**WCF Binding Decision Tree**

If you’re a .NET developer, specifically a WCF developer, you know that choosing correct WCF binding for your application is always confusing. This is not only due to wide range of bindings available, but you also need to consider features like security, reliability, transport protocol, legacy application support, in process – out of process communication, intranet vs internet applications etc. Some of the WCF bindings like NetTcpBinding or BasicHttpBinding clearly indicate the transport protocol; however you still need to understand differences between BasicHttpBinding and WsHttpBinding in order to choose between the two. I have been part of large scale, complex application development built specifically using WCF and I know how critical it is to choose appropriate binding which supports all the technical features required in the application.

I am a huge admirer of [Pluralsight](http://www.pluralsight.com/) and they have outstanding courses on WCF. One of such course is [WCF End-to-End](http://www.pluralsight.com/courses/wcf-end-to-end) from [Miguel Castro](http://www.pluralsight.com/author/miguel-castro), in which the author has created a very useful flowchart to choose appropriate WCF binding depending on the features required. I have made few minor changes to it just to avoid overlapping over the arrows.

The diagram shown below does not cover all the bindings available in WCF today; however it covers most commonly used bindings. Feel free to enhance it for additional bindings. I have already uploaded editable diagram [visio file] on my github repository.



Initially I thought to end this article just by embedding the flowchart, but considering number of bindings and their features, I think it would make more sense to provide a short description for each of the step in the flowchart.

* For any service based application, you need to identify the service boundary i.e. whether service will be accessed **inside the firewall only** [for e.g. intranet based application] or outside the firewall [internet based web application]. However just by knowing the service boundary won’t help you to choose appropriate binding. Next, you need to identify the service consumer – client application. If the client application is built using .NET itself, that means you have .NET to .NET communication. In this case it is recommended to use **WsHttpBinding** which is the most advanced HTTP binding. It provides many useful features like message or transport level security, reliable messaging, transaction handling etc.
* Now if you’re building an internet based application and you also need to **support legacy clients** built using Java or .Net 2.0, then you have to use **BasicHttpBinding**. This binding is not as feature rich as WsHttpBinding and you don’t get all the security and reliability features mentioned earlier, but it is the most lightweight binding and doesn’t add any overhead during client - server communication.
* If you’re inside a firewall, you again need to check if your service clients are .NET or not. If they are not .NET, then again you need to jump to the legacy support question and choose appropriate binding [BasicHttpBinding or WsHttpBinding].
* If you’re inside a firewall and all the consuming clients are .NET, next you need to check is whether WCF service and .NET clients are on the same machine. If they are, then **NetNamedPipeBinding** provides the fastest communication between client and server. Remember NetNamedPipeBinding is the best mechanism for interprocess communication which crosses process boundaries on same machine. With NetNamedPipeBinding the security can be achieved only through transport level security as it does not support message level security. However messages are secured as only the same machine can call the service operations.
* If your services and consuming clients are on different machines [common scenario], you need to check if you need to support any queueing mechanism so that clients can be disconnected. If yes, then go for **NetMsmqBinding**.
* If you don’t need queuing support, you should check about message reliability requirement. If you don’t need reliable messaging then go for **NetUdpBinding** otherwise choose **NetTcpBinding**.

I hope that was useful.