localhost localhost.pfx
My
================ Certificate 0 ================
Serial Number: 813901bebc8207b6450637167ddf0060
Issuer: CN=localhost
 NotBefore: 10/1/2010 7:33 AM
 NotAfter: 12/31/2039 5:59 PM
Subject: CN=localhost
Signature matches Public Key
Root Certificate: Subject matches Issuer
Template:
Cert Hash(sha1): 5e f6 f1 85 5c c1 9c ff 1f 3c 50 55 3c a4 26 f8 f1 4f 98 e3
  Key Container = a2559969-0e17-45e3-9d89-db738df86bec
  Unique container name: 6b2b368d1dfa1b1902fc921c53832525_2059ab7e-8b8d-4b0e-846
d-9dbe9f0b3503
  Provider = Microsoft Strong Cryptographic Provider
Encryption test passed
CertUtil: -exportPFX command completed successfully.

C:\temp>
```

Using PingFederate as an IP-STS via metadata

Follow these steps to create a WCF Web Service client that uses PingFederate as an IP STS.

Create a Visual Studio project

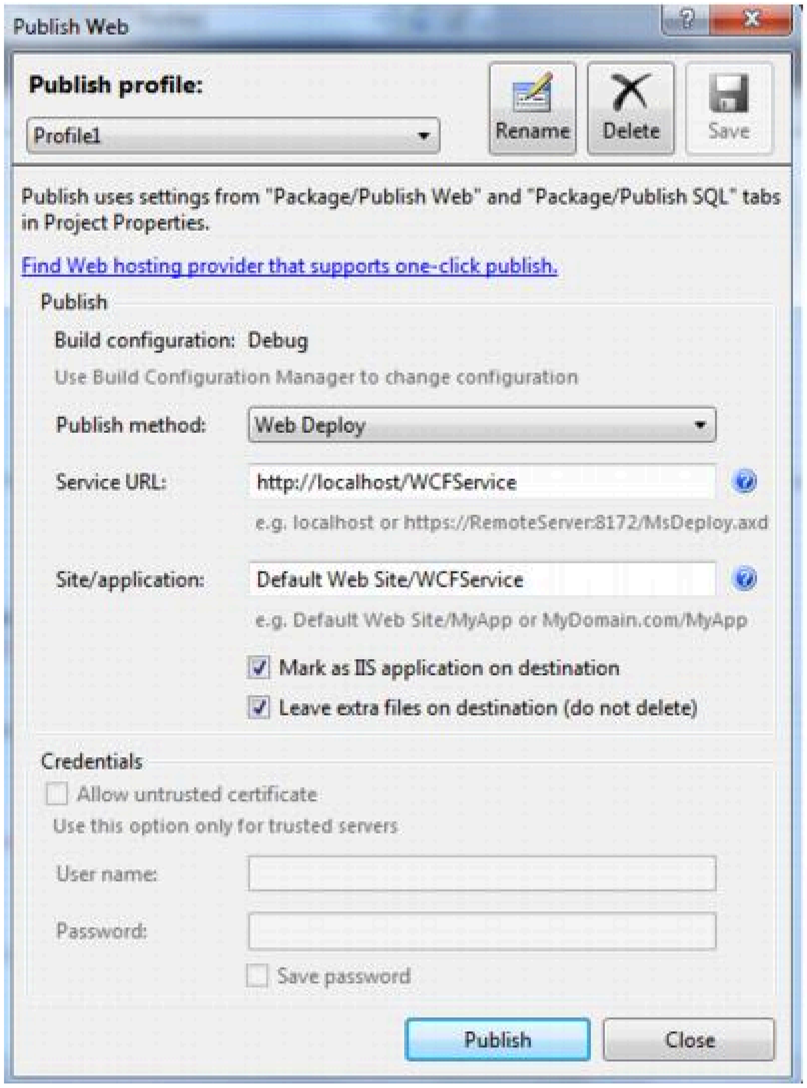
This exercise consists of creating a Visual C# WCF Service Application and a Visual C# client Console Application with STS configuration conducted via metadata available from PingFederate. Begin by creating a new Solution via Visual Studio with the two application projects (see below). Ensure the client Console Application is set as the “StartUp Project”.



Publish the WCF service application and verify deployment

Publish the WCF Service Application to the local IIS instance and ensure the service is available by testing the availability of the Web application via a Web browser. Publishing can be accomplished in a variety of ways; Web Deploy is used for the example shown in Figure 1.

Simply retrieving the service URL in a browser will render a page indicating a successful deployment, as shown in Figure 2.



Publish profile: Profile1

Publish uses settings from "Package/Publish Web" and "Package/Publish SQL" tabs in Project Properties.

Find Web hosting provider that supports one-click publish.

Publish

Build configuration: Debug

Use Build Configuration Manager to change configuration

Publish method: Web Deploy

Service URL: http://localhost/WCFService

e.g. localhost or https://RemoteServer:8172/MsDeploy.axd

Site/application: Default Web Site/WCFService

e.g. Default Web Site/MyApp or MyDomain.com/MyApp

[✓] Mark as IIS application on destination

[✓] Leave extra files on destination (do not delete)

Credentials

[ ] Allow untrusted certificate

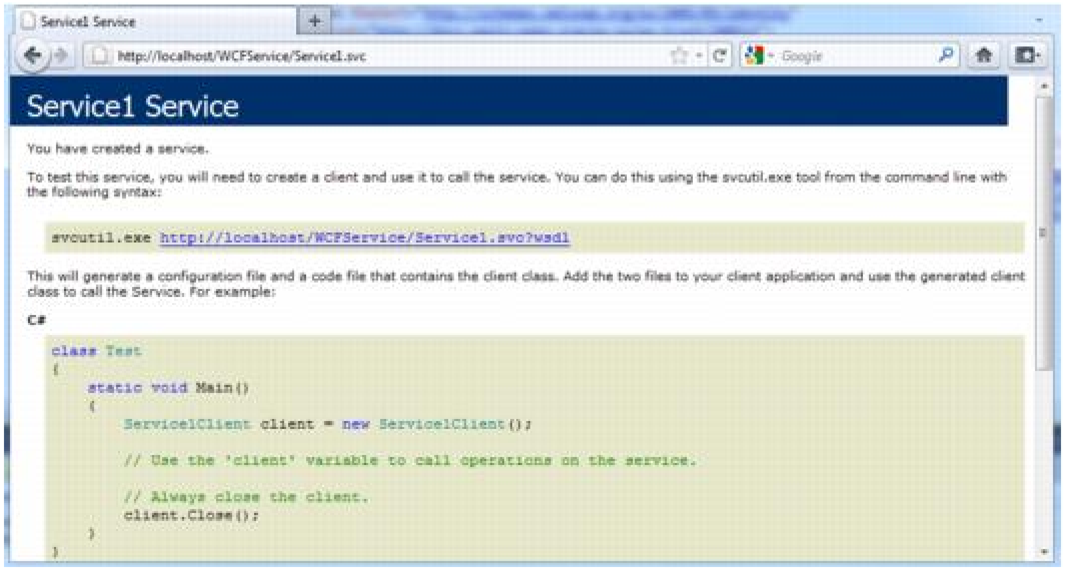
Use this option only for trusted servers

User name:

Password:

[ ] Save password

MY TITLE Publishing the WCF Service Application



Service1 Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost/WCFService/Service1.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

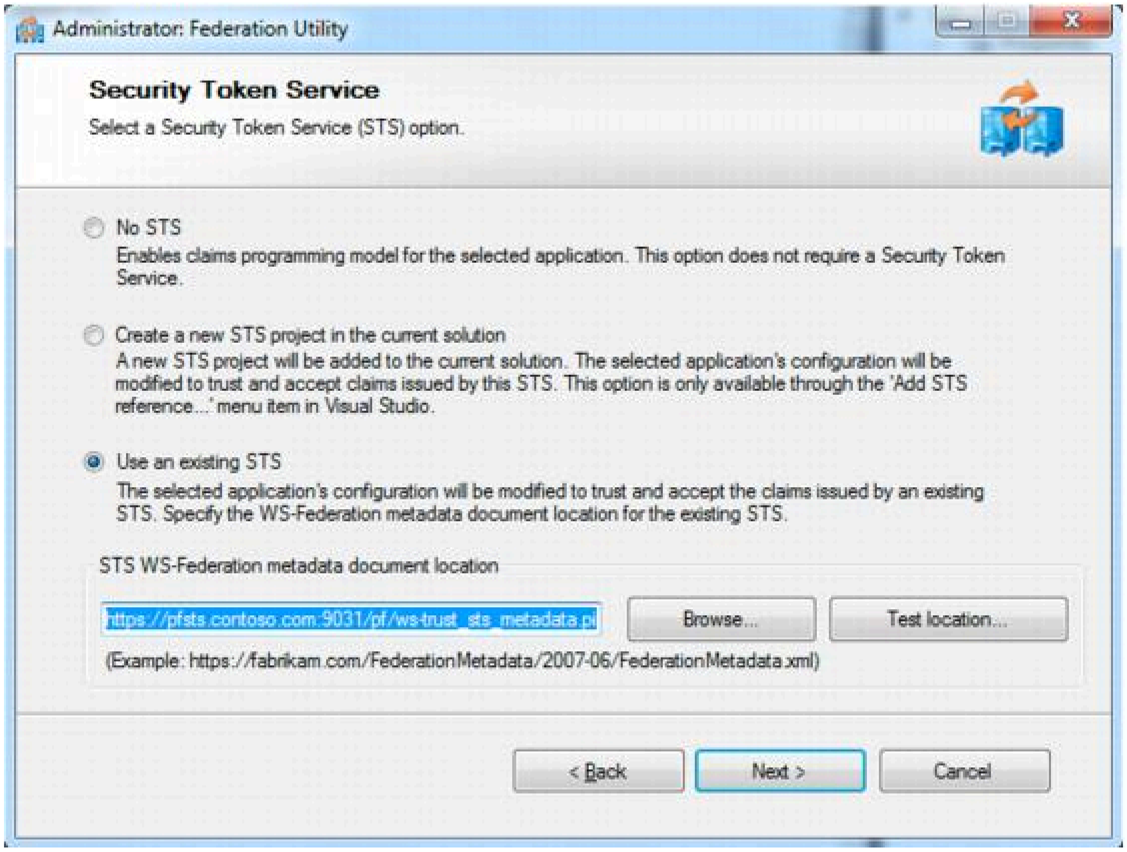
```
class Test
{
    static void Main()
    {
        Service1Client client = new Service1Client();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

MY TITLE WCF Service Application Information Page

Add an STS reference to the WCF service application

Add an STS reference to the WCF Service Application. Leave the default value for the Application configuration location. For the application URI use `http://localhost/WCFService`. Select the radio button to specify the use of an existing STS as shown in Figure 1. For the STS WS-Federation metadata document location use the following.

```
https://pfsts.contoso.com:9031/pf/ws-trust_sts_metadata.ping?
   PartnerSpId=http://localhost/WCFService
```


Administrator: Federation Utility

Security Token Service

Select a Security Token Service (STS) option.

No STS
Enables claims programming model for the selected application. This option does not require a Security Token Service.

Create a new STS project in the current solution
A new STS project will be added to the current solution. The selected application's configuration will be modified to trust and accept claims issued by this STS. This option is only available through the 'Add STS reference...' menu item in Visual Studio.

[✓ Use an existing STS]
The selected application's configuration will be modified to trust and accept the claims issued by an existing STS. Specify the WS-Federation metadata document location for the existing STS.

STS WS-Federation metadata document location: https://pfsts.contoso.com:9031/pf/ws-trust_sts_metadata.pi

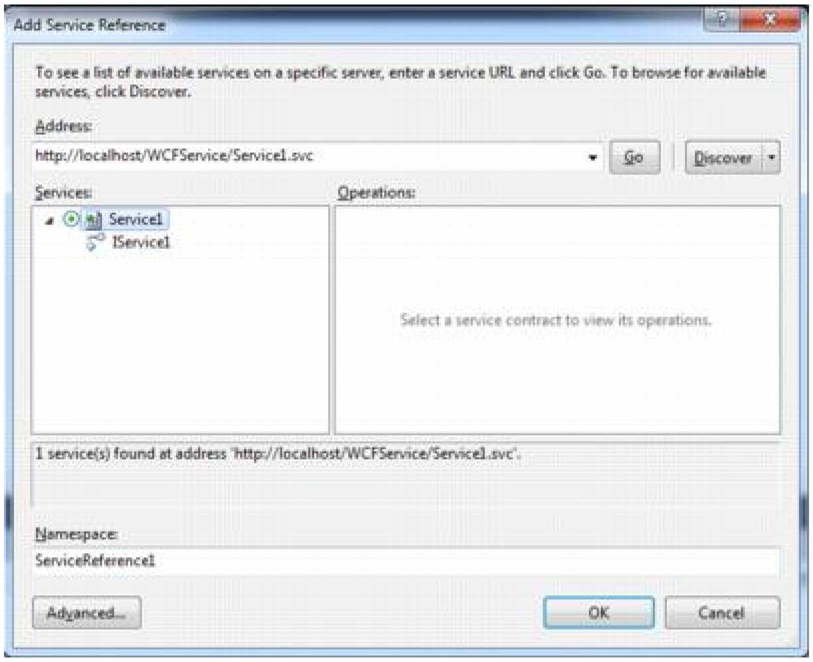
(Example: https://fabrikam.com/FederationMetadata/2007-06/FederationMetadata.xml)

MY TITLE STS Options Configuration

Continue through the configuration wizard and enable encryption specifying `CN=localhost` as the encryption certificate. For the rest of the configuration, leave default options selected. At the end a dialog box is displayed stating “Federation Utility completed successfully.” Re-publish the WCFService to the local IIS instance.

Add a service reference to the client console application

Specify the URL to the WCF Service Application endpoint using value http://localhost/WCFService/Service1.svc, as shown in Figure 1. Click **Go** and then **Ok** to add the Service Reference. The result of this operation is the insertion of the WCF binding descriptors in the `App.config` of the client Console Application, needed for accessing the WCF Service Application and the respective STS.


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost/WCFService/Service1.svc

Services: Service1, IService1

Operations: Select a service contract to view its operations.

1 service(s) found at address 'http://localhost/WCFService/Service1.svc'.

Namespace: ServiceReference1

MY TITLE Add Service Reference

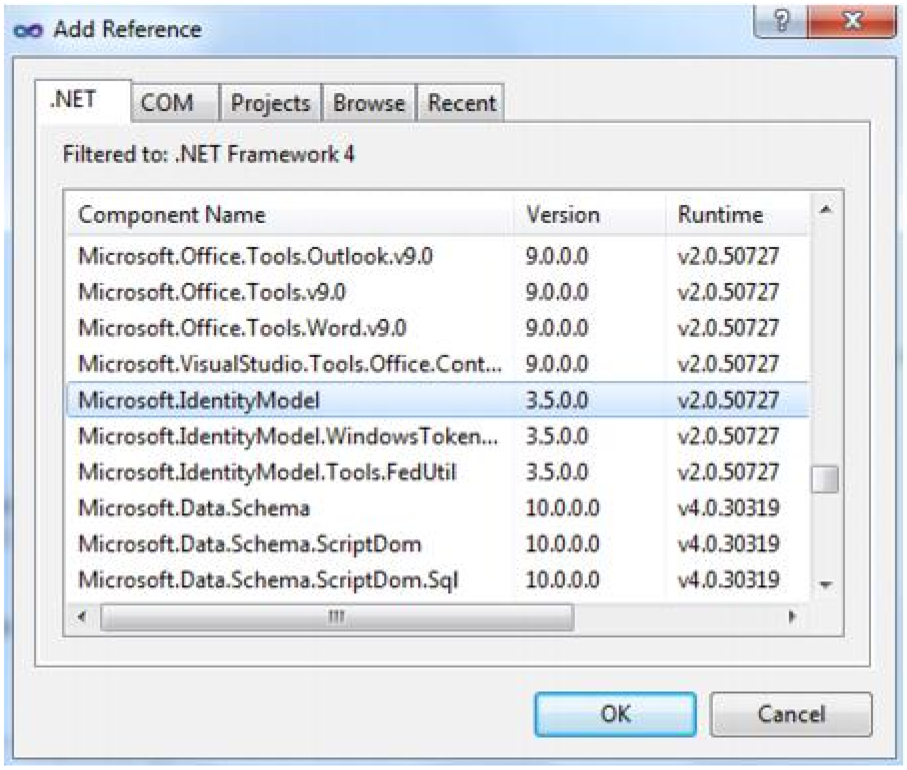
Invoke the WCF service application from the client console application

Add the following lines of code to the Program.cs file of the client Console Application within the main() method.

```
ServiceReference1.Service1Client client = new
 ServiceReference1.Service1Client();
    client.ClientCredentials.UserName.UserName = "joe";
    client.ClientCredentials.UserName.Password = "2Federate";
    System.Console.WriteLine(client.GetDasda(10));
```

Modify the WCF service application to use IClaimsPrincipal

Add a reference to the WCFService for Microsoft.IdentityModel.



Component Name	Version	Runtime
Microsoft.Office.Tools.Outlook.v9.0	9.0.0.0	v2.0.50727
Microsoft.Office.Tools.v9.0	9.0.0.0	v2.0.50727
Microsoft.Office.Tools.Word.v9.0	9.0.0.0	v2.0.50727
Microsoft.VisualStudio.Tools.Office.Cont...	9.0.0.0	v2.0.50727
Microsoft.IdentityModel	3.5.0.0	v2.0.50727
Microsoft.IdentityModel.WindowsToken...	3.5.0.0	v2.0.50727
Microsoft.IdentityModel.Tools.FedUtil	3.5.0.0	v2.0.50727
Microsoft.Data.Schema	10.0.0.0	v4.0.30319
Microsoft.Data.Schema.ScriptDom	10.0.0.0	v4.0.30319
Microsoft.Data.Schema.ScriptDom.Sql	10.0.0.0	v4.0.30319

MY TITLE Add Reference to WCF Service Application

Add the following statements to the top of Service1.svc.cs file.

```
using Microsoft.IdentityModel.Claims;
    using System.Threading;
```

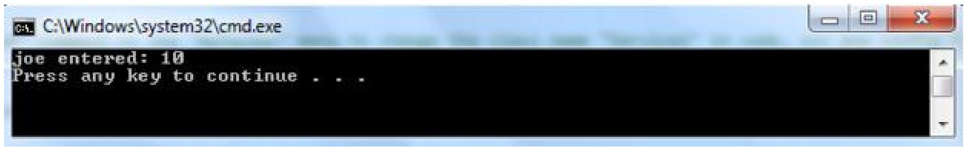
Replace the implementation of the GetData(int value) method in the Service1.svc.cs file with the following code.

```
IClaimsPrincipal icp = Thread.CurrentPrincipal as IClaimsPrincipal;
return string.Format("{0} entered: {1}", icp.Identity.Name, value);
```

Re-publish the WCF Service Application.

Start the client console applications

Start the client Console Application via the menu option **Debug** -> **Start Without Debug**



```
joe entered: 10
Press any key to continue . . .
```

See the PingFederate command window or the server log for processing information.

Trace logging

If any exceptions or errors occur during the runtime of the sample application, it is helpful to enable trace logging to obtain more details about the fault details. To enable trace logging, add the following XML directives to the App.config of the respective component of the sample application. For the WCF Service Application, add the path `C:\Windows\Temp` to the initializeData attribute value.

```
<system.diagnostics>
      <sources>
        <source name="Microsoft.IdentityModel" switchValue="Verbose">
          <listeners>
            <add name="xml"
              type="System.Diagnostics.XmlWriterTraceListener"
                initializeData="WIFTrace.svclog" />
           </listeners>
        </source>
        <source name="System.ServiceModel"
          switchValue="Information, ActivityTracing"
          propagateActivity="true">
          <listeners>
            <add name="sdt"
              type="System.Diagnostics.XmlWriterTraceListener"
              initializeData= "WCFTrace.svclog" />
          </listeners>
        </source>
      </sources>
      <trace autoflush="true" />
    </system.diagnostics>
```

The trace logging XML directives create two files. If no path is specified as part of the initializeData attribute value, the files are located in the current directory of the executable. These files are associated with the Microsoft Service Trace Viewer. For more details about trace logging, see the MSDN article “*Service Trace Viewer Tool*” (msdn.microsoft.com/en-us/library/ms732023.aspx).