Security Token Service and Microsoft Interoperability
About Integration Notes
PingFederate software developers, specialty teams, and third parties periodically submit supplemental documentation to give our customers insight into configuration, deployment, integration, or use cases that are not specifically covered by the core product documentation.

Disclaimer
This document is provided for informational purposes only, and the information herein is subject to change without notice. Ping Identity does not provide any warranties and specifically disclaims any liability in connection with this document.

Note that Ping Identity may not provide support for any sample configurations provided in this document. The variability inherent among security environments prevents full testing and support for all possible platform configurations. If you need special assistance or would like to inquire about implementation or support programs, please contact Ping Identity Support (www.pingidentity.com/support).

Document Lifetime
Ping Identity may occasionally update online documentation between releases of the related software. Consequently, if this PDF was not downloaded recently, it may not contain the most up-to-date information. Please refer to the online documentation at documentation.pingidentity.com for the most current information.

From the Web site, you may also download and refresh this PDF if it has been updated, as indicated by a change in this date: May 15, 2014.

Contact Information
Ping Identity Corporation
1001 17th Street, Suite 100
Denver, CO 80202
U.S.A.

Phone: 877.898.2905 (+1 303.468.2882 outside North America)
Fax: 303.468.2909
E-mail: info@pingidentity.com
Web Site: www.pingidentity.com
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:

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

The Translator plug-in is available from the Downloads page at pingidentity.com.

- Imported Key Pair and Certificate for digital signature and digital signature verification, see Key Pair and Certificate on page 5.

- 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 in Server Settings (on the Federation Info screen) with a Base URL of http://pfsts.contoso.com:9031.

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.

Note: For PingFederate 7.2 or higher, Username Token Processor is part of the product and does not require a separate download or installation.
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

Key Pair and Certificate

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:

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

   ```
   C:\Program Files\Microsoft Visual Studio 10.0\VC>makecert -pe -n "CN=localhost" -r -h "local\localmachine -m\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).

3. Import the public certificate into the Trusted Root Certification Authorities. The following command window shows both commands:
The first command, `certutil -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:
PingFederate Configuration

Configure PingFederate to include a WS-Trust SP Connection (see Configuring SP Connections for STS in the PingFederate Administrator’s Manual). Use the following steps to create the connection.

**Note:** It is assumed that PingFederate System Settings are configured, including using an SSL certificate trusted by the Microsoft Windows development workstation.

To configure PingFederate:

1. Import the localhost.pfx file into the Digital Signing and Decryption Keys and Certificates store.

2. Create a Username Token Processor instance with processing scheme User Table with Password and add a single row with username `joe` and password `2Federate`.

   (For detailed instructions, see the User Guide for the Username Token Processor.)

3. Follow the configuration steps for a new WS-Trust SP Connection, including the following settings:
   a. On the Connection Type screen, select WS-Trust STS using SAML 2.0 as the Default Token Type.
   b. On the General Info screen, enter the following:
      - Partner Entity ID: `http://localhost/WCFService`
      - Virtual Server ID: `http://localhost/STS`
   c. On the Protocol Settings screen under WS-Trust STS:
      - Partner Service ID: `http://localhost/WCFService`
      - Select the checkbox Generate Key for SAML Holder of Key Subject Confirmation Method
   d. On the Attribute Contract screen under Token Creation, extend the attribute contract with an attribute named:
      `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name`
   e. On the Attribute Contract Fulfillment screen:
      - Map the attribute defined above using Token as the Source and username as the Value.
      - Map the SAML subject in the same way, using the username attribute from the Token.
   f. Under Credentials:
      - On the Digital Signature Settings screen, select the certificate defined earlier with CN=localhost and select the checkbox to include the certificate in the signature <KeyInfo> element.
• On the Select XML Encryption Certificate screen, click Manage Certificates to import the public certificate localhost.crt and then select it as the encryption certificate.

g. Save and activate the connection.

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.

Step 1 – Create Visual Studio Projects

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 Figure 1). Ensure the client Console Application is set as the "StartUp Project".

![Solution Explorer](image)

Figure 1 - Initial Visual Studio solution and project setup

Step 2 – 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 2. Simply retrieving the service URL in a browser will render a page indicating a successful deployment, as shown in Figure 3.
Figure 2 - Publishing the WCF Service Application
Step 3 – 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 4. For the STS WS-Federation metadata document location use `https://pfsts.contoso.com:9031/pf/ws-trust_sts_metadata.ping?PartnerSpId=http://localhost/WCFService`.

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.
Step 4 – 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 5. 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.

![Figure 5 - Add Service Reference](image)

Step 5 – 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.

```csharp
ServiceReference1.Service1Client client = new ServiceReference1.Service1Client();
client.ClientCredentials.UserName.UserName = "joe";
client.ClientCredentials.UserName.Password = "2Federate";
System.Console.WriteLine(client.GetData(10));
```
Step 6 – Modify the WCF Service Application to use IClaimsPrincipal

Add a reference to the WCFService for Microsoft.IdentityModel (see Figure 6).

![Add Reference to WCF Service Application](image)

Figure 6 - Add Reference to WCF Service Application

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

```csharp
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.

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

Re-publish the WCF Service Application.

Step 7 – Start the Client Console Application

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

```xml
<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).