**WCF的ABC原则:地址(address)、绑定（binding)与契约 (contract).**

“A”是地址，意味着在哪里（where)。地址定义的是网络消息送达之处，即端点接 收消息之处，客户端必须将消息送到此处。对于HTTP来说，地址类似 [http://myserver/myscrvice/:对于](http://myserver/myscrvice/:%e5%af%b9%e4%ba%8e) TCP,地址类似 net.tcp://myserver:8080/myservice。

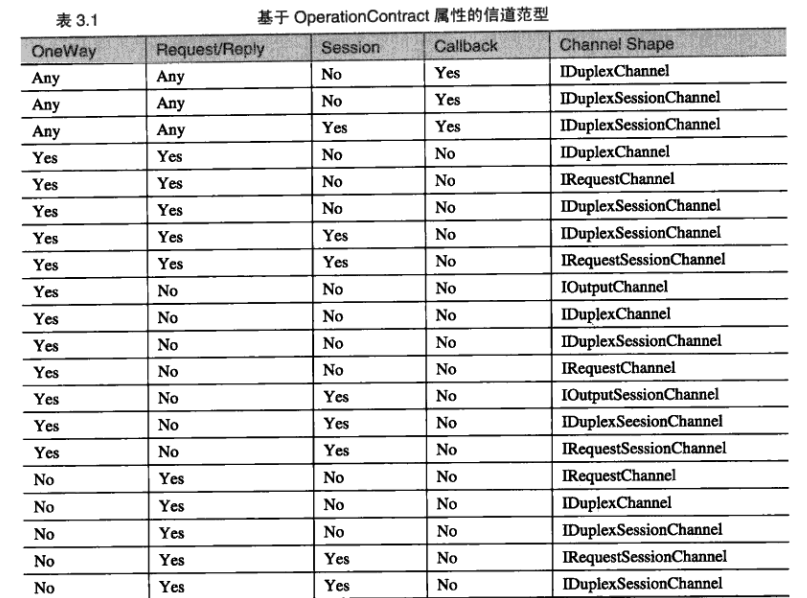
“B”是绑定，意味着怎么做（how）绑定定义的是与端点通信的信道（channel)。 信道是一个所有WCF应用程序传递消息的管道。信道包括一系列绑定元素（binding elements)。最底层的绑定元素是传输（transport),它负责在网络上传递消息。内置的传输包 括HTTP、TCP、命名管道（NamedPipes)、PeerChannel3和MSMQ。在此之上的绑定元素规 定安全（security)和事务（transactions〉。幸运的是，WCF中包含了系统提供的绑定，其信 道已配置安排就绪，使用绑定能节省考虑配置的时间，basicHttpBinding能与2007年前的大 多数Web服务轻松通信。它符合WS-IBP 1.1标准，具有广泛的互搡作性。wsHttpBinding实 现了通用的WS-\*协议，具有安全、可靠和事务化的消息能力。

“C”是契约，意味着内容（what),它定义端点提供的功能或功能集合。契约定义 了端点对外发布的操作（operation)以及这些操作所要求的消息格式。契约的操作映射到 实现端点的类方法（class methods),包括其输入、输出参数的签名（the signature of parameters)。

在网络上传输数据时，一次传输大量数据远胜于多次传输少量数据，这样能减少网路流量并提高响应速度。但这要求每次调用时传递更多的信息，需要使用复杂类型的输入值和输出值。

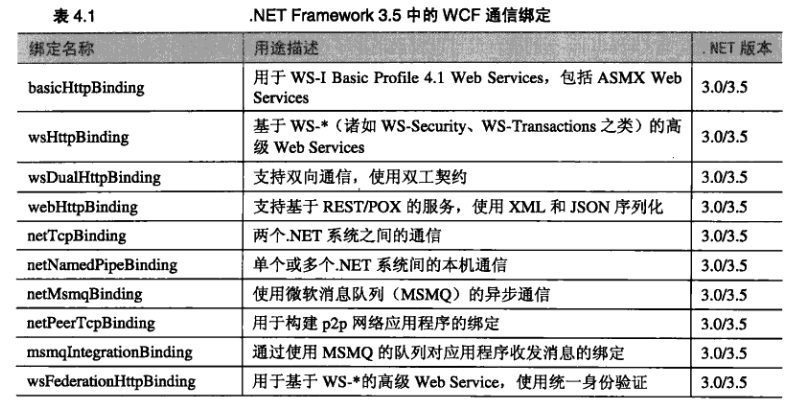
**信道**

开发者不用直接操作信道范型，而是由WCF根据服务OperationContract来选择合适的信道范型。**大多数信道范型都有无会话两种变体**。有会话信道会在客户端和服务器端传递一个标识，这样就可以保持客户端和服务器之间的状态信息。这种方式和ASP.NET进行状态管理的方法类似。WCF虽然没有内置的状态管理功能，但是如果使用会话状态，就可以管理状态了。



在WCF中，**信道监听器**构成了服务器通信的基础。它们负责监听传入的消息，创建信道栈，并为应用程序提供对栈顶的引用。它们从传输信道或其他位于信道栈底层的信道接收消息。大多数开发者不会直接使用信道监听器。而是使用ServiceHost的类来托管服务，这个类会使用一个信道监听器来监听消息。

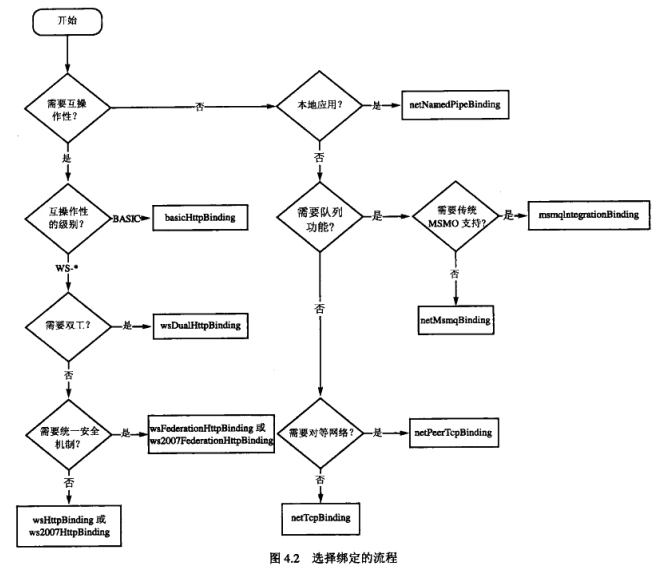
信道栈是由一个或多个处理消息的信道组成的多层次通信栈。**绑定**就是预先配置好的信道栈，它代表了客户端和服务器端之间线路级别上的约定。每个绑定都会制定通信所用的传输，编码以及协议。通过绑定，WCF封装了各种不同的通信场景下所使用的配置。



WCF共有9种预先定义好的绑定，每种绑定都对一种特定的分布计算需求提供了手段。要确定应该为特定应用程序选择哪种绑定，有多种因素需要考虑，包括安全性，互通性，可靠性，性能及事务要求。



每种绑定都支持一个特定的通信场景，如跨机，本机或者使用WebService做互操作通信。在检查绑定之外，用户也需要考察这些场景。还有其他一些场景，如统一安全认证与对等通信等。

****

**名词解释：**

**互操作性**：就[软件](https://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6)而言，互操作性——这条术语用来描述的是不同的[程序](https://zh.wikipedia.org/wiki/%E7%A8%8B%E5%BA%8F)（programs）借助于同一套[交换格式](https://zh.wikipedia.org/w/index.php?title=%E4%BA%A4%E6%8D%A2%E6%A0%BC%E5%BC%8F&action=edit&redlink=1)（exchange formats）来交换数据，读写相同[文件格式](https://zh.wikipedia.org/wiki/%E6%96%87%E4%BB%B6%E6%A0%BC%E5%BC%8F)（file formats）以及采用相同[协议](https://zh.wikipedia.org/wiki/%E5%8D%8F%E8%AE%AE)（protocols）的能力。