# Positive Affirmations WCF Service Library & Windows Services Hosting & Client “Tester" Windows Form Application

## About

This project presents a simple WCF Positive Affirmations Service that demos architectural styles of setting up, hosting via a Windows Service, configuring services, and testing for tcp protocols. The service accepts a simple string, which represents a name, and returns a simple string with a randomized positive affirmation that acknowledges the name. Instead of using IIS Express, the simple service is hosted using Windows Services. The Visual Studio solution also has one client “tester” Windows Form application that tests the tcp protocol connection to the hosted service.

## Architecture

The demo project consists of these component topics:

* MyAffirmationServiceLib Service Library
  + IAffirmationService (Interface for Service)
  + AffirmationService (Code that Implements the Service Interface)
* PositiveAffirmationsHost “Service Host” Windows Service Application
  + Hosts the Service
  + Installer Included and Configured
  + Started/Stopped Using Windows Services
* TestClientGUI “Tester to Service” Windows Form Application
  + Connected Service “Proxy Reference”
  + Simple Program to test the service @ tcp endpoint

## Positive Affirmations “MyAffirmationServiceLib” Service Library

A WCF Service Library project was added to my Visual Studio solution. The code is available on GitHub here [].

[insert code here]

### IAffirmationService (Interface for Service)

The ServiceContract for the Simple Greeting service has only one OperationContract: a method called “AffirmMe” that accepts a string representing a name. The “AffirmMe” returns string to the caller that includes a positive affirmation plus the name. The code is available on GitHub here [].

[insert code here]

### AffirmationService (Code that Implements the Service Interface)

The service implementation code details the “AffirmMe” method. It returns a simple string containing a positive affirmation plus the name that the caller sent. A list of approximately 30 positive affirmations are loaded into memory when the service spins up with a random number generator. When the service is called, it randomly returns an item from the list of positive affirmations along with a personalized message to return to the caller. The code is available on GitHub here [].

[insert code here]

## PositiveAffirmationsHost “Service Host” Windows Services Application

Instead of hosting with IIS Express, the Positive Affirmations Service is hosted and managed via Windows Services with specific ports, configuration, endpoints, etc. to allow access via tcp protocol (suited for local or Intranet traffic). The service is managed via a Windows Service application that starts and stops the service. The application includes an installer that specifies the Windows Service behavior to be manually started along with other custom settings. The code is available on GitHub here [].

### Main Program

The main program spins up an instance of the Positive Affirmations Service Library (MyAffirmationServiceLib) logging that the user that it has successfully started the service (Windows Event Log) and waits for the user input to stop the service. It catches any errors, alerts to the windows event log, and finalizes by properly closing the service host after the service is stopped. The code is available on GitHub here [].

The Program.cs is the Main entry point for the Windows Service to create an instance.

[insert code here]

The PositiveAffirmationService manages the creation and access of a ServiceHost that hosts the Positive Affirmations Service Library. The ServiceHost creates an instance where the program can then be accessed. This hosting manages the opening and closing of the service via the OnStart and OnStop methods. The events, errors, and success messages are logged in the Windows Event Log.

[insert code here]

### App Configuration “App.Config”

The app configuration “app.config” file for the service host is vital for the correct operations of the service host. The ServiceModel from the Positive Affirmations Service Library (MyAffirmationServiceLib) is copied and pasted to the actual executable host project, in this case, the PositiveAffirmationsHost (Windows Service). The *serviceModel* from the service library “dll” cannot run unless it’s on an executable, so that is why the serviceModel configuration is copied to where the actual service will be hosted. It gives instructions on how to host it, kind of like a recipe.

The service model was customized and configured to have endpoints for the tcp protocol (suitable for local and Intranet access). The ABC’s (address, binding, contract) are specified for each endpoint. There are two endpoints, one of which contains the metadata exchange endpoint, while the other is service for the tcp. This app configuration has to be customized for the service project, as only basic configuration is auto-generated per the service library project template. The code is available on GitHub here [].

Notes and Helpful Knowledge:

1. TCP binding protocol was specified as this is suitable to local or Intranet traffic
2. Check your TCP port availability before customizing this service to use the port
3. A behaviorConfiguration was added and referenced to address this run-time error:

*The contract name ‘IMetadataExchange’ could not be found in the list of contracts implemented by the service MyService. Add a ServiceMetadataBehavior to the configuration file or to the ServiceHost directly to enable support for this contract.*

Kudos to KIRAN DHAPPURI for writing a blog article here [<https://kirandhappuri.wordpress.com/2014/05/14/wcf-the-contract-name-imetadataexchange-could-not-be-found-in-the-list-of-contracts/comment-page-1/>] that explains how to modify your application configuration file correctly to address this issue.

[insert code here]

### Install into Windows Services

The project installer was added to allow the installation of the service with custom settings (such as Manual starting). The service can be installed into Windows services by the following procedure:

1. Open and run the Visual Studio Developer Command prompt in administrative mode
2. Navigate to the executable directory (ex: Release) of the Windows Service application
3. Enter the following command: ***installutil -i PositiveAffirmationsHost.exe***

### Uninstall from Windows Services

The project installer was added to allow the uninstall of the service. The service can be uninstalled into Windows services by the following procedure\*:

1. Open and run the Visual Studio Developer Command prompt in administrative mode
2. Navigate to the executable directory (ex: Release) of the Windows Service application
3. Enter the following command: ***installutil -u PositiveAffirmationsHost.exe***

\*Note: You need to have the executable available to uninstall. I have also successfully uninstalled the service from different versions of the executable with the same name.

### Starting the Windows Service

The Positive Affirmations Service was designed to be started manually in Windows Service. The user can start the service using the Services Control Panel or via command window. The service can also be started using this command: ***net start "Positive Affirmations"***

### Stopping the Windows Service

The Positive Affirmations Service was designed to be stopped manually in Windows Service. The user can stop the service using the Services Control Panel or via command window. The service can also be started using this command: ***net stop "Positive Affirmations"***

## TcpClient “Tester to Service” Windows Form Application (TestClientGUI)

The TcpClient “tester to service” is a simple windows form application project in the same solution that connects to the “Positive Affirmations Service” by use of a proxy generated by SVCUTIL. The client program will use this proxy and the net.tcp protocol to test the OperationContract or method available in the ServiceContract and return the results to the user on the GUI.

### Connected Service “Proxy Reference” MyAffirmServiceRef

I used the simple “Add Service Reference” wizard to create a Service Reference to an existing service in my Visual Studio solution. The existing Windows Service host application had to be started (*not in Debug Mode*) for the wizard to be able to connect to the metadata exchange via the base address provided and add a service reference. Note: the base address provided was specific to the protocol. For the tcp client, the metadata would be available after entering this address into the wizard:

net.tcp://localhost:999/AffirmMe/

Note: the protocol used was tcp. The client was able to then recognize the Positive Affirmations Windows Service and build the service reference using the exposed meta-exchange data “WSDL”. The auto-generated code is available on GitHub here [].

### Main Form Program

The main form program in the client “tester” windows form application creates a proxy using the service reference that was previously created using the wizard, to connect to the simple positive affirmations service. Since there is one service endpoint (tcp), the proxy does not need to specify the specific endpoint name on the service as it relates to the protocol.

AffirmationServiceClient proxy = new AffirmationServiceClient();

#### Good to Know – Endpoints and Client Implementations

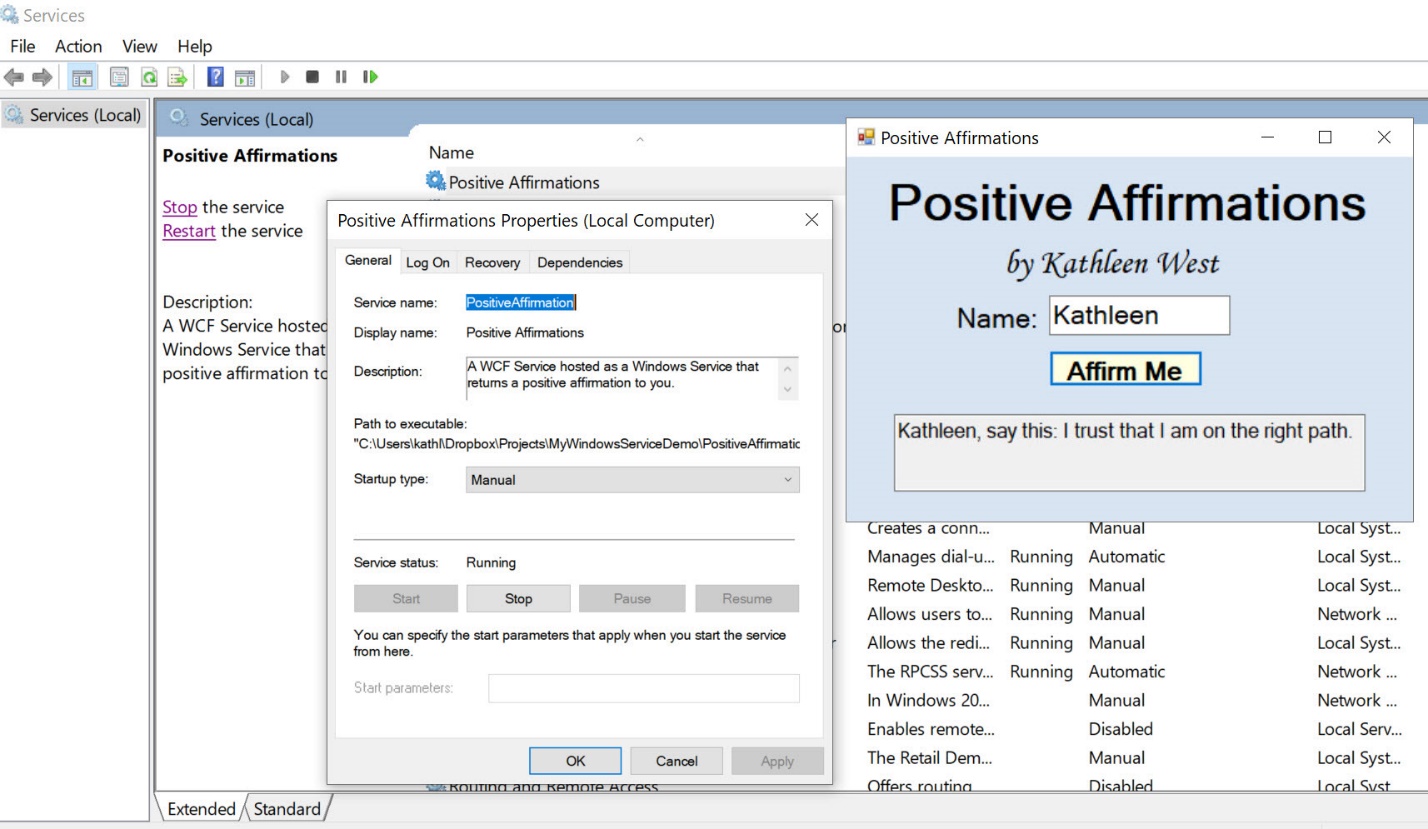
The service could also be implemented using the http protocol, and it in that case, the endpoint would have to be specified in the constructor above. The endpoint name for the specific endpoint differentiates the specific protocol (ex: net.tcp) and make sure it does not connect to the default binding (ex: http). The endpoint name is available in the app configuration file on the service host, so the client has to be aware of the endpoints. They can also learn this from the metadata exchange or WSDL on the service. The correct endpoint name for the protocol must be specified in the creation of the proxy to use that protocol if it not the default.

### Testing the Service

After the proxy is setup, the tester windows form application loads the GUI and allows the user to enter their name. After they click the button “AffirmMe”, the program then makes a request to the service (instancing the proxy) passing their name as a string parameter. After it receives the response from the service, it outputs the result as a string to the text box on the form. If it cannot connect to the service or fails, it will say so in the text box on the GUI. The result is a personalized positive affirmation to the user who requested the service. The code is available on GitHub here [].

[insert code here]

## Demo



## Code

The entire project code repository is available on GitHub here.