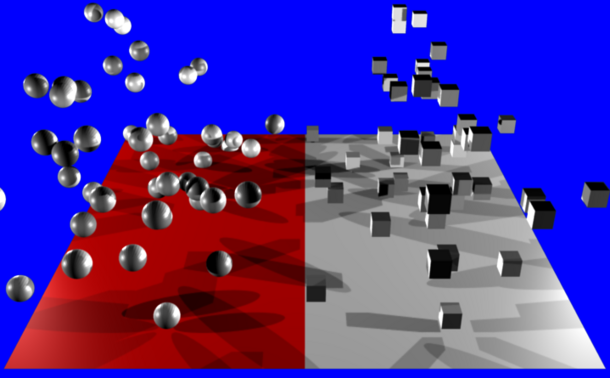
**Instancing in WCF**

****

**Rameshkartik. RS**

**Table of Contents**

[Introduction 3](#_Toc418598839)

[Available Parameters for Protecting Data 3](#_Toc418598840)

[Protection Level 3](#_Toc418598841)

[Configuring Protection Level 4](#_Toc418598842)

[ProtectionLevel.None 5](#_Toc418598843)

[PerSession 5](#_Toc418598844)

[PerCall 8](#_Toc418598845)

[The above one is declaration of InstanceMode->Percall. You need to use the service behavior attribute at the top of the service implementation class. 9](#_Toc418598846)

[Single 10](#_Toc418598847)

[Attachment 12](#_Toc418598848)

[Summary 12](#_Toc418598849)

**WHAT’s IN THIS ARTicle?**

* What is InstanceContextMode?
* PerSession
* PerCall
* Single
* Attachment
* Summary

# Introduction

In the modern world we are about to communicate with different persons,firms in our day to day life to get our business done, in which we are always concern about the confidentiality and integrity of the information we send. Often we send information which is highly secure for the business,In few cases we send information which is not to be a secured one or it might be irrelevant to the business. Ok, Lets talk about it in detail with the WCF technology.

# Available Parameters for Protecting Data

Sign – It conforms the message has not been tampered or changed by anyone. It signs all the messages on the wire to provide the message integrity

Encrypt and Sign – It conforms the message we send is confidential and it has not been tampered anywhere by any one.

# Protection Level

Protection Level = Throttle(Control) Message Protection

By default WCF protects all the message on wire to provide the data confidentiality and message integrity. If the developer wants to control the flow of message protection, Protection Level would be an ideal option. Yes , In Few scenarios,messages are not required to be a secured one since it doesn’t have any sensitive information,In few scenarios messages should transmit only in a secure mode. For these set of things, WCF provides the flexible approach called ‘Protection Level’ . The developer can set their messages which requires message integrity or message confidentiality or Both

If the data is not sensitive over the wire then I would **recommend** you to set the ProtectionLevel as **None** because it produces a greater performance. Ofcourse, Encryption or Sign will definetly take an additional time in every request and response. Message contracts also support a way to override the protection level for the operation.

# Configuring Protection Level

The supported message protection level in the WCF are

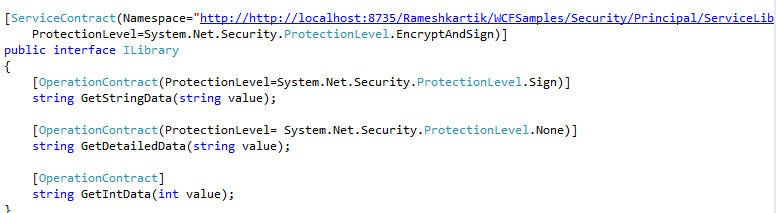
* ProtectionLevel.None
* ProtectionLevel.Sign
* ProtectionLevel.EncryptAndSign

The Protection levels can be done at all the levels

* Service Contract
  + Operation Contract
    - Message Contract
      * Message Header
        + Message Body

If we are defining our protection level at the top level,it is applicable till the root level unless otherwise if we are overriding it in the root level . You can define it like ProtectionLevel is EncryptAndSign in the top level(Service Contract) and override it in the bottom level(Operation Contract) like ProtectionLevel is None. The developer has a flexible approach for controlling the message protection across all the levels.

## Declaration



In the above snapshot you can find that the protection level has to be declared over the service method .For the method *GetStringData* i have made it as Sign,because I don’t care about the confidentiality here but I care for the message integrity, That’s why I have chosen Protection Level.Sign.

For the *GetDetailedData* method I have set the Protection Level as ProtectionLevel.None though my service contract Protection Level configured as ProtectionLevel.EncryptAndSign because i don’t want to protect this method’s message on wire.

For the GetIntData method, I didn’t say anything about the Protection Level, Then What would be the Message Protection Level? Of course it will take the Protection level we said in the Service contract level.

## ProtectionLevel.None

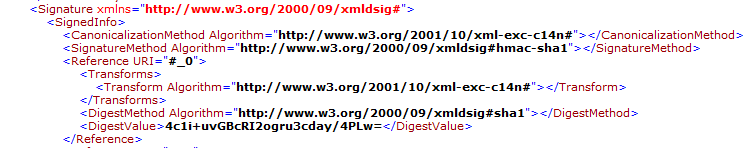
The following is the snap shot of the message flow if we set the ProtectionLevel.None.



You can find that the values are transmitting over the wire as a plain text. There is no encryption or Sign. Because we have mentioned the Protection Level is None,Kindly refer App.config of the Service Library which is enclosed , you can find the ways to generate the logs. Once it generated have a look on the messages.svclog .

## ProtectionLevel.Sign

The following is the snap shot of the message flow if we set the ProtectionLevel.Sign.

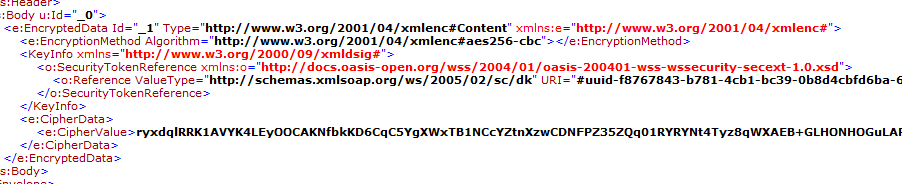




You can find that Data Integrity is maintained but the confidentiality is not maintained. Signature information in the above snapshot will take care of the Data Integratity related things, In the body of the message in the above snapshot you can find that the messages were plain and not encrypted.

## ProtectionLevel.EncryptedAndSign

The following is the snap shot of the message flow if we set the ProtectionLevel.EncryptedAndSign.



If you find the body of the message, you can find that messages were encrypted and signed. By setting the different ProtectionLevel in different levels we can make our service more efficient and responsive.

## Binding Constraints

Production Level is relies on WS-Addressing to support this kind of different level of production level in different level, so it will result an unexpected behavior if the binding does not support the WS-Addressing spec, For Ex:BasicHttpBinding.

Also that it does not have any impact if you are configuring the protection level in Transport Layer security. For Example if you are running WCF application over HTTP SSL,it does not matter which Protection Level we are using as by default it will be encrypt and sign at the transport layer

-protection level at levels

-supported values

- Declarative which binding, configurable which binding

InstanceContextMode determines the client proxy to communicate with the server in the following manners

* Using one and the same instance of the server object (PerSession)
* Using the new instance that is created on, every method call(PerCall)
* Using the single instance that is created for all the clients(Single)

What exactly the benefit you will get on using each InstanceContextMode? In which situation each InstanceContextMode should be used? We will discuss all the InstanceContextMode in detail

# PerSession

ONE SERVER INSTANCE PER CLIENT SESSION

-Means the client proxy communicates with one and same server instance till the session times out. In other words server object resides on the server as long as session times out(Default 10 minutes).

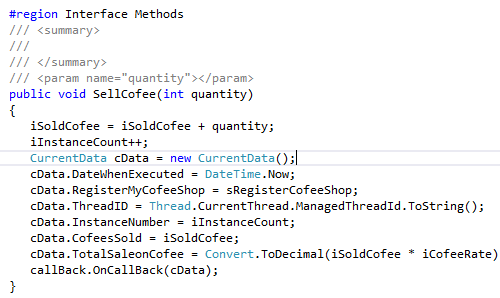
Advantage – The main advantage of Per session is that values which have been set are retained in the server side without any additional measures.

Disadvantage – scalability is the main disadvantage here, because every active session takes memory in the server regardless of the client currently requires or not.

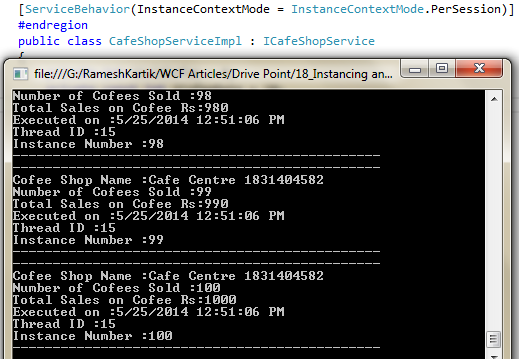
Persession requires you to use a protocol which supports session. For example, you can use nettcpbinding with the connection oriented protocol TCP which allows the server to allocate the incoming message to the correct server object. The basichttpbinding is unsuitable because the transfer protocol HTTP is connectionless. You can also enforce the service to use the right binding by using the keyword SessionMode.Required in the service contract attribute.

DeclarationSession.png

The above is the declaration of InstanceContextMode->Session. You need to use the service behavior attribute at the top of the service implementation class.



In the attached source code ,if you see the service interface method the iInstanceCount will be used. It is just used to hold the number of times this method has been executed.As I told you earlier, the Persession actually creates one server instance per session client,and the information will be retained in the server till the session expires.In our case the iInstanceCount will be incremented on every method call from client proxy.And the below snapshot is the output of the program

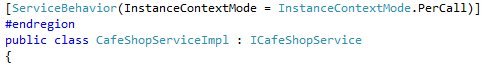


You can find that information will be maintained in the service. For further details please refer the attached source code.

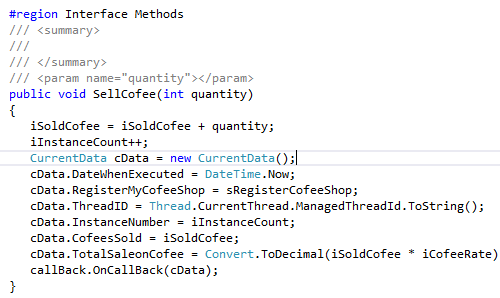
# PerCall

ONE SERVER INSTANCE PER METHOD CALL

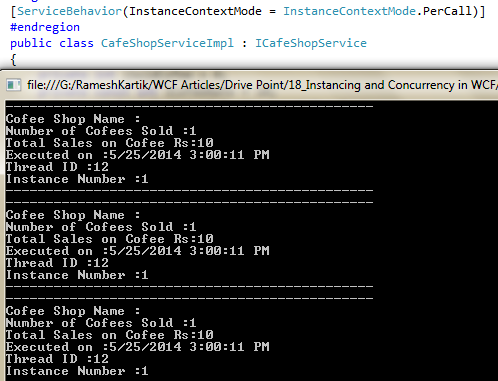
-Means the client proxy communicates with the new service instance, which is created on, every method call from the client proxy. Yes, on every server method invocation from the client proxy the new instances will be created,as soon as the method is processed the new instance will be released again. If the service object implements the IDisposable interface, the Dispose method is called automatically after the result is sent to the proxy. This is extremely scalable since the object is released on, every method call,Also no need to worry about threading problems. But the main drawback here is we cannot able to maintain the information in the server side.



# The above one is declaration of InstanceMode->Percall. You need to use the service behavior attribute at the top of the service implementation class.



In the attached source code ,if you see the service interface method the iInstanceCount will be used. It is just used to hold the number of times this method has been executed.As I told you earlier, the PerCall actually creates one server instance per method call,so the information will not be retained on the server. In our case the iInstanceCount will be incremented on, every method call from client proxy and it will be disposed after the method call. And the below snapshot is the output of the program

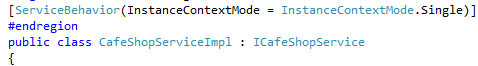


Refer the attached code for the further details

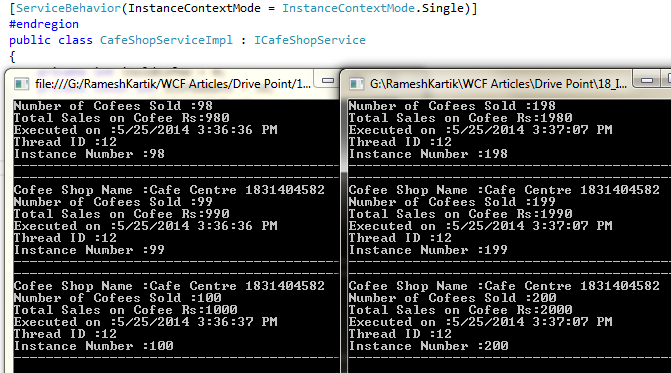
# Single

ONE SERVER INSTANCE FOR ALL THE CLIENTS

As the name suggests one server instance that is created will be for all the clients whoever connecting the server. Often we will come across the situation of creating global instances for all the clients,For that InstanceMode.Single would be very useful. The service instance created to serve for client 1 , will also used to serve the clients 2 and 3.



Above one is the declaration for the InstanceMode->Single.



As I told you earlier the global server instance, will be created for all the clients. If you notice the above output you can find that Instance number will be shared across all the clients. If you run the second client the Instancenumber value will be incremented from the 100 where the client1 has finished the process. You can refer the attached source code for further details

# Attachment

Refer the source code attachment for further details

# Summary

|  |  |
| --- | --- |
| **PerSession** | One Server instance on every client session |
| **PerCall** | One Server instance on every method call |
| **Single** | One Server instance for all the clients |