



# Nine simple steps to enable X.509 certificates on WCF



Shivprasad koirala, 29 Aug 2012

CPOL



4.90 (49 votes)

Nine simple steps to enable X.509 certificates on WCF.

[Download source code - 34.6 KB](#)

## Table of contents

- [Introduction and goal](#)
- [Beginner WCF FAQs](#)
- Step 1: Create client and server certificates
- Step 2: Copy the certificates in trusted people certificates
- Step 3: Specify the certification path and mode in the WCF service *web.config* file
- Step 4: Define binding
- Step 5: Tie up the bindings with the end point
- Step 6: Make your web application client for consuming the WCF service
- Step 7: Define certificates in WCF client
- Step 8: Tie up the behavior with the end point on WCF client
- Step 9: Enjoy your hard work
- [Download code](#)

## Introduction and goal

In this article, we will discuss how we can enable certificates on a WCF service. WCF has two modes by which it transfers data: transport and message. This tutorial will concentrate on how we can enable certificates on the message mode of data transfer.

Nowadays I am distributing my 400 questions and answers ebook which covers major .NET related topics like WCF, WPF, WWF, AJAX, Core .NET, SQL Server, architecture, and a lot more. I am sure you will enjoy this ebook:

<http://www.questpond.com/SampleDotNetInterviewQuestionBook.zip>. I have also been recording videos on .NET technologies, you can catch all the action [here](#).

## Beginner WCF FAQs

In case you are fresh to WCF, please refer the below two WCF FAQ articles:

- WCF FAQ [Part 1](#): This is a 20 question FAQ for beginners which explains the basic concepts of WCF like End Points, contracts, and bindings. It also discusses the various hosting methodologies of WCF services. The article finally talks about bindings and one way operations in WCF.
- WCF FAQ [Part 2](#): This FAQ covers 10 questions which talks about concepts like duplex contracts, hosting WCF on different protocols, MSMQ bindings, transaction isolation levels, and two way communication. The article also talks about two queues: volatile and dead letter queue.

## Step 1: Create client and server certificates

Create two certificates, one for the server and the other for the client, using *makecert.exe*. You can get *makecert.exe* from the "C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\Bin" folder. You can go to the DOS prompt and run the below command snippet:

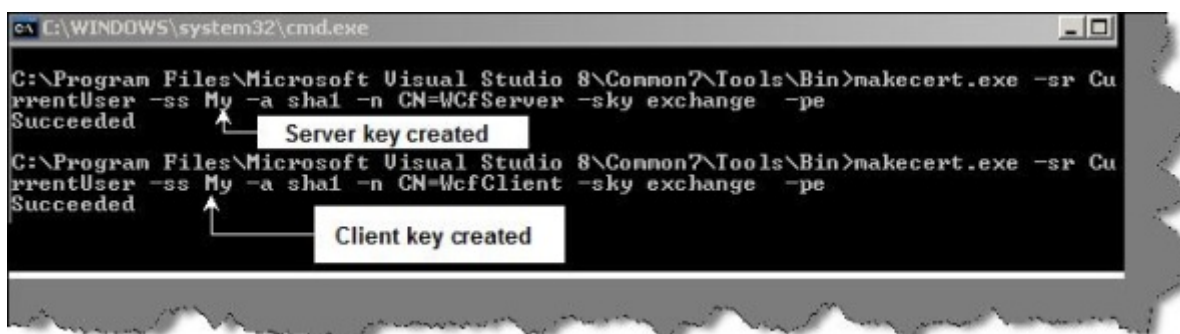
```
makecert.exe -sr CurrentUser -ss My -a sha1 -n CN=WcfServer -sky exchange -pe
makecert.exe -sr CurrentUser -ss My -a sha1 -n CN=WcfClient -sky exchange -pe
```

Below is a detailed explanation of the various attributes specified in *makecert.exe*.

Attribute	Explanation
-sr	Specifies the Registry location of the certificate store. The <b>SubjectCertStoreLocation</b> argument must be either of the following: <ul style="list-style-type: none"> <li><b>currentUser</b>: Specifies the registry location HKEY_CURRENT_USER.</li> <li><b>localMachine</b>: Specifies the registry location HKEY_LOCAL_MACHINE.</li> </ul>
-ss	Specifies the name of the certificate store where the generated certificate is saved.
-a	Specifies the algorithm. Can be either MD5 or SHA1.
-n	Specifies a name for the certificate. This name must conform to the X.500 standard. The simplest method is to use the "CN=MyName" format. If the /n switch is not specified, the default name of the certificate is "Joe's Software Emporium".
-sky	Specifies the key type. Can be either exchange or signature.
-pe	This makes the key exportable.

**Note:** *Makecert.exe* is a free tool provided by Microsoft which helps to create X.509 certificates that are signed by a system test root key or by another specified key. This is a test certificate and not a real one and should not be used for production purposes. For production, buy proper certificates from Thawte, Verisign, GeoTrust, etc.

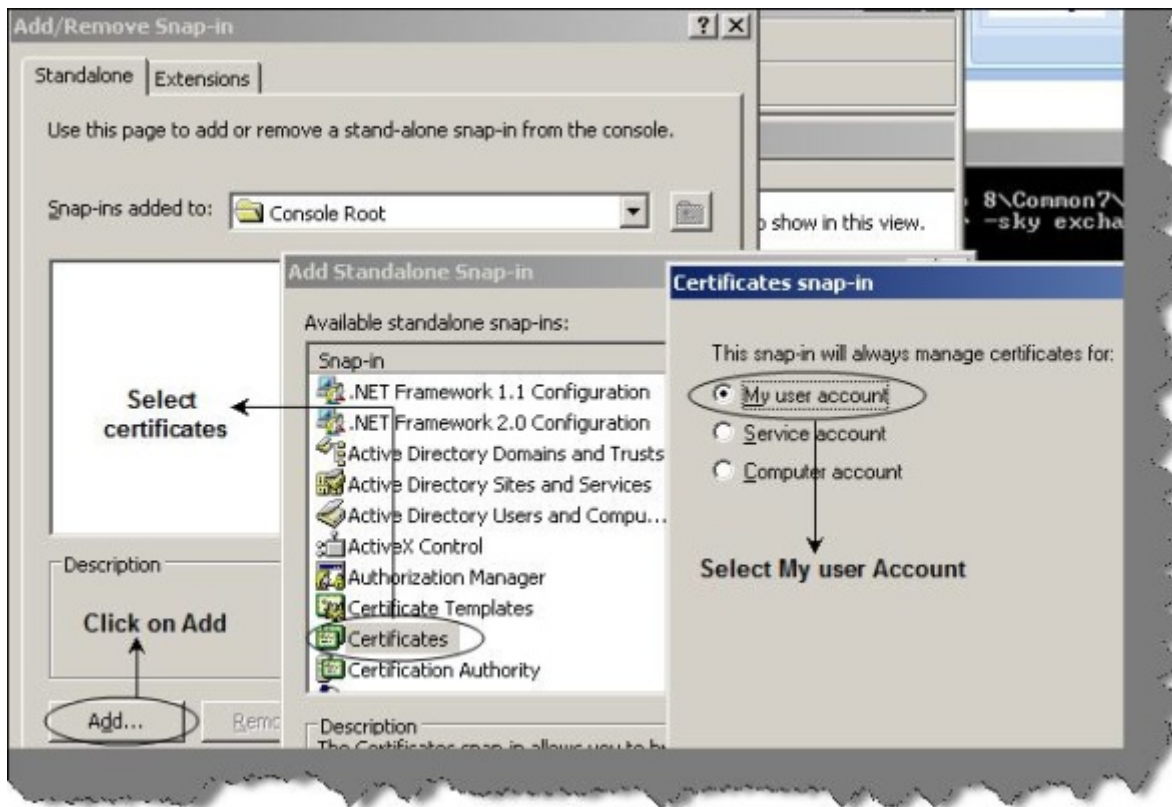
Currently, we have specified that we want to create the client key with the **WcfClient** name and server key with **WcfServer**. The certificates should be created for the current user and should be exportable.



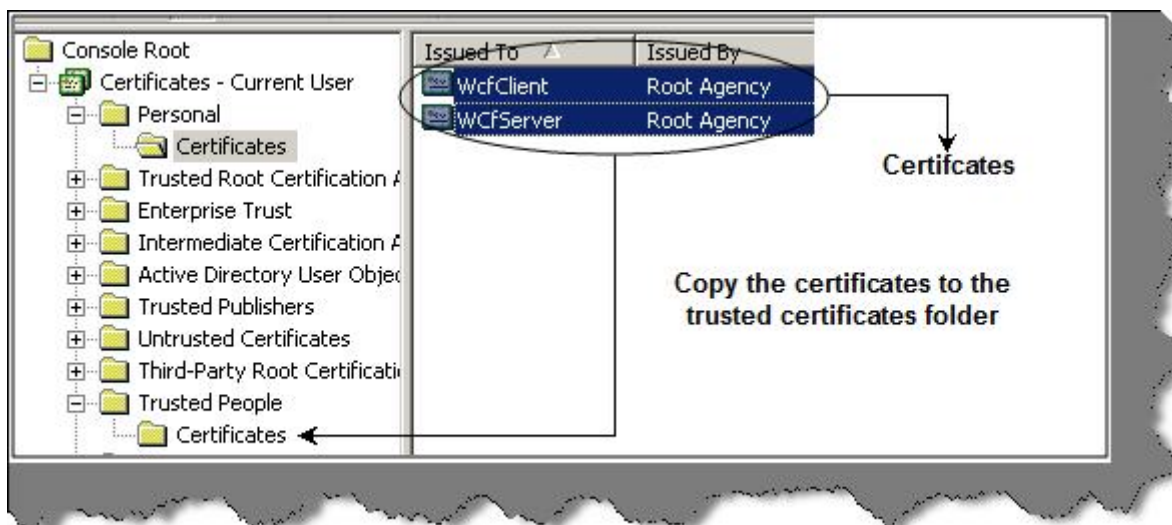
Once you run the command, you should see the **Succeeded** message as shown in the below figure. The below figure shows keys created for both the server and client.

## Step 2: Copy the certificates in trusted people certificates

Go to Start -> Run and type MMC and press Enter. You will be popped with the MMC console. Click on File -> Add/remove snap-in. You will be popped up with an Add/Remove snap-in, click on the Add button, select Certificates, and select 'My user account'.

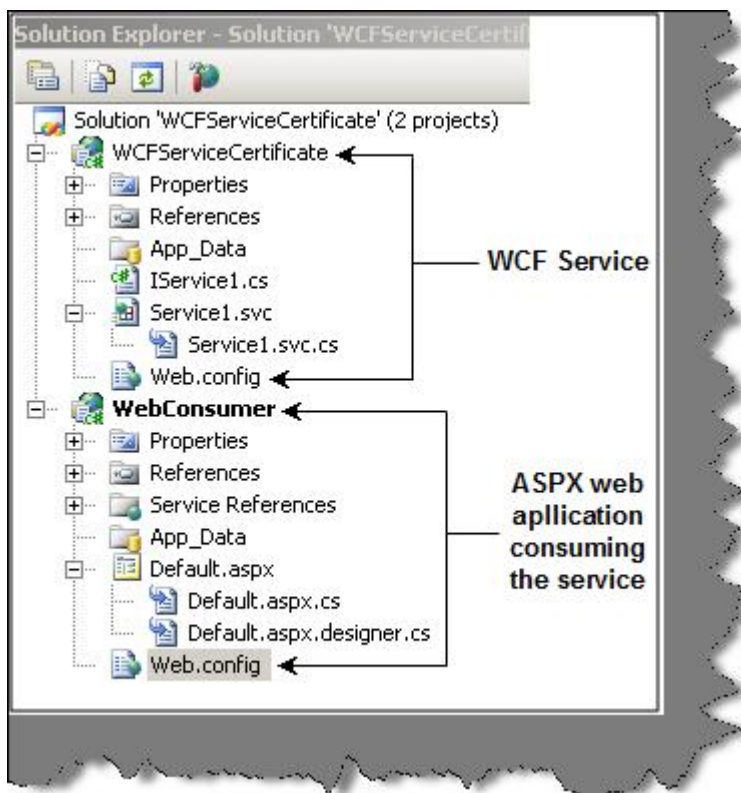


You can see the certificates created for the client and server in the personal certificates folder. We need to copy those certificates in the Trusted people -> Certificates folder.



## Step 3: Specify the certification path and mode in the WCF service web.config file

Now that we have created both the certificates, we need to refer these certificates in our WCF project. We have created two projects: one that has the WCF service and the other a web application which will consume the WCF service.



Let's open the *web.config* file of the WCF service and enter two important things:

- Where the certificate is stored, location, and how the WCF application should find it. This is defined using the **serviceCertificate** tag as shown in the below snippet.
- **certificationvalidationmode** defines how the client certificates will be authenticated.

Certification validation mode	Description
Chain trust	In this situation, the client certificate is validated against the root certificate.
Peer trust	PeerTrust ensures that the public key portion of the certificate is in the Trusted People certificate folder on the client's computer
ChainORPeertrust	This is just an OR condition for both chain and peer.

The above two points are clubbed together and entered in the *web.config* file of the WCF service.

```
<serviceCredentials>
  <clientCertificate>
    <authentication certificationValidationMode="PeerTrust"/>
  </clientCertificate>
  <serviceCertificate findValue="WCfServer"

    storeLocation="CurrentUser"

    storeName="My"

    x509FindType="FindBySubjectName" />
</serviceCredentials>
```

## Step 4: Define bindings

Now that we have defined our certificates and authentication type, we need to define that the authentication values will be sent through a message using certificates. You can see we have defined the **WsHttpBinding** with a **message** attribute specifying that the WCF client needs to send a certificate for validation.

```
<bindings>
  <wsHttpBinding>
    <binding name="wsHttpEndpointBinding">
      <security>
        <message clientCredentialType="Certificate" />
      </security>
    </binding>
  </wsHttpBinding>
</bindings>
```

## Step 5: Tie up the bindings with the endpoint

Once done, we need to tie up this binding with the end point. This is done by using the **bindingConfiguration** tag as shown in the below code snippet.

```
<endpoint address="" binding="wsHttpBinding"
  bindingConfiguration="wsHttpEndpointBinding" contract="WCFServiceCertificate.IService1">
```

## Step 6: Make your web application client for consuming the WCF service

That's all we need from the WCF service perspective. Compile the WCF service and reference it in the ASP.NET web application using 'Service reference'. Below is the code snippet where we have referenced the service and called the **GetData** function of the service.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WebConsumer.ServiceReference1;
namespace WebConsumer
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Service1Client obj = new Service1Client();
            Response.Write(obj.GetData(12));
        }
    }
}
```

Now if you try to run the client, i.e., the web application, as it is, you should get an error as shown below. The error clearly indicates you can not use the WCF service until you provide the client certificate.

### Server Error in '/' Application.

**The client certificate is not provided. Specify a client certificate in ClientCredentials.**

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error.

**Exception Details:** System.InvalidOperationException: The client certificate is not provided. Specify a client certificate in ClientCredentials.

## Step 7: Define the certificates in the WCF client

Let's start the process of defining certificates in the WCF client. The way we have defined the authentication certification mode and the path of the certificate, the same way we need to define it for the WCF client. You can see we have defined the authentication mode as **peertrust** and we have specified the client certificate name as **WcfClient**.

```
<behaviors>
  <endpointBehaviors>
    <behavior name="CustomBehavior">
      <clientCredentials>
        <clientCertificate findValue="WcfClient" x509FindType="FindBySubjectName"

          storeLocation="CurrentUser" storeName="My" />
        <serviceCertificate>
          <authentication certificateValidationMode="PeerTrust"/>
        </serviceCertificate>
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>
```

## Step 8: Tie up the behavior with the end point on the WCF client

We need to tie up the above defined behavior with the end point. You can see we have bound the behavior using the **behaviorConfiguration** property. We also need to specify that the DNS value will be **WcfServer** which is your server certificate name.

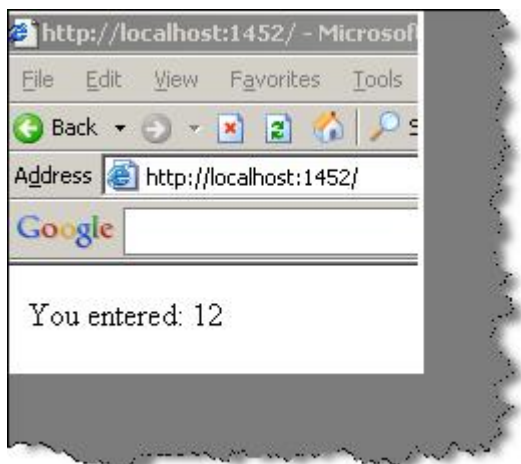
```
<client>
  <endpoint address="http://localhost:1387/Service1.svc" binding="wsHttpBinding"

    bindingConfiguration="WSHttpBinding_IService1" contract="ServiceReference1.IService1"

    name="WSHttpBinding_IService1" behaviorConfiguration="CustomBehavior">
    <identity>
      <dns value="WcfServer" />
    </identity>
  </endpoint>
</client>
```

## Step 9: Enjoy your hard work

Once we are done, you can run the ASP.NET web app and you should see the below display.





## Download code

You can download both the server and client code from [here](#).

## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

## Share

## About the Author



### Shivprasad koirala

Architect <http://www.questpond.com>

India 

Do not forget to watch my Learn step by step video series.

- [Learn MVC 5 step by step in 16 hours](#)
- [Learn MVC Core \(MVC 6\) Step by Step](#)
- [Learn Angular 2 for beginners \( step by step\)](#)
- [Learn AngularJS 1.x Step by Step](#)
- [Learn Design Pattern in 8 hours](#)
- [Learn C# in 100 hours series](#)
- [Learn SQL Server in 16 hours series](#)
- [Learn Javascript in 2 hours series](#)
- [Learn MSBI in 32 hours](#)
- [Learn SharePoint Step by Step in 8 hours](#)
- [Learn TypeScript in 45 minutes](#)

## You may also be interested in...

[Seven simple steps to enable HTTPS on WCF WsHttp bindings](#)

[Equipment Activity Monitor in Python](#)


[6 Steps to Enable Transactions in WCF](#)

[Doorbell in Python](#)

[IoT Reference Implementation: How to Build an Environment Monitor Solution](#)

[SAPrefs - Netscape-like Preferences Dialog](#)

# Comments and Discussions

 **65 messages** have been posted for this article Visit <https://www.codeproject.com/Articles/36683/simple-steps-to-enable-X-certificates-on-WCF> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | Mobile  
Web02 | 2.8.170422.1 | Last Updated 29 Aug 2012

Select Language ▼

Article Copyright 2009 by **Shivprasad koirala**  
Everything else Copyright © [CodeProject](#), 1999-2017