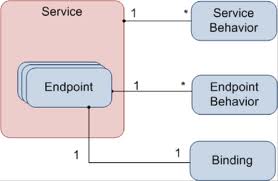
**Service Behavior in WCF**

****

**Rameshkartik.RS**

**Table of Contents**

[Introduction 4](#_Toc389911313)

[Service Behavior 4](#_Toc389911314)

[Types of Transactions 7](#_Toc389911315)

[Transaction Protocols 7](#_Toc389911316)

[WCF Transactions 9](#_Toc389911317)

[Transaction Bindings 10](#_Toc389911318)

[Transaction Flow Option 11](#_Toc389911324)

[ Transaction Flow option – Not Allowed 11](#_Toc389911327)

[ Transaction Flow option – Allowed 11](#_Toc389911328)

[ Transaction Flow option – Mandatory 11](#_Toc389911329)

[Transaction Scope 13](#_Toc389911343)

[Configuring Transaction Modes 14](#_Toc389911344)

[Throttling 17](#_Toc389911345)

[Attachment 18](#_Toc389911346)

[Summary 18](#_Toc389911347)

**WHAT’s IN THIS ARTicle?**

* Introduction
* Transactions
* Throttling
* Attachment
* Summary

# Introduction

Your good behavior make you a better person, Am I right? Yes If you are showing your good behavior towards other person the better communication will form. AS like that if you apply the right behavior on different parts of your WCF system, you will be able to influence WCF service in regard to things concurrency, session management, throttling, and transactions. Behaviors can be applied at service, operation, endpoint and contract levels. Few behavior configurations can be done on configuration side and few can be at the source code side.

# Service Behavior

The [Service Behavior] attribute is used to apply behavior at the service level. It allows you to control things such as

* Concurrency
* Throttling
* Transaction
* Session Management
* Thread Behavior

The above can be achieved by setting its properties

|  |  |  |
| --- | --- | --- |
| **Service Behavior Properties** | | |
| **S.No** | **Property** | **What for?** |
| 1 | Address Filter Mode | Allows you to change the Message Filter,This property has three values Any,Exact,Prefix.Used by Dispatcher to identify the correct endpoint responsible to handle incoming messages |
| 2 | Automatic Session ShutDown | Boolean Field, Stops the server from closing the session when all the incoming messages have been processed. By Default it is true.By make it false you are able to control the session life time |
| 3 | Concurrency Mode | This sets if the service runs on single thread or multiple threads |
| 4 | IgnoreExtensionDataObject | Boolean Field. By Default False. IF set to true, Any unknown serialization data is not sent with the message |
| 5 | IncludeExceptionDetailinFaults | Set to True ,if you want to get the unhandled exception sent to client as a SOAP fault. Set to False in a production environment, Set to True in development enviroment |
| 6 | InstanceContextMode | Used to set the lifetime of the service instance, Allowed values are Persession,Percall and Single |
| 7 | MaxItemsinObjectGraph | This sets to Maximum allowed items in the serialized/deserialized object graph.Sometimes you receive an exception when you serialize/deserialize the maximum number of items.Increase this property to match your case |
| 8 | ReleaseServiceInstanceOnTransactionComplete | If set to true,the service object will be released when the active transaction is complete |
| 9 | TransactionAutoCompleteOnSessionClose | Set to true,if you want to mark the active transaction as complete when the session is closed by client without error |
| 10 | TransactionTimeOut | Sometimes a transaction can take more time to complete. You can set a timeout after which the session is considered as aborted and the rollback process is started |

Session management and the Concurrency are really important and most useful properties in Service Behavior. In my upcoming article you can find detailed explanations on Session Management and concurrency, But here we will concentrate only on the Transactions and Throttling.

**What is a Transaction?**

The transaction is a logic unit of work consisting multiple activities that’s needs to be succeed or fail. Yes, you are trying to make an online purchase – dress, your amount got debited from the bank but your ordered dress was not delivered to you. So what did you get from the above example? Bank operations hosted in a separate service and the online purchase service (Ordering the goods) hosted as a other separate service. For the above example online purchase service did not worked well as expected due to the database query error, without knowing it the Bank operations were completed his job perfectly. For this we really need transactions to make sure that either of the operations should succeed or fail. Before we go to the WCF transactions, Let we discuss what exactly the basics behind the transactions?

# Types of Transactions

* Atomic
* Long Running

Atomic transactions are the one which takes less time to finish, For Example updating the relative tables in the database. As I told you in the last paragraph, bank related operations table got updated, but the Purchase related tables were not updated. This we can call it as an Atomic Transactions.

Long Running transactions are the one which takes more time to finish, For Example client sending a request to the service but it takes more days or month to give response to the client. Yes of course till that period of time we should not lock down the resources like SQL server, so this case cannot be handled by Atomic Transactions.

# Transaction Protocols

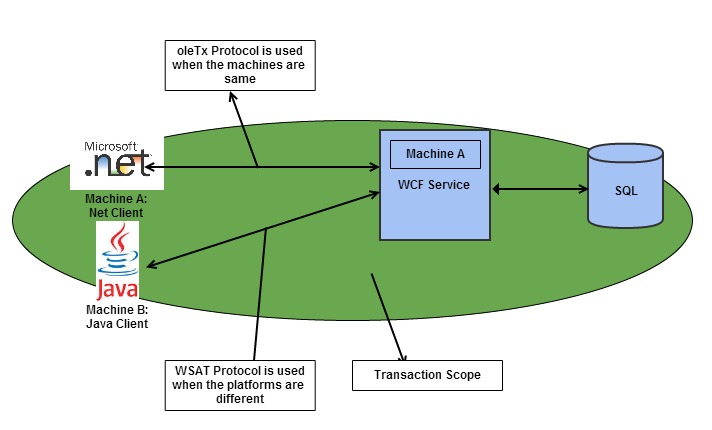
* Lightweight Protocol
* OleTx Protocol
* WSAT Protocol

Lightweight Protocol – It is used when a single application inside an appdomain is working with single RM (SQL server or any component participating in transactions)

OleTx Protocol - It allows Cross AppDomain, Used for Windows-intranet scenario. No cross platform communication allowed, No communication through firewall is allowed

WSAT Protocol – Similar like OleTx, but this will go across firewalls and platforms

Hope the above one is still unclear for you guys, Let me try to explain you in other way, A WCF service calls into SQL server for multiple updates to be done, which are all enlisted as a single transaction. WCF service is called by the clients running on same machine or different windows machine. In this case OleTx protocol is used to maintain the transaction. In the same above situation if one of your client is a Java client ,WSAT protocol will be used since it is an cross platform interoperability.



# WCF Transactions

We shall see here How the Transactions will be achieved using WCF? Following are the steps are really important to configure Transactions.

* Transaction Bindings
* Transaction Flow Option
* TransactionScope(Client and Server)
* Configuring Transaction Modes

# Transaction Bindings

# Before you think about achieving the transaction in WCF, we need to consider the

# binding what you choose, not all Binding normally supports Transaction, The following

# are the binding which supports the Transaction

* NetNamedPipeBinding
* NetTCPBinding
* WSHttpBinding
* WSDualHttpBinding
* WSFederationHttpBinding

# NetNamedPipeBinding and NetTCPBinding support transactions because they deal with

# AppDomains very well, either in the same machine or another machine.

WS bindings not an issue, they implement the Web Services Enhancement standards (WSE) namely WS-Atomic Transaction. The following is the declaration of the binding

<bindings>

<wsHttpBinding>

<binding name="wshttpbind" transactionFlow="true"></binding>

</wsHttpBinding>

</bindings>

# Transaction Flow Option

# This option we should declare in service interface methods. Following are the

# three set of options available

# Transaction Flow option – Not Allowed

# Transaction Flow option – Allowed

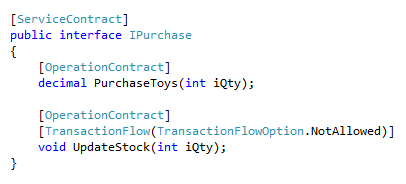
# Transaction Flow option – Mandatory

# Transaction Flow Option – Not Allowed

# The client cannot propagate its transaction into the service operation. No matter if the

# binding supports transaction. Even if the client propagates its transaction, it will be

# simply ignored, no exception will be thrown.



# Transaction Flow Option - Allowed

# The service operation allows the transaction propagation if the client wants to. That is

# the client creates the transaction, binding that you are using also supports

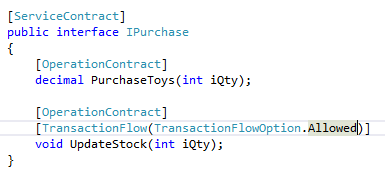
# transaction. If the transaction flow option is allowed in service method, Then the

# Transaction will be propagated to the service,In other words the transaction will not be

# Propagated. What if? When the client propagates the transaction, but the binding in

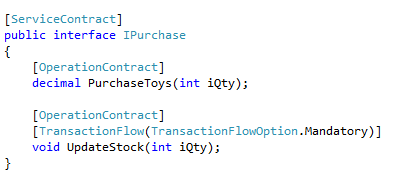
# the service did not support transactions,As a result exception will be thrown while the

# client propagates the transaction



# Transaction Flow Option – Mandatory

The client and the service must use the binding that supports transactions, Any violations the exception will be thrown



# Transaction Scope

Let, we discuss two important attributes we need to be defined in the transaction are TransactionScopeRequired and the TransactionAutoComplete properties in operation behavior.

TransactionScopeRequired –True, it means if the client propagates the transaction to the operation, it will be the one to use in the service side. If the client did not propagates any transaction,it creates own transaction by own and use it. Please look the below source code to know how we should declare in the client and server side.

TScopEnabled.png

TransactionAutoComplete – True, This setting instructs the WCF to auto commit the transaction at the end of code block. It can be configured only at the service level.

# Configuring Transaction Modes

So far we have learned about the usage of transaction attributes, now lets we see different options in configuring transaction modes.

**Use Client’s T or Use your OWN T**

Did you understood the Title **‘***Use Client’s T or Use your OWN T***’**? Hope you couldn’t ☺

Yes,Let me explain it here, The following steps tells that use the client’s transaction if it is propagated or otherwise create your own transaction and use it well.

* Set the transaction Flow attribute to True on both the client and service
* Set the Transaction Flow allowed on the operation contract
* Set the TransactionScopeRequired and TransactionAutoComplete attributes of operation contract to True

TScopEnabled.png

If you still not clear ,please refer the attached source code

**Use only Client’s T**

Use only Client’s T –. Yes the following steps tells that Hello Service, Don’t use or create any transaction, use only the client’s transaction

* Set the transaction Flow attribute to True on both the client and service
* Set the Transaction Flow Mandatory on the operation contract
* Set the TransactionScopeRequired and TransactionAutoComplete attributes of operation contract to True

**Use only Service’s T**

Use only Service’s T – The following steps tells that *Hello* Service, Don’t use other’s transaction ,use your own transaction(Service)

* Set the transaction Flow attribute to True on both the client and service
* Set the Transaction Flow Not Allowed on the operation contract
* Set the TransactionScopeRequired and TransactionAutoComplete attributes of operation contract to True

**Don’t use T**

Don’t use T – The Following steps tells that *Hello* Service, Don’t use any transaction.

* Set the transaction Flow attribute to True on both the client and service
* Set the Transaction Flow Not Allowed on the operation contract
* Set the TransactionScopeRequired and TransactionAutoComplete attributes of operation contract to True

So far we have seen What is transactions? What are all the types of protocols it uses ? What are all the Steps involved in configuring transactions in WCF? How should we configure different transaction modes? Now we shall see what exactly the process involved during the transaction. Transactional manager is a one who is managing the transactions across the process or boundaries. It uses either the OleTx Protocol or the WSAT protocol to manage. Namely Distributed transaction manager(DTC). DTC actually uses the management protocol 2 Phase Commit Protocol to manage the distributed transaction across machines. There are two phases that DTM uses to manage the distributed transaction

1. Prepare
2. Commit

In the Prepare phase ,DTC asks each Resource manager(SQL) to vote on either commit or abort through an RPC call, Then each Resource manager (SQL) gives their votes to DTC saying Yes or No.

In the Commit phase, DTC asks all Resource managers to commit or abort based on the votes received yes or no.

Coordinator Cohort

QUERY TO COMMIT

-------------------------------->

VOTE YES/NO prepare\*/abort\*

<-------------------------------

commit\*/abort\* COMMIT/ROLLBACK

-------------------------------->

ACKNOWLEDGMENT commit\*/abort\*

<--------------------------------

end

But please don’t worry about which transaction manager I need to use, how I need to manage and all, because all these operations are managed by the WCF internally. Please refer the attached code for further details

# Throttling

Service Behavior –Throttling exposes properties that you can control how many instances or sessions are created at the application level. For example the number of service instances creating should not be more than the particular value.

**MaxConcurrentCalls**

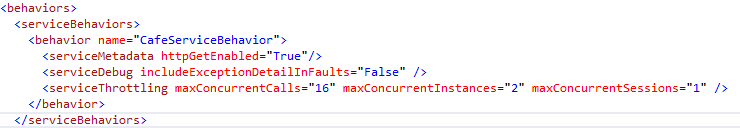
To define the number of concurrent calls that the service will accept. Default value in .net 4.0 is 16 \* processor count. Prior to the .net 4.0 is 16

**MaxConcurrentSessions**

To define the number of session channels that the service will accept. By default in .net 4.0 the default value is 100 \* processor count, prior to .net 4.0 the default value is 10

**MaxConcurrentInstances**

To define the number of service instances will be created. By default in .net 4.0 the default value will be 26, where as in .net 4.0 the default value will be 116 \* processor count



# Attachment

Please refer the folder Transaction for the Transaction sample,refer the folder Throttling for the Throttling samples

# Summary

CTS – Concurrency, Throttling, Transactions are the important topics in service behavior