Xbox One Web Server Quick Start Guide

By Cameron Goodwin

Advanced Technology Group

Published: May 12, 2014

Updated: March 30, 2016

**This document is confidential** and provided to you under a Microsoft non-disclosure agreement. While we have tried to ensure the accuracy of this document, we provide no express warranties or guarantees regarding the information. The information is subject to change. Microsoft may have intellectual property rights in the subject matter of this paper. This document doesn’t grant you a license to those rights—it’s for informational purposes only.

Abstract

This document is a quick-start guide with step-by-step instructions for setting up a simple web server for development on Xbox One. It describes configuring two samples, **SimpleAuthService** and **DelegatedAuthService**, for communication from the Xbox One console to RESTful web services, and walks though setting up Internet Information Services (IIS) on a computer running Windows Server 2012 R2.

The March 2016 update includes a reorganization of the white paper to improve the process flow.

The December 2015 update to this white paper adds the following key items:

* Instructions for setting up Web Deploy to remotely deploy the website to the Azure server from your development PC
* Instructions for how to remotely debug your website from Visual Studio on your development PC
* Setting up Fiddler to capture server-to-server communication with Xbox Live
* Updated instructions for creating a Relying Party certificate

Contents

[Introduction 3](#_Toc447121410)

[Server setup 3](#_Toc447121411)

[Server roles and features setup 4](#_Toc447121412)

[Certificate installation 5](#_Toc447121413)

[Configuring IIS and running the sample website 6](#_Toc447121414)

[Enabling HTTPS for your web service 7](#_Toc447121415)

[Configuring the NSAL, token definitions, and SSL certificates for the web service 8](#_Toc447121416)

[Using the Xbox One Web Services sample to verify your server and NSAL configuration 12](#_Toc447121417)

[Building and deploying SimpleAuthService 12](#_Toc447121418)

[Debugging your web service 14](#_Toc447121419)

[Setting up DelegatedAuthService 16](#_Toc447121420)

[FAQs and troubleshooting 17](#_Toc447121421)

# Introduction

The Xbox One and the Xbox Live platforms rely heavily on HTTPS communication with RESTful web services. Using this form of communication has also been made available to developers because it makes setting up game servers easier than doing so was for Xbox 360, which originally required connection to a server running Xbox Live Server Platform. Being able to use HTTPS with RESTful services gives you extended flexibility to develop game services that are quick and reliable. This white paper describes how to set up Windows Server running the sample web service provided on Game Developer Network (GDN) so that you can start exploring the use of Xbox Secure Token Service (XSTS) tokens and HTTPS for communication between your Xbox One title and your servers.

# Server setup

The Relying Party SDK and samples provided on GDN for running an Xbox One web service will work with Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2. If you have access to Microsoft Azure, setting up a server for development and testing is simple and quick. If you do not have access to Azure, a physical computer with Windows Server installed will also work.

The following instructions are tailored to setting up a server running Windows Server 2012 R2 for use with **SimpleAuthService** and the Relying Party SDK. Even if your server is running another compatible version of Windows Server, the settings should be approximately the same, although the location of some settings may be different.

**To set up a server for use with SimpleAuthService and the Relying Party SDK**

1. Log on with your account to the [Azure portal](https://manage.windowsazure.com/).
2. Select **Virtual Machines**, and then click **NEW**.
3. Select **QUICK CREATE**.
4. Enter a DNS name for your service.

This will be the domain name that you use to make HTTP calls to the server (with .cloudapp.net appended to the name).

1. Select **Windows Server 2012 R2** and **Size** setting **A1**. You can choose others, but A1 works fine for the sample.
2. Enter a user name and password that you will use with Remote Desktop to connect to the machine.
3. Select which region you want the server to run in. A region close to your physical location is best.
4. To finish, click **CREATE A VIRTUAL MACHINE.**
5. Wait for provisioning to complete, and then click the machine’s name to configure the Azure settings.
6. Click **ENDPOINTS**.
7. Click **ADD**, and then select **ADD A STAND-ALONE ENDPOINT**.
8. Click **🡪** to continue.
9. Select **HTTP** (TCP port 80) from the **NAME** list, and then click the checkmark button (**✔**). Do not select **Enable Direct Server Return**, because this option might block incoming traffic to your web service.
10. Add an endpoint for **HTTPS** (TCP port 443) by following the same steps.
11. Add an endpoint named **Web Deployment for TCP public and private port 8172**; this will be used to remotely deploy the website from Visual Studio on your desktop.
12. Click **DASHBOARD**, verify that the machine has started, and then select **CONNECT** to open the .rdp file and remote desktop to the machine.
13. Log on to the machine by using the user name and password that you configured earlier in this procedure. If you are on a computer that is joined to a domain, you may need to precede the user name with a backslash to eliminate the domain name from the user name—that is, type **\*user-name***.

## Server roles and features setup

After your Azure virtual machine (VM) is running, or you have installed the OS on your physical machine, you need to configure it as a web server running Internet Information Services (IIS). To do this, you need to add *roles* to the server.

**To configure roles on your server**

1. Open **Server Manager**.
2. In the menu bar, click **Manage**, and then click **Add Roles and Features**. Windows Server displays the Add Roles and Features Wizard.
3. Click **Next**, select **Role-based or feature-based installation**, and then click **Next** again.
4. Select your server from the list under **Server Pool**, and then click **Next**.
5. From the list under **Roles**, select **Application Server** and **Web Server (IIS)**, and then click **Next**. Adding a role requires some processing time after each selection, and when you add Web Server (IIS), you might be prompted to install additional required features.
6. On the Features list, select **ASP.NET 4.5** under **.NET Framework 4.5 Features** and **HTTP Activation** under **WCF Services** in the same group. When you add these features, you may be prompted to install additional required services and features.
7. Click **Next** until the wizard displays **Roles Services** for **Application Server**.
8. Select **Web Server (IIS) Support**, and then click **Add Features** in the window that prompts you to install additional required services and features.
9. Click **Next** until the wizard displays **Role Services** for **Web Server**.
10. Keep all the default selections, and then click **Next**.
11. Review the list of items that will be installed, and then click **Install**.
12. Wait until installation of all the software and services has finished.

**To configure your server for Web Deployment from Visual Studio**

1. Install [Web Deploy 3.5](http://www.iis.net/downloads/microsoft/web-deploy) on the server.
2. Open **Server Manager**.
3. In the menu bar, click **Manage**, and then click **Add Roles and Features**.
4. Go to **Server Roles** andadd **Management Service** found under **Web Server (IIS)** -> **Management Tools**.
5. Click **Next** and finish installing the Management Service.
6. In the Azure portal, ensure that http (port 80), https (port 443), and web deploy (port 8172) endpoints are enabled.
7. Add the **IIS Web Deployment Handler** to the Web Deployment Framework:
   1. In Control Panel on the server, go to **Programs and Features**.
   2. Click **Microsoft Web Deploy 3.5**,click **Next**,and then click **Change.**
   3. In the Microsoft Web Deploy 3.5 Setup Wizard, click **Change.**
   4. Add the feature **ISS Deployment Handler** and its sub-features.
   5. Click **Next** and then finish the installation.

## Certificate installation

You must install the certificates described in this section to enable Xbox One consoles to make web service calls to your server with XSTS tokens. A procedure for installing certificates follows the list.

* **Xbox Live Signing certificate**

Download the Xbox Live Signing certificate from GDN: [Microsoft\_Services\_and\_Relying\_Party\_SDK.zip](https://developer.xboxlive.com/_layouts/xna/download.ashx?file=Microsoft_Services_and_Relying_Party_SDK.zip&folder=platform\RelyingParty). Install the certificate to the certificate stores **Personal** and **Trusted People** under **Local Computer**.

* **Your service’s Relying Party certificate**

Created by you, and then the public key is provided to Microsoft in the Xbox One Title Configuration spreadsheets on the **Relying Party** tab. Install it to the **Personal** certificate store, under **Local Computer**.

* **Business Partner certificate**

You only need the Business Partner certificate if your web service will be talking directly to Xbox Live on behalf of the user (Delegated Authentication). Install it to the **Personal** certificate store, under **Local Computer**. For more information, see [Business Partner Cert creation, management, and best practices](https://forums.xboxlive.com/questions/6709/business-partner-cert-creation-management-and-best.html), a post in the Xbox Developer Forums.

**To install a certificate on your server**

1. Open Microsoft Management Console (MMC). You can find mmc.exe in %SystemRoot%\System32\.
2. On the **File** menu, click **Add/Remove Snap-in**.
3. In the **Add or Remove Snap-ins** window, under **Available snap-ins**, select **Certificates,** and then click **Add**.
4. In **Certificates snap-in**, select **Computer account**, and then click **Next**.
5. In **Select Computer**, ensure that **Local computer** is selected, and then click **Finish**.
6. In **Add or Remove Snap-ins**, click **OK**.
7. Under **Console Root**, expand **Certificates (Local Computer)**.
8. Expand the folder of the certificate store where you will import the certificate.
9. On the **Action** menu, point to **All Tasks**, and then click **Import**. Windows Server opens the Certificate Import Wizard.
10. Use the wizard to locate and install the certificate in the specified store. By default, the wizard looks for .cer files. To install your Relying Party or Business Partner certificates, you may need to change the view in the **Open** window to **All Files (\*.\*)**.

**Note**    You might want to save your MMC configuration so that you don’t need to add the Certificates snap-in again.

Additionally, your Relying Party and Business Partner certificates will have private keys that the IIS service needs to read in order to handle XSTS tokens. To provide the IIS service access to the private keys, you need to give it permission.

**To give IIS permission to read your private keys**

1. In MMC, navigate to a Relying Party or Business Partner certificate.
2. Right-click the name of the certificate, point to **All Tasks**, and then click **Manage Private Keys**.
3. Click **Add.**
4. In **Select Users or Groups**, type the name of the local computer followed by a backslash and **SERVICE**, and then click **OK**. That is, *local-computer-name***\SERVICE**.
5. In the **Permissions** window, ensure that the SERVICE account has **Read** access to the private key.
6. Click **OK**.

# Configuring IIS and running the sample website

Before the console will be able to talk to your web server, you need to configure your service in IIS. For simplicity, start with running **SimpleAuthService**. The following steps will help you to set up an HTTP/HTTPS web service through IIS for that sample.

**To configure a web service for SimpleAuthService**

1. Open the **Internet Information Services (IIS) Manager**.
2. Expand your server’s list and right-click **Application Pools**, and then click **Add New Pool**.
3. Name the pool **SampleServicePool**, and set the pool to **.NET CLR Version 4.0**.
4. Expand the **Sites** list, and then click **Default Web Site**.
5. Under **Manage Website**, click **Stop**.
6. Right-click **Sites**, and then click **Add Website**.
7. Name the site **SampleService**, set **Application Pool** to **SampleServicePool**, and then set the physical path to **C:\SampleService\**.
8. Leave the default to port 80, click **OK**, and then click **Yes** on the binding warning. The default binding site is now turned off so there won’t be a conflict on port 80 when you run your own web service.
9. Run the following command: **%windir%\Microsoft.Net\Framework64\v4.0.30319\aspnet\_regiis.exe -ir**
10. Under **Sites**, make sure that the default website is stopped, and then restart your web service.

At this point your server should be configured for HTTP traffic. Before you enable XSTS token handling, set up the server for HTTPS traffic so that troubleshooting issues is easier.

# Enabling HTTPS for your web service

HTTPS is required for all web traffic from the console in the Retail environment, so setting this up now will help ensure your service and title are compliant.

First, check whether your server has a certificate. If you have set up your server through Azure, your server will already have a certificate.

**To check whether your server has a certificate**

1. Open **Internet Information Services (IIS) Manager**.
2. Select your server’s name from the Connection’s page.
3. Open **Server Certificates**.
4. Verify that there is a certificate with your server’s domain in the list. For example, server.cloudapp.net.
5. Select the certificate, and then under **Actions**, click **View**.
6. Click **Certification Path**, and verify that the tab displays your server’s DNS name (for example, server.cloudapp.net).

If you are manually creating an SSL certificate, make sure that for the common name (CN) you use the full DNS name of your server (for example, server.contoso.com). If your certificate does not use the full name of your server, the console will not trust the certificate because the console is talking to a server with a name that does not match the certificate.

**To create an HTTPS binding for your web service in IIS**

1. Open IIS Manager.
2. Under **Connections**, select your website.
3. Under **Actions**, click **Bindings**.
4. In **Site Bindings**, click **Add**.
5. In **Type**, select **https**.
6. In **SSL Certificate**, select your self-signed certificate.
7. Click **OK**.

You should now be able to try loading [*https://localhost*](https://localhost)*.* If you see a warning in Internet Explorer that there is a problem with the website’s certificate, this is only because the certificate is self-signed and does not chain to a trusted certificate authority (CA). Before you ship your title, you will need to set up your web service with a real certificate for your domain that is requested through a trusted CA.

The last step is getting your development kit to trust the SSL certificate from your service. If your server is using an SSL certificate that chains to a trusted CA, you should be able to call the HTTPS version of your web service without any additional steps.

Alternatively, if you are using a self-signed certificate for SSL communication, you will need to add that certificate to your endpoint definition in XDP as outlined in the next section, [Configuring the NSAL, token definitions, and SSL certificates for the web service](#_Configure_Web_Servicethe). Make sure that you have fully configured your HTTPS endpoint definition, added it to your title’s NSAL configuration, and republished it as outlined in that section. After you have configured an endpoint for your site with the SSL certificate, open up the web service sample for Xbox One.

# Configuring the NSAL, token definitions, and SSL certificates for the web service

The console uses the Network Security Authorization List (NSAL) to define the web services that your title is authorized to call. The NSAL also defines which relying party and claims list to use when generating a token to communicate with that specific endpoint.

You configure the NSAL, token definitions, and SSL certificates for web services in the Xbox Developer Portal (XDP). Once configured in XDP, the NSAL is automatically enforced on your title.

**Note**    XDP only allows you to create token definitions for HTTPS endpoints. Selecting HTTP will not allow you to assign a Relying Party certificate to that HTTP endpoint.

**To create a new Relying Party certificate**

1. On your computer, open the **Developer Command Prompt for Visual Studio**.
2. Run the following command, replacing the example name (Contoso) with your own:

makecert -sv RP\_Private\_Key.pvk -n "CN=Contoso Relying Party" RP\_Cert.cer -b 01/01/2015 -e 12/31/2199 -sky exchange -ss My -a sha256 -len 2048 -r –pe

1. When prompted, enter a password for the private key.
2. Run the following command, replacing the password at the end for your own:

PVK2PFX –pvk RP\_Private\_Key.pvk –spc RP\_Cert.cer –pfx RP\_Full\_Cert.pfx -po PasswordForPFX!

1. When prompted, enter the password you set for the private key in step 3.
2. Use the RP\_Cert.cer file to set up your token definition setup in XDP.
3. Copy the RP\_Full\_Cert.pfx file to your server and import it as outlined in the [Certificate installation](#_Certificate_installation) instructions.
4. When importing the full certificate, make sure you select the **Mark this key as exportable** option so that you can export it to other servers if needed.
5. If you ever need to re-export the public key .cer for this certificate, select the **Base-64 encoded X.509 (.CER)** option in the Certificate Export Wizard.

**To create a new SSL certificate**

**When you set up the Azure virtual machine, it should already have an SSL certificate that matches the domain name (for example, YourServer.cloudapp.net). However, if you need to create a new SSL certificate, do the following:**

1. **Open up the Windows PowerShell command prompt with Administrator privileges.**
2. **Run the following script replacing the DNS name with your server’s domain:**

New-SelfSignedCertificate -DnsName YourServer.cloudapp.net -CertStoreLocation cert:\LocalMachine\My

**To export the server’s SSL certificate public key**

1. Open MMC. You can find mmc.exe in %SystemRoot%\System32\.
2. On the **File** menu, click **Add/Remove Snap-in**.
3. In the **Add or Remove Snap-ins** window, under **Available snap-ins**, select **Certificates,** and then click **Add**.
4. In **Certificates snap-in**, select **Computer account**, and then click **Next.**
5. In **Select Computer**, ensure that **Local computer** is selected, and then click **Finish**.
6. In **Add or Remove Snap-ins**, click **OK**.
7. Under **Console Root**, expand the **Certificates (Local Computer)\Personal\Certificates** list.
8. Right-click the certificate that is issued to and by your server’s domain name (for example, YourServer.cloudapp.net).
9. Point to **All Tasks**, and then click **Export**.
10. In the Certificate Export Wizard, click **Next**.
11. Select **No, do not export the private key**, and then click **Next**.
12. Select **Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)** and the **Include all certificates in the certification path if possible** check box.
13. Click **Next**.
14. Name the certificate file with a recognizable name, and then click **Next**.
15. Click **Finish**.
16. Copy the resulting .p7b file to your development PC.

Now that you have the public keys exported for both your SSL (.p7b) and Relying Party certificate (.cer), you can add them to your NSAL endpoint definition in XDP to set up your Relying Party (XSTS) token. This enables all consoles running your title to authenticate with your web service and trust the SSL communication.

**To configure the NSAL, token definitions, and SSL certificates for the web service in XDP**

1. Go to your **Publisher** page in XDP.
2. On the **Manage** menu, click **Web Services**.
3. If you do not have a current web service to add your server to, click **New** to create one.
4. Select a definition under **Web Services** to display its list of endpoints.
5. Click **New** under **Endpoints**.
6. In **Add New Endpoint**, select the proper host type. For this quick start guide, set **Host Type** to **FQDN** (for *fully qualified domain name*).
7. Set **Protocol** to **https**. Even if your server is not yet configured for HTTPS, you can only define and add the definition for the Relying Party token by having specified HTTPS as the protocol for the endpoint.
8. For **Host Name**, enter the name of your server. For example, *yourserver.cloudapp.net*.
9. Select **Requires Single Sign On Token**. Doing so indicates that this endpoint requires an XSTS token.
10. For **Service certificate chain**, select an existing XSTS token/claim definition. If you need to create a definition, and you want instructions, see **To create a new token definition**, a procedure later in this section.
11. Click **Add New** for **Service certificate chain** to upload your SSL certificate and have it trusted by the console through Xbox Live. For more information, see [Enabling HTTPS for your web service](#_Enabling_HTTPS_for_1), later in this paper.
12. Select **SSL** as the certificate type, select **Contains Leaf**, and then click **Browse** to upload the public .p7b file for your server’s SSL certificate.
13. Click **Add Certificate Chain**.
14. Click **Save Endpoint**.
15. Click **Publish** to publish the changes.
16. Under **Web Service Management, XDP** it will display **Publish: Scheduled**, and then **Publish: In Progress**, and finally **Publish: Completed** when everything is ready.
17. Go to your title’s page in XDP.
18. For the product instance you want to modify, click **Service Configuration**.
19. Click **Network Security Authorization List**.
20. Make sure **Add to list** is selected for the web service associated with your server’s endpoint.
21. Click **Save**.
22. Click **Publish**.
23. Under **Destination Sandbox**, select the sandbox to publish the changes to.
24. Make sure **Network Security Authorization List (NSAL)** is selected.
25. Click **Submit**.
26. Click **Confirm**.
27. Wait a few minutes for the changes to be propagated through Xbox Live.

While in **Web Service Management**, whether you are adding a new endpoint or editing an existing endpoint, you can use the following procedure to create a new token definition. Start from the **Add New Endpoint** or **Edit Endpoint**, which both display the settings for an endpoint.

When you add a new token definition, some claims are added by default, but you can move these and/or add more. If you need the user’s XUID added as a claim, contact your Microsoft representative.

The name of the token must be in the form of a host URI and must end in a trailing forward slash, but the name can be different from the URI of the actual service. For example, [*http://myservice.com/*](http://myservice.com/) could be the name of service with an actual URI of *https://game.myservice.com/action/*.

**To create a new token definition**

1. While viewing the settings for an endpoint, click **Add New** next to the **Token Definition** list.
2. Enter the lifetime of a token; the default is 4 hours (00:04:00:00). To keep using a token, it must be refreshed before this time period has elapsed.
3. Click **Browse** to locate and upload the public .cer file of your Relying Party certificate. Upload only the public key .cer file that you previously created for the service to sign your tokens with.
4. Set all claims that your Relying Party token (XSTS token) will include for your services to use.
5. Click **Add Token**.

# Using the Xbox One Web Services sample to verify your server and NSAL configuration

From the Xbox One console, you should now be able to make an HTTPS call to your server’s default page <https://YourServerName/>. This will help verify that HTTPS and SSL are set up properly before we enable the service handling XSTS tokens.

**To make a call from a console to your server with the Web Services sample**

1. Download the Web Services sample for Xbox One from [Samples](https://developer.xboxlive.com/en-us/platform/development/education/Pages/Samples.aspx) on GDN, and extract it to a development PC that has the Xbox One XDK installed.
2. Open **\live\WebServices\WebServices140.sln**.
3. Open **WebServices.cpp**.
4. Update the definition of **CUSTOMURI\_HOST** to be the domain of your test server. For example, yourserver.cloudapp.net.
5. Change **CUSTOMURI\_REQUEST\_URL\_1** by temporarily commenting out L”//restservice.svc/getclaims”.
6. Open **Package.appxmanifest.**
7. Update the values of **TitleId** and **PrimaryServiceConfigId** to match the title ID and primary service configuration ID of your title in development.
8. Make sure your dev kit console is in the sandbox where your Relying Party certificate is set up.
9. Sign in with one of your developer accounts on the console.
10. Compile and run the solution.
11. Press the **Y** button on the Xbox One controller to make a call to the People Service and verify that the client sample is running properly.
12. Press the **X** buttons to make a call to the default page of your server

If you have properly set up your SSL certificate and NSAL, you should see an HTTP 200 request from the call and the response on the screen of your Xbox. If you get an error while trying to have the client sample talk to your server’s default page, see the [FAQ and troubleshooting](#_I’m_still_having) section later in this white paper.

After you have tested that SSL and HTTPS are working on your service, add back the code that you commented out earlier for **CUSTOMURI\_REQUEST\_URL\_1**.

# Building and deploying SimpleAuthService

Now that the server is configured and you have verified that IIS is set up properly, you can set up the sample service and deploy it to the web server. Visual Studio provides built in options for deploying and debugging the site on your server from your development PC. Alternatively, you can compile the sample on your development computer and then copy the compiled web service to your test server or install Visual Studio on your server.

**To build SimpleAuthService**

1. Click **Xbox One Simple Web Server Sample** on [Xbox One XDK Software Downloads](https://developer.xboxlive.com/en-us/platform/development/downloads/Pages/home.aspx) to download the sample from GDN.
2. Click **Xbox Service and Relying Party SDK** on [Xbox One XDK Software Downloads](https://developer.xboxlive.com/en-us/platform/development/downloads/Pages/home.aspx) to download the Relying Party SDK from GDN, and extract its files.
3. Open **SimpleAuthService** in Visual Studio 2015.
4. Install the IdentityModel for handling JSON web tokens from [JSON Web Token Handler for the Microsoft .NET Framework 4.5, Developer Preview 0.1.0](https://www.nuget.org/packages/Microsoft.IdentityModel.Tokens.Jwt/) on NuGet.com.
5. On the **Project** menu, click **Add Reference**.
6. Click **Browse,** and then locate and select the **Microsoft.XboxLive.Auth.dll** file that you extracted from the downloaded Relying Party SDK (.zip) file.
7. Click **OK**.
8. Open **SimpleAuthService\Web.config**, and find the **audienceUris** node.
9. Replace **http://YourRelyingParty.com/** with your publisher or service’s relying party name. This is the relying party *name* and not the actual URL for the hosted service. The name and the URL could be the same, but the value in **audienceUris** is primarily for looking up the relying party.
10. For testing, to make sure the website is active, enable directory browsing temporarily in web.config by changing the following value of **directoryBrowse** to **true**.
11. Compile the solution to verify that it succeeds. If it does compile successfully, you are ready to publish the service.

**To publish SimpleAuthService to your server remotely by using Web Deploy**

1. In Visual Studio 2015, open **Server Explorer** and sign in with your Azure account.
2. On the **Build** menu, click **Publish**.
3. Select **More Options** and **Microsoft Azure Virtual Machines**.
4. Select your VM from the list.
5. Set **Web Deploy** for the **Publish** method.
6. Enter the **username** and **password** used for your remote desktop access to the VM.
7. On the VM, enter the name of the website in IIS Manager that you created (**SampleService**).
8. Click **Validate Connection** and verify that the connection works; otherwise, see [FAQs and troubleshooting](#_FAQs_and_troubleshooting).
9. Click **Next**.
10. Select to publish the **Debug** configuration for debugging.
11. Click **Publish** to save the profile and publish your site.

**To manually publish the project to the server**

1. On the **Build** menu, click **Publish**.
2. In **Select or import a publish profile**, select **<New Profile…>** or **Custom**.
3. Name your profile **SimpleAuthService**, and then click **Next**.
4. In **Publish method**, select **File System**.
5. Set the target location to a new folder on your development PC, and then click **Next**. If you are running Visual Studio on your server, you can set this to **C:\SampleService\**, the location where IIS is configured to find the website, in the next section of this white paper.
6. Select **Debug** for the configuration so that you can debug the site later if needed, and then click **Publish**.
7. If you compiled and configured **SimpleAuthService** on your development computer, copy the resulting files in your output folder to your web server, placing the files in C:\SampleService\.

To verify that everything is working so far, try to access [*http://localhost/*](http://localhost/)from your server, and verify that you see the directory information for your deployed **SimpleAuthService**. At this point, if your server is accessible through the public Internet, you will want to disable directory browsing by setting the value of **directoryBrowse** back to **false** in the web.config file to prevent unauthorized access and discovery of your service.

Now that we know that IIS is set up, try calling one of the services from the server itself and then from your development PC, and check that you get a JSON response back:

*http://localhost/RESTService.svc/messageoftheday* (from server)

*http://myservername.com/RESTService.svc/messageoftheday* (from development PC)

If that is working, go back to the Xbox One Web Services sample and change **CUSTOMURI\_REQUIRE\_XSTS to TRUE**. This will show you any errors in obtaining an XSTS token from the client side in the sample’s UI. Compile and run the sample again on your console and use the **X** and **Y** buttons to make calls from your title to your web service. You should now be seeing additional data from the claims in the XSTS token, and the message of the day should return with the gamertag of the signed-in user on the Xbox console.

# Debugging your web service

If you set up your server with Web Deployment from Visual Studio as outlined earlier, you can also easily attach a debugger to your website from Visual Studio on your development PC.

**To attach a debug instance in Visual Studio to your server in Visual Studio**

1. Open Visual Studio and the web project.
2. Open **Server Explorer**.
3. Go to the **Virtual Machines** list.
4. Right-click your VM running the server and select **Enable Debugging**.
5. Wait for the debugging tools to finish installing.
6. Right-click the VM again and select **Attach debugger**.
7. Select the **w3wp.exe** process from the process list.
8. If the **w3wp.exe** process is not active in the list, make a call from the console to your service. This should start **w3wp.exe** if it is not currently running.
9. Place breakpoints where needed in your .cs files. In the WebService sample, this is **RestServices.svc.cs**.
10. Make the call from the console to the web service.

You should now be real-time debugging and able to place breakpoints in the example code remotely from your desktop. You can also set up Fiddler on your server to capture the HTTPS traffic from your web service, which is especially helpful when implementing B2B and obtaining the correct tokens for Delegated Authentication.

**To enable Fiddler on your server and capture your B2B network traffic**

1. Install Fiddler on your server.
2. Copy the following into your web.config file of the project:

<system.net>

<connectionManagement>

<add address="\*" maxconnection="65535" />

</connectionManagement>

<defaultProxy>

<!-- Normal networking -->

<proxy usesystemdefault="True" bypassonlocal="False" />

<!--Route through fiddler -->

<!--<proxy usesystemdefault="False" bypassonlocal="False"

proxyaddress="http://127.0.0.1:8888" /> -->

</defaultProxy>

</system.net>

1. Comment out / remove the entry under **Normal networking**.
2. Uncomment the entry under **Route through fiddler**.
3. Build and deploy your website.
4. Open Fiddler on your server.
5. Go to **Tools -> Fiddler Options -> HTTPS**.
6. Select **Capture HTTPS CONNECTs** and **Decrypt HTTPS traffic**.
7. When prompted, **trust the Fiddler Root certificate** to decrypt HTTPS traffic.
8. In **Skip decryption for the following hosts**, enter **service.auth.xboxlive.com**.
9. Click **OK**.
10. Run your server to make B2B calls.

It is critical to exclude **service.auth.xboxlive.com** from HTTPS decryption because this is the XASS service, which needs the SSL certificate to be your Business Partner certificate. With Fiddler HTTPS decryption enabled, your BP cert will be removed and the call to XASS will fail. All other calls to XSTS and Xbox Live can be decrypted just fine.

# Setting up DelegatedAuthService

Now that you have successfully deployed **SimpleAuthService** to your server and have an Xbox One console communicating to it over HTTPS with XSTS tokens, you are ready to set up **DelegatedAuthService**, another sample, on your test server. This sample will use your Business Partner certificate and the delegation claim from the XSTS token to create a delegated authentication token that you can use to call many Xbox Live services on behalf of the user. This is especially useful for the following scenarios:

* Checking the user’s inventory for quantity of consumables and consuming the balance without the user’s console acting as the middle-man.
* Getting privacy and privilege settings for a user (for example, a list of blocked users).
* Getting the user’s friends list.
* Calling Multiplayer Session services (which is required if you use your own matchmaking system).

Your next step is to install and run **DelegatedAuthService**, as described in the following procedure. To proceed, you must have set up **SimpleAuthService** and followed the instructions in this white paper to ensure that all needed software is installed on the server.

You also need to know the CN value of your Business Partner certificate. You can see the CN value of an installed certificate on your server by right-clicking its name, clicking **View**, and then clicking the **Details** tab. Select **Subject** to display the CN value.

**To install and run DelegatedAuthService**

1. Set up a new website for **DelegatedAuthService** by following the steps for **SimpleAuthService**, listed earlier in [Configuring IIS and running the sample website](#_Configuring_IIS_and). Use the local directory of **C:\DelegatedAuthService\** for the website.
2. Configure **DelegatedAuthService** to use HTTPS in IIS.
3. Stop the **SimpleAuthService** website in IIS, and then start the new **DelegatedAuthService** website.
4. Click **Xbox One Delegated Authentication Server Sample** on [Xbox One XDK Software Downloads](https://developer.xboxlive.com/en-us/platform/development/downloads/Pages/home.aspx) to download the sample from GDN.
5. Extract the files, and open the **DelegatedAuthService** solution.
6. Download and install the **Identity Model for handling JSON web tokens (JWT)** from [JSON Web Token Handler for the Microsoft .Net Framework 4.5, Developer Preview 0.1.0](https://www.nuget.org/packages/Microsoft.IdentityModel.Tokens.Jwt/) on NuGet.
7. Right-click the **DelegatedAuthService** project, and then click **Add Reference**.
8. Click **Browse,** and then locate and select the **Microsoft.XboxLive.Auth.dll** file that you extracted from the downloaded Relying Party SDK (.zip) file.
9. Click **OK**.
10. In the project, open **DelegatedAuthService\Web.config** and find the **audienceUris** node.
11. Replace **http://YourRelyingParty.com/** with the name of your publisher or service’s relying party. This is the relying party name and not the actual URL for the hosted service. The name and the URL could be the same, but the value in **audienceUris** is primarily for looking up the relying party.
12. Install (or create) your Business Partner certificate, as described earlier in [Certificate installation](#_Certificate_Iinstallation).
13. Open **RESTService.svc.cs** and change the value of **BUSINESS\_PARTNER\_CERT\_SUBJECT\_NAME** to be the CN value of your Business Partner certificate.
14. Compile the solution to verify that it succeeds. If it does compile successfully, you are ready to publish the service.
15. Follow the instructions under **To configure your server for Web Deployment from Visual Studio** by using the website name **DelegatedAuthService** instead of **SimpleAuthService,** and then publish the website.
16. Set **ENABLE\_DELEGATED\_AUTH\_URLS** to **TRUE** in WebServices.cpp of the client-side Web Services sample.
17. Compile the Web Services sample, and then deploy it to the console.

# FAQs and troubleshooting

#### When the client-side sample tries to call our service, it is getting errors from GetTokenAndSignatureAsync(). How can I resolve these errors?

Most of the NSAL or token-related errors from the console start with 0x87DD\*. [Error Codes 0x87DD\* when calling GetTokenAndSignatureAsync or Xbox.Services APIs](https://forums.xboxlive.com/questions/2936/error-codes-0x87dd-when-calling-gettokenandsignatu.html), a post in the Xbox Developer Forums, provides a list of the most common errors and how to resolve them. (Note: if when clicking this link you go to a general forum page, try pasting the link into a browser address bar to get directly to the page.)

#### When trying to access a web service URL, I’m getting HTTP error 404.3.

If you get this error when initially setting up your server with the sample, it usually means that there is a component missing. Make sure the following features are enabled in Server Manager by running the Add Roles and Features Wizard and adding any of the following that are missing:

* + **HTTP Activation** under **.NET Framework 3.5 Features**
  + **ASP.NET 4.5** under **.NET Framework 4.5 Features**
  + **HTTP Activation** under **.NET Framework 4.5 Features** / **WCF Services**

For more instructions about adding features, see [Server roles and features setup](#_Server_Rroles_and) earlier in this white paper.

If your server is running Windows Server 2008 R2, run the following command line as an administrator:

%windir%\"Microsoft.NET\Framework\v3.0\Windows Communication Foundation\ servicemodelreg" –i

#### Windows Server 2008 R2 – Config Error: The configuration section ‘system.serviceModel’ cannot be read because it is missing a section declaration.

Install .NET Framework 3.5. For more information, see [How to install/enable .NET 3.5 SP1 on Windows Server 2008 R2 for SQL Server 2008 and SQL Server 2008 R2](http://blogs.msdn.com/b/sqlblog/archive/2010/01/08/how-to-install-net-framework-3-5-sp1-on-windows-server-2008-r2-environments.aspx), a blog post on MSDN.

#### Windows Server 2008 R2 - HTTP Error 500.21 - Handler "svc-Integrated" has a bad module “ManagedPipelineHandler" in its module list.

As an administrator, run the following command line:

%windir%\Microsoft.NET\Framework\v4.0.21006\aspnet\_regiis.exe –i

#### HTTP error 500.19 and 0x80070021

If you receive this error, the handlers section on your server is currently locked. To fix this, run the following command line:

%windir%\system32\inetsrv\appcmd.exe unlock config "Default Web Site/exchange" -section:handlers -commitpath:apphost

#### HRESULT error 0x800c0019 when trying to call my web service over HTTPS from the console.

This error means that the SSL certificate that you used for HTTPS traffic is not trusted by the console, or the certificate has been misconfigured on the server.

To correct this, first make sure that the certificate you created and used for SSL on your server uses the full domain name of your server. To check, double-click the exported .cer file and look at the **Issued to** value (for example, “Issued to: Server.Contoso.com”)**.**

Next, if you have submitted your SSL certificate as a .p7b file for ingestion by XDP, check whether XDP has the **Contains Leaf** option on the certificate. If not, create another .p7b file, and be sure to include all certificates in the chain when submitting the certificate for ingestion by XDP. After submitting the certificate with the entire chain, publish the Web Services changes, wait for publication to complete, and then republish the NSAL for your title in the sandbox.

If you are manually deploying the certificate with a .cer file, make sure that the .cer file is base-64, and create a replacement if you’re not sure. When you export the .cer file, you can export a replacement that is base-64 by selecting the **Base-64** option. For more information about creating and exporting certificates, see [Enabling HTTPS for your web service](#_Enabling_HTTPS_for) earlier in this white paper.

Finally, you can use IIS Manager to check whether the HTTPS port (443) is using the certificate that you exported and not another one. Also, be sure to re-install the certificate on your dev kit.

#### I’m getting 403 errors when my console tries to talk to my service, even though it is using open HTTP and not HTTPS.

This probably means that one or both of the certificates for your relying party or the XSTS token are not in the proper certificate stores. Make sure that the certificate for the XSTS token is in Trusted People and Personal, and that the certificate for your relying party is in Personal, as described in [Certificate installation](#_Certificate_Iinstallation), earlier in this white paper.

#### When trying to run the DelegatedAuthService, I’m getting an exception when trying to obtain a token from the XASS server with my Business Partner certificate. The exception states, “An error occurred while sending this request.” Also, “The underlying connection was closed: An unexpected error occurred on a send."

This exception is probably due to an expired Business Partner certificate. Check the expiration date of your certificate to make sure that the date has not passed. Relying Party certificates will still work with the XSTS system even if they expire, but your Business Partner certificate must be in a good and valid state to be accepted by the Xbox Authentication Service for Services (XASS). XASS does not give you a 403 when presented with an invalid or expired certificate; instead, it simply drops the connection, which results in an exception and the messaging about a closed connection during debugging. To correct this, simply create a new Business Partner certificate through XDP.

#### I’m still having issues with how to properly set up my NSAL through XDP. Is there some way I can verify what is currently configured through the services?

Yes, there is a way to do this. Attach Fiddler and watch for the traffic when your title starts up. You will see a call to "title.mgt.xboxlive.com/titles/title-number/endpoints", which downloads the NSAL list for your title/sandbox. If the return call on that Fiddler entry is blank and was HTTP status code 304, copy the raw request and remove the "If-None-Match" header from the request, and then replay it in Fiddler. You should then get the full NSAL view for your title, and you can inspect it to see what might be going on.

#### I’m getting an error that the server refused the connection when trying to connect to my VM using Web Deploy.

This usually means that you did not enable an endpoint for traffic on port 8172 on the Azure portal for your server. Go to the Azure portal and create an endpoint for TCP port 8172 as outlined in [Server setup](#_Server_setup).

#### I’m getting an error of “Could not connect to the remote computer” with ERROR\_COULD\_NOT\_CONNECT\_TO\_REMOTESVC when trying to connect to my VM using Web Deploy.

This is due to the IIS Deployment Handler not being enabled on your server. Follow the instructions on [this webpage](http://stackoverflow.com/a/26254344) or see the setup instructions in [Server setup](#_Server_setup).