# WCF

## WINDOWS COMMUNICATION FOUNDATION

### OVERVIEW OF WCF, MICROSOFTS UNIFIED COMMUNICATION FRAMEWORK FOR .NET APPLICATIONS

Peter R. Egli
peteregli.net

## Contents

# 1. What is WCF (1/3)?

**WCF is a unified communication framework for distributed applications.**

**WCF defines a common programming model and unified API for clients and services to send messages between each other.**

**WCF is the current and future standard for distributed .Net applications.**

**Key characteristics of WCF:**

**→ Service-oriented programming model (SOA):**

**Services are offered on an endpoint.**

**WCF completely separates service hosting, endpoints and services.**

**→ Interoperability with non-WCF web services due to use of SOAP messages:**

**WCF implements many of the WS-* standards.**

**See http://msdn.microsoft.com/en-us/library/ms734776.aspx.**

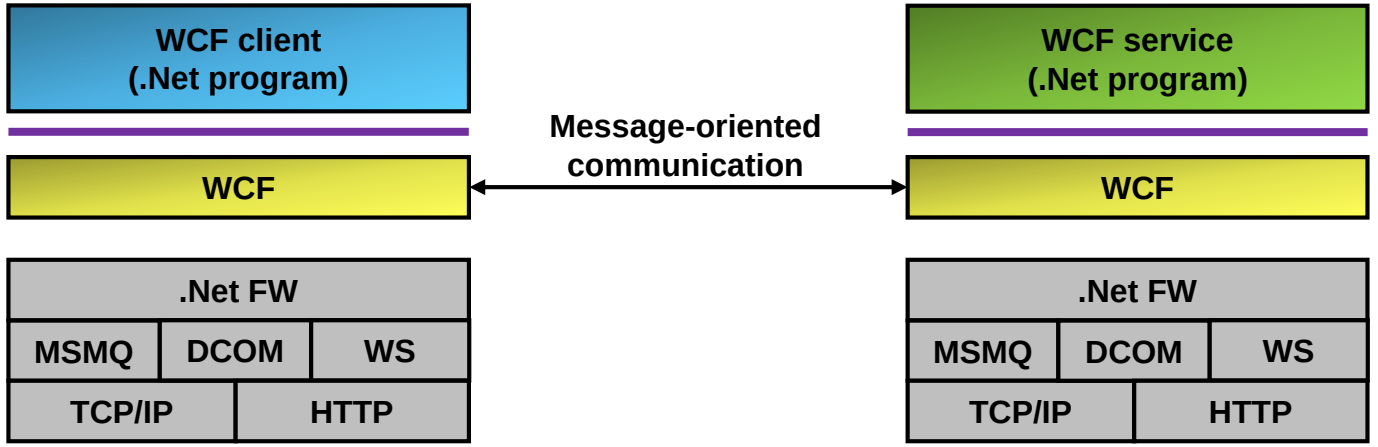
**→ Extensibility:**

**WCF client / server can be configured to interoperate with REST, ATOM-feeds, plain XML or JSON messages (extensions for interfacing to any message-based service).**

# 1. What is WCF (2/3)?

→ It provides a unified API to use various types of communication from an application.
→ WCF sits on top of the .Net Framework.

WCF provides a unified API for various communication models.




**WCF versus web services:**

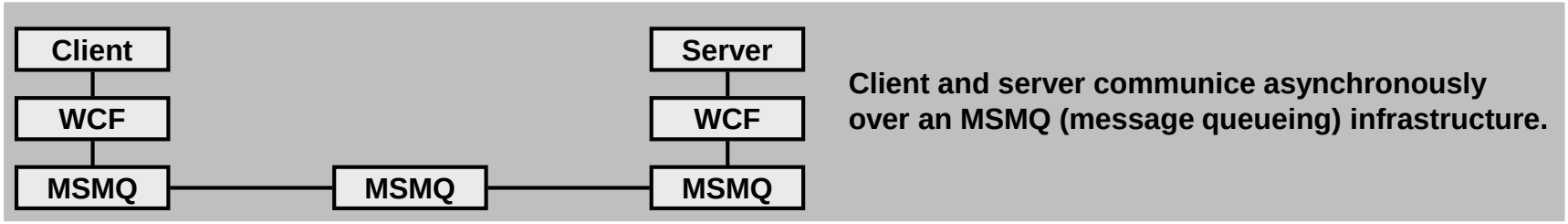
WCF is service-oriented. By separating the service interface (contract) from the transport protocol and message encoding, a (logical) service can be reused in different scenarios.
A developer is forced to specify a service interface on a logical level and then define how the service may be used (transport protocol, message encoding).
These characteristics nicely map to the web service paradigm where a WSDL-file defines the *what* (service interface), the *how* (transport, message encoding) and the *where* (endpoint address).
But: WCF also supports distributed applications that are not web service based.

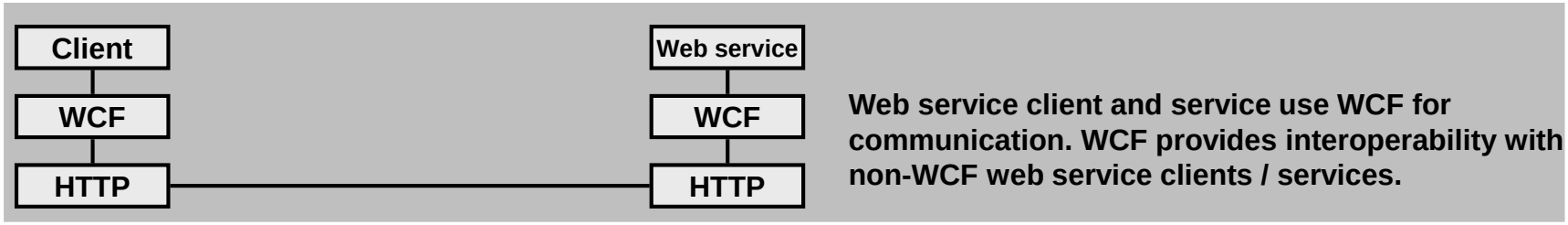
# 1. What is WCF (3/3)?

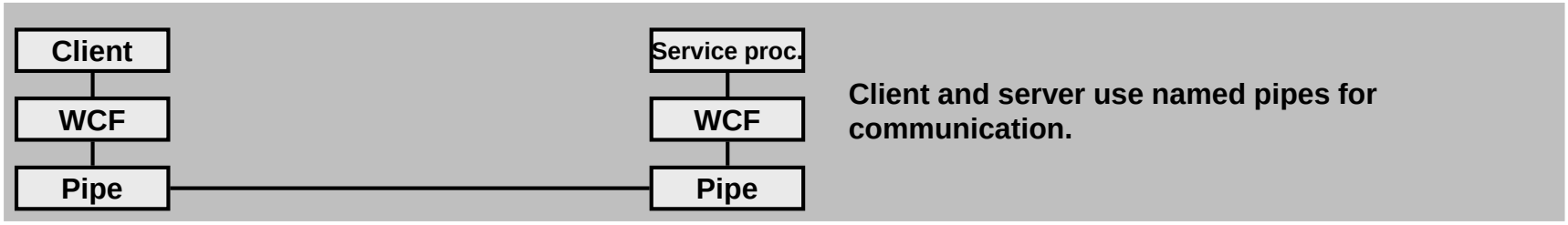
**→ WCF is a generic communication framework. WCF supports WS and other means of communication.**

**→ WCF supports different (message-based) communication protocols.**

## Possible scenarios with WCF:



## 2. WCF‘s ABC

**The core concept of WCF is a service that is provided on an endpoint and accessible over the network through a transport protocol.**
**Thus a WCF service is defined by ABC:**

**A = Address:**
***Where*** **is the service available (the endpoint's URI in case of a web service).**

**B = Binding:**
***How*** **can the service be accessed (what transport protocol is used).**

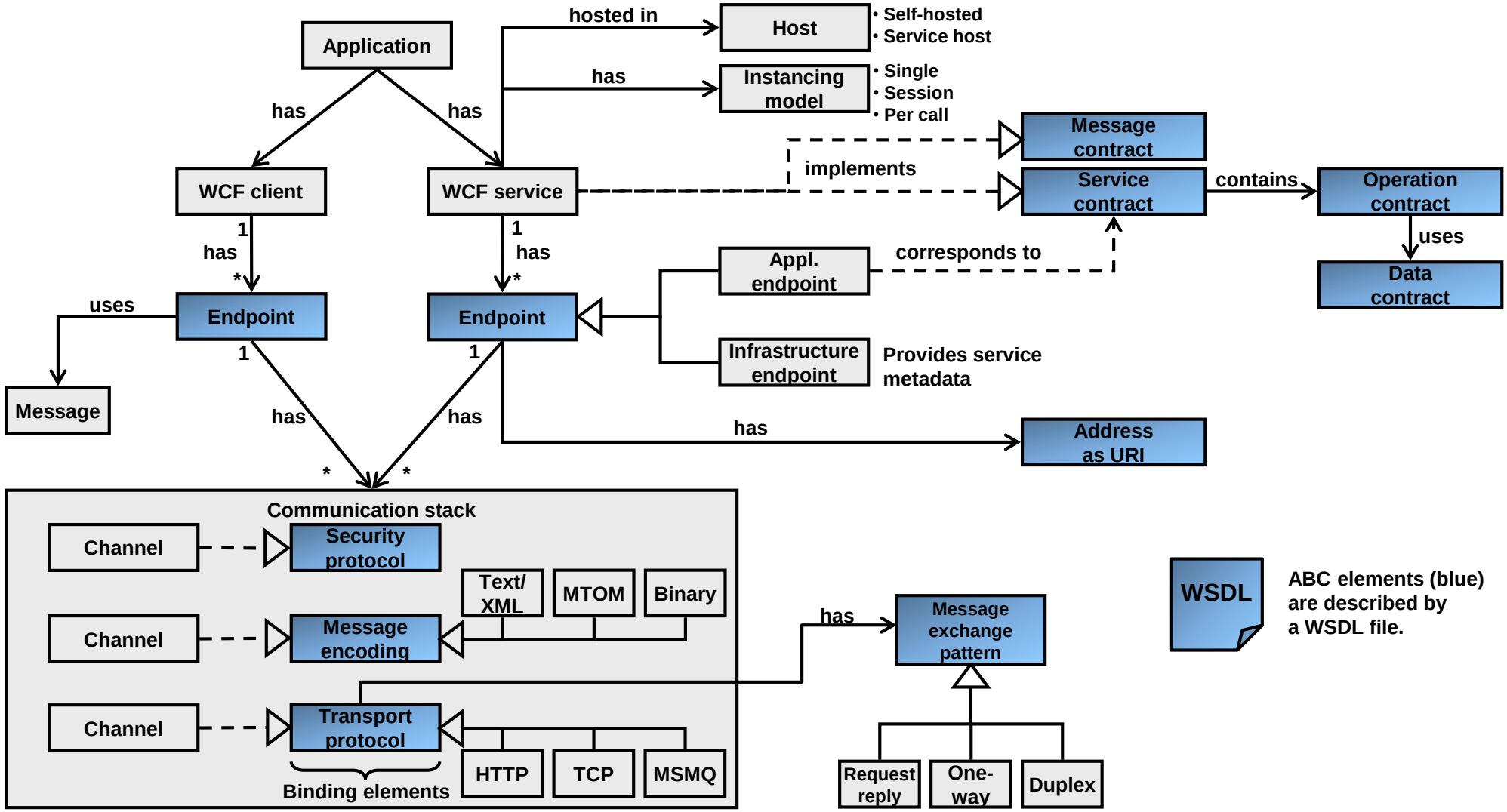
**C = Contract:**
***What*** **does the service interface look like (operations, data-types).**

**The mapping of the ABCs to WSDL:**

| WCF term | Question | WSDL element |
|---|---|---|
| A (Address) | *Where* | <service> including element <endpoint> |
| B (Binding) | *How* | <binding> |
| C (Contract) | *What* | <types><br><interface> |

# 3. WCF model (1/4)

**WCF defines a (consistent) service model with entities and relationships.**



## 3. WCF model (2/4)

**Application:**
**An application has a WCF client and / or WCF service.**

**WCF client:**
**Component with a WCF endpoint for communicating with a WCF service through messages.**

**WCF service:**
**Counterpart to WCF client. Runs in its own process (self-hosted) or in a specific service hosting process (IIS, Windows Activation Service, Windows Service).**

**Endpoint:**
**WCF client and service use an endpoint to connect to each other.**
**Application endpoint: Endpoint on which an application service is exposed / offered.**
**Infrastructure endpoint: Part of WCF system to offer metadata of an application service.**

**Message:**
**Unit of data exchange between WCF client and service. WCF is strictly message-based.**

**Address:**
**Physical address comprising hostname, port number and service name (=URI). The application service is accessible under such an address.**
**Example.: http://localhost:8000/HSZ-TWSMW/DateTimeService**

## 3. WCF model (3/4)

**Service contract:**
**Set of operations which define the application service. The service contract is implemented by an interface in a .Net language (e.g. C# interface).**

**Operation contract:**
**Defines the signature (parameters and return type) of an individual operation of a service.**

**Message contract (SOAP message layout):**
**Describes the format of a message (which information elements should go into the header, which should go into the body, level of security to be applied to message etc.).**

**Instancing model:**
**Defines how the service is instantiated: Singleton, Session, Single-Call.**

**Binding (element):**
**Defines how an endpoint communicates with its peer endpoint. Binding elements define individual aspects of the communication, basically the transport protocol (raw TCP, HTTP), message encoding (text, binary) and the security.**

**Channel:**
**Concrete implementation of a binding element (binding element = configuration, channel = implementation).**

## 3. WCF model (4/4)

**Transport protocol (transport binding):**
**Defines the protocol used to transfer messages "over the wire" (wire protocol). A common transport protocol is HTTP (HTTP over TCP).**

**Message encoding (binding element):**
**Defines the formatting of messages before they are sent over the wire (using the transport protocol). Common message encodings are text/XML (SOAP) or MTOM (Message Transmission Optimized Mechanism) for efficient transfer of binary data.**

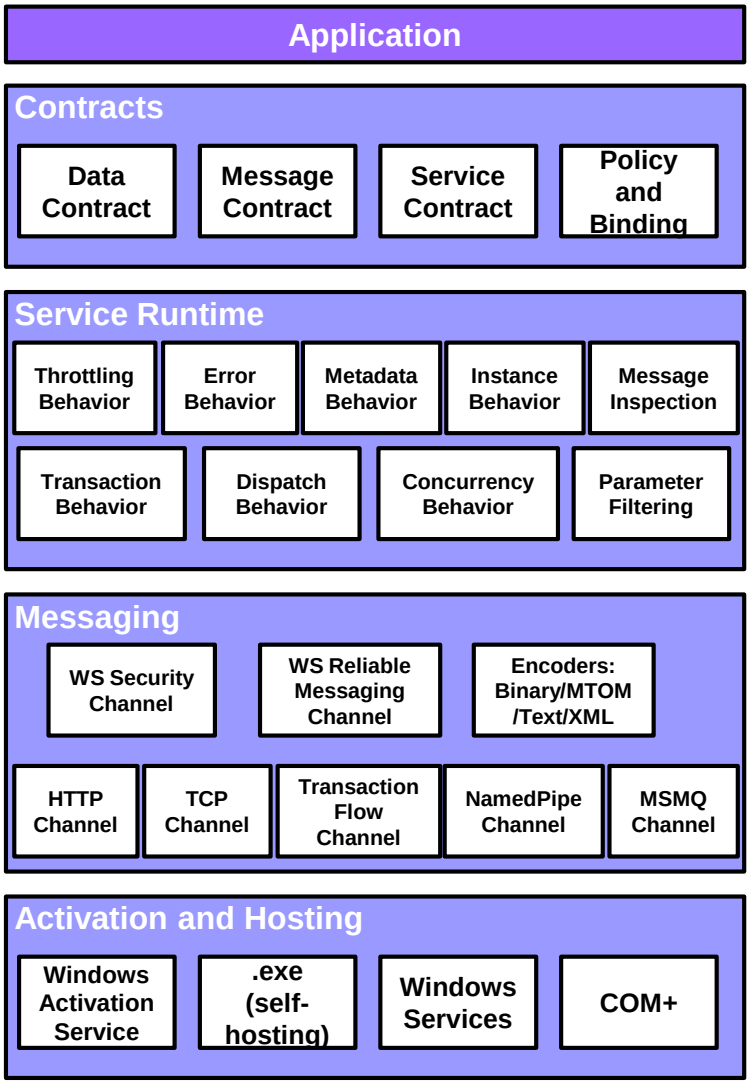
**Security protocol (security binding):**
**Defines what security functions (authentication, encryption) are applied to messages.**

**Message exchange pattern:**
**Defines how messages are exchanged (request-reply, one-way, duplex).**

**Communication stack:**
**Comprises different binding elements, at a minimum a transport binding element and message encoding binding element.**

**Host:**
**Runtime environment for a service. Either self-hosted (specific process for service) or a general hosting process, e.g. IIS, WAS or Windows service.**

# 4. The 4 layers of the WCF architecture




**Contracts define the service (as XSD):**
Data contract → Definition of complex types (structures, classes, generics).
Service contract (C) → Definition of operation signatures.
Message contract (C) → Layout of (SOAP) messages.
Policies and binding (B) → Definition of transport protocol (TCP, HTTP) and message encoding (text, binary); security functions used (if any).

**Service runtime contains runtime behavior (functions of service host):**
Examples: Error behavior, throttling message reception, delivery of service metadata to the outside world.

**Messaging layer with channels (channel = implementation of binding element):**
- Security (authentication & encryption) & reliability (reliable messaging) channels.
- Message encoding channels (text, binary, MTOT, XML).
- Transport channels (TCP, HTTP).

All used channels comprise the channel stack.

Channels, service runtime behaviors and service itself (modules implementing the contracts) are run in a host process. Services can be:
a. Self-hosted (run in their own process).
b. Hosted by an external agent (WAS, IIS, Windows services).

# 5. WCF programming model (classes and interfaces)

**The ABC elements of WCF are offered as specific class libraries.**

| WCF term | Corresponding class library (namespace) |
|---|---|
| A (Address) | System.Uri |
| B (Binding) | System.ServiceModel<br>E.g.<br>BasicHttpBinding (SOAP, non-secure, interoperable, non-duplex)<br>WebHttpBinding (REST-style binding, i.e. non-SOAP) |
| C (Contract) | Interfaces / classes annotated with System.ServiceModel attributes:<br>[OperationContract]<br>[ServiceContract]<br>[MessageContract]<br><br>Data contract (definitions of types used in operation contracts):<br>[DataContract] (System.Runtime.Serialization) |
| E (Endpoint) | System.ServiceModel.ServiceEndpoint |

# 6. WCF address (the 'A' in 'ABC')

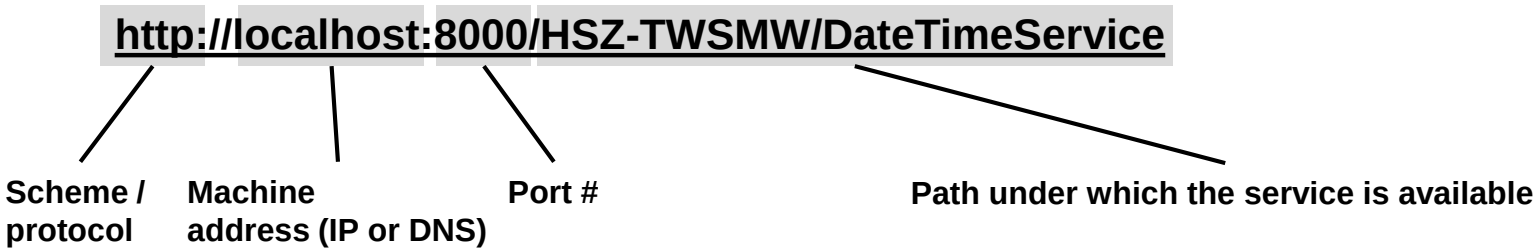
**The WCF address defines *where* a web service (endpoint) is accessible.**
**WCF models an address as an endpoint reference (EPR) as per the WS-Addressing standard.**

## WS-Addressing EPR schema:

```
<wsa:EndpointReference>
    <wsa:Address>xs:anyURI</wsa:Address>                                          URI
    <wsa:ReferenceProperties>... </wsa:ReferenceProperties> ?                     Properties to identify the endpoint
    <wsa:ReferenceParameters>... </wsa:ReferenceParameters> ?                     Parameters associated with an endpoint
    <wsa:PortType>xs:QName</wsa:PortType> ?                                       Endpoint type (WSDL 1.0 portType, WSDL 2.0 interface)
    <wsa:ServiceName PortName="xs:NCName"?>xs:QName</wsa:ServiceName> ?           Link to WSDL service element containing the endpoint descr.
    <wsp:Policy> ... </wsp:Policy>*                                               Security settings of endpoint
</wsa:EndpointReference>
```

WS-addressing EPR see http://www.w3.org/Submission/ws-addressing/#_Toc77464317

## Example endpoint address URI:




## Endpoint address class `System.ServiceModel.EndpointAddress`:

`EndpointAddress.Uri` → URI
`EndpointAddress.Headers` → Reference properties and parameters
`EndpointAddress.Identity` → Security settings

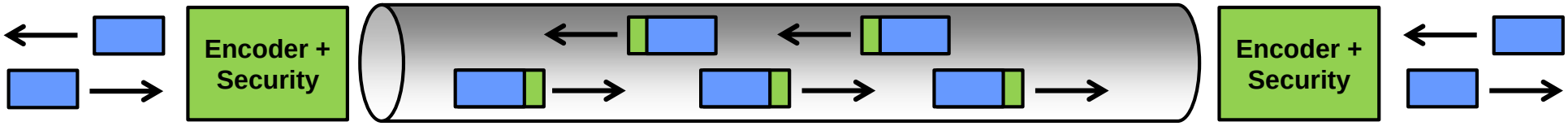
# 7. WCF binding (the 'B' in 'ABC') (1/2)

**The binding defines *how* a web service endpoint is accessed.**
**A binding contains the following elements:**

**1. Transport protocol:**
**→ Underlying transport protocol to use when interacting with the web service.**
**→ Examples: TCP, HTTP, MSMQ.**

**2. Message encoding:**
**→ Definition of the message encoding.**
**→ Examples: Text/XML (SOAP), binary, MTOM (Message Transfer Optimized Mechanism).**

**3. Security / reliability settings:**
**→ Message security settings (e.g. encryption and authentication of message).**
**→ Transport security (e.g. encryption of transport connection).**



# 7. WCF binding (the 'B' in 'ABC') (2/2)

## WCF provides the following bindings:

| Binding | Interoperability | Security | Session | Transactions | Duplex | Encoding |
|---|---|---|---|---|---|---|
| BasicHttpBinding | WS-I Basic Profile | N, T, M, m | N | N | No | Text, MTOM |
| WSHttpBinding | WS-* standards | T, M, m | N, RS, SS | N, Yes | No | Text, MTOM |
| WSDualHttpBinding | WS-* standards | M, m | RS, SS | N, Yes | Yes | Text, MTOM |
| WSFederationHttpBinding | WS-Federation | N, M, m | RS, SS | N, Yes | No | Text, MTOM |
| NetTcpBinding | .NET | T, M, m, N | TS, RS, SS | N, Yes | Yes | Binary |
| NetNamedPipeBinding | .NET | T, N | N, TS | N, Yes | Yes | Binary |
| NetMsmqBinding | .NET (WCF) | M, T, N | N, TS | N, Yes | No | Binary |
| NetPeerTcpBinding | .NET | T | N | N | Yes | N/A |
| MsmqIntegrationBinding | MSMQ | T | N | N, Yes | No | MSMQ |
| BasicHttpContextBinding | WS-I Basic Profile | N, T, M, m | N | N | No | Text, MTOM |
| NetTcpContextBinding | .NET | N, T, M, m | T, RS, SS | N, Yes | Yes | Binary |
| WSHttpContextBinding | WS-* standards | T, M, m | N, RS, SS | N, Yes | No | Text, MTOM |
| WebHttpBinding | HTTP (REST) | N | N | N | No | POX |

**More details on WCF bindings see http://msdn.microsoft.com/en-us/library/ms730879.aspx.**

Key:

| | | | | |
|---|---|---|---|---|
| N: | None | RS: | Reliable Session (WS-ReliableMessaging) | POX: Plain Old XML |
| T: | Transport | SS: | Security Session | |
| M: | Message | TS: | Transport Session | |
| m: | mixed | | | |

# 8. WCF contract (the 'C' in 'ABC')

**WCF interfaces are called “contracts” (both client and server must comply with the contract). Contracts describe operations, data structures and messages.**

**A service contract defines:**

- **a. Grouping of operations in a service → .Net attribute [ServiceContract]**
- **b. Signature of operations → .Net attribute [OperationContract]**
- **c. Data types of operations → .Net attribute [DataContract]**

**From contract to code (and back):**

**The utility SvcUtil.exe may be used to map between abstract and interoperable service definitions (WSDL documents) and programmatic definitions of interfaces (C# code). SvcUtil.exe either retrieves the service definition from a WSDL file on disk using the DISCO protocol (Microsoft proprietary WS discovery protocol) or directly from a running service instance through the standard WS-MetaDataExchange protocol.**

# 9. WCF service hosting

**A service host is the process that runs (executes) the service(s).**
**WCF provides different possibilities to host a service.**

**A. Self-hosted service:**
**The service runs in an application process that the developer created.**
**The lifecycle (when is service opened and closed) is controlled by the service itself.**

**B. Standard Windows hosting process:**
**.Net and Windows offer system processes that are designed to host services.**
***a. IIS (Internet Information Services):***
**→ Only HTTP transport supported (IIS is a web server).**

***b. WAS (Windows Activation Services):***
**→ New process activation framework on which IIS 7.0 is based.**
**→ Supports HTTP, TCP, and pipe transports.**

***c. Windows services:***
**→ Standard Windows execution with security etc.**
**→ Lifecycle of service controlled by service startup configuration.**

**More information and a comparison of WCF hosting options see http://msdn.microsoft.com/en-us/library/ms730158.aspx.**

# 10. Steps for creating and running a WCF service (1/4)

**The following steps outline how to create a web service in Visual Studio 2010.**

**1. Visual Studio 2010 project selection (1/2):**

**The hosting environment of a web service defines the project template to be used.**

***a. Visual C# / WCF / WCF Service Library project:***

**→ Output: Service DLL (Dynamic Link Library).**
**→ To be run in WAS.**
**→ Files: IService1.cs with [ServiceContract] attributes (service interface = contract).**
**Service1.cs (service implementation of interface).**
**App.config (configuration file with transport binding and other settings).**

***b. Visual C# / WCF / WCF Service Application project:***

**→ Output: Service DLL.**
**→ To be run in IIS.**
**→ Files: IService1.cs with [ServiceContract] attributes.**
**Service1.cs (service implementation).**
**Web.config (configuration file with HTTP-transport binding and other settings).**

# 10. Steps for creating and running a WCF service (2/4)

**The following steps outline how to create a web service in Visual Studio 2010.**

**1. Visual Studio 2010 project selection (2/2):**

***c. Self-hosted:***

**→ Standard C# application project.**

***d. Managed Windows service (formerly called NT service):***

**→ Windows service project.**

**→ Implementing class has to inherit from `ServiceBase` (base class for Windows service) as well as from a WCF service interface.**

# 10. Steps for creating and running a WCF service (3/4)

## 2. Interface definition (service contract as C# interface or C# class):

**Define the interface of the web service with a .Net interface.**

**Example interface contract definition in C# (containing operation contracts):**

```
[ServiceContract(Namespace = "http://indigoo.WSMW")]
public interface IMyInterface
{
  [OperationContract]
  string MyFunction();
  [OperationContract]
  int MyOtherFunction();
}
```

## 3. Service contract implementation (C# class implementing the interface):

**Create .Net class that implements the web service interface.**

```
public class MyService : IMyInterface
{
  public string MyFunction() {...}
  ...
}
```

# 10. Steps for creating and running a WCF service (4/4)

## 4. Deploy and run the service:

**The final step is to deploy the service assembly (DLL) into the hosting environment or to start it in case of self-hosting.**

**For self-hosting services, the following code needs to be added to the service (definition of endpoint with address and binding):**

```
selfHost.AddServiceEndpoint(typeof(MyService), new BasicHttpBinding(), "MyService");
```

**Finally the service can be started in the WCF self-hosting environment as follows:**

```
selfHost.Open();
```

# 11. WCF configuration versus programming (1/2)

**WCF client and service can be defined by configuration (XML file, declarative) or programmatically (.Net classes).**

**Good practice:**

**For improved reusability of web services in different environments, contract ('C') and address ('A') as well as transport binding ('B') should be separated.**

**→ Define interface and implementation programmatically (C#) without transport and message encodings.**

**→ Definition of transport binding, address and message encoding in configuration file.**

**Declarative service configuration:**

**Major sections of a service configuration (App.config or Web.config)) file are:**

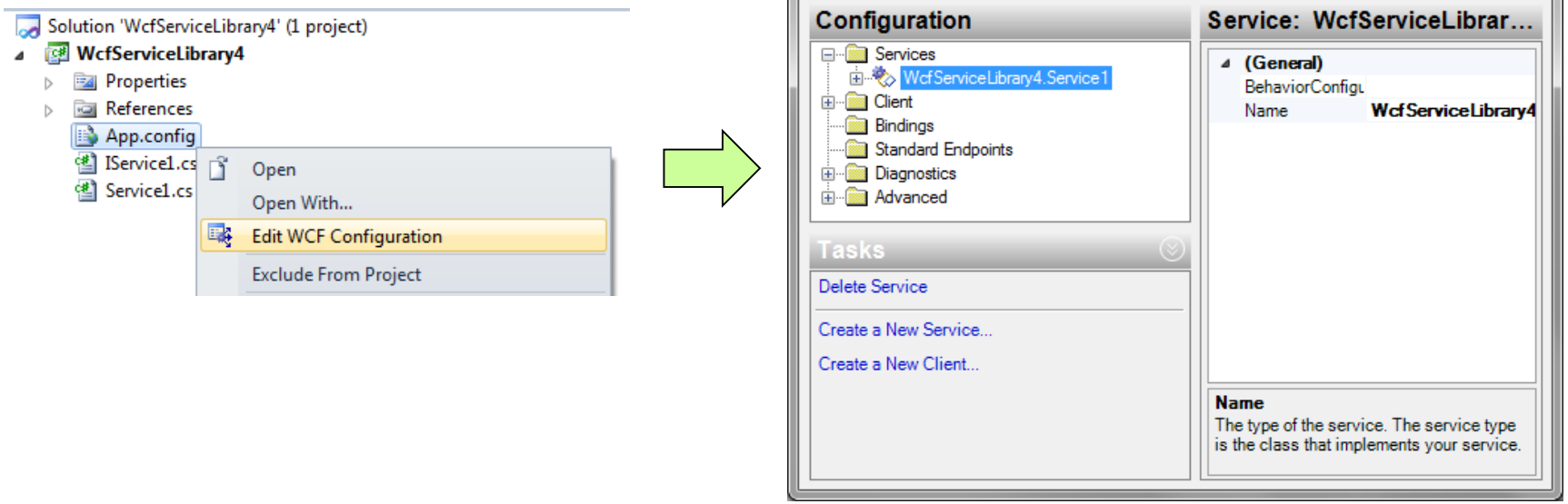
```
<system.ServiceModel>
  <services>
    <service>
      <endpoint/>
    </service>
  </services>
  <bindings>
    <!-- Specify one or more of the system-provided binding elements, for example, <basicHttpBinding> -->
    <!-- Alternatively, <customBinding> elements. -->
    <binding> <!-- For example, a <BasicHttpBinding> element. --></binding>
  </bindings>
  <behaviors> <!-- One or more of the system-provided or custom behavior elements. -->
    <behavior> <!-- For example, a <throttling> element. --> </behavior>
  </behaviors>
</system.ServiceModel>
```

# 11. WCF configuration versus programming (2/2)

## Graphical configuration of service with SvcConfigEditor.exe:

**The service can be configured graphically with the utility SvcConfigEditor.exe (path 'Microsoft SDKs\Windows\v7.0A\Bin\SvcConfigEditor.exe).**

**SvcConfigEditor.exe can also be launched through the App.config / Web.config files' context menu:**



# 12. WCF and SOAP messages

**Usually WCF hides the mapping of operations (operation contracts, data contracts) to SOAP messages (WCF bindings that use SOAP messages automatically map operations into SOAP messages).**
**If necessary, WCF MessageContracts provide complete control over the structure of the exchanged SOAP messages (similar concept to JAX-WS @WebServiceProvider annotations).**

**MessageContracts define a class that can be used as argument type in operations. In MessageContracts, the class members can be mapped to the SOAP header or body as follows:**

```
namespace WSMW_WCF {
  [ServiceContract]
  interface IHelloWorld {
    [OperationContract]
    HelloResponseMessage SayHello(String helloMsg);
  }
  [MessageContract]
  public class HelloResponseMessage {
    private string localResponse = "...";
    private string extra = "...";
    [MessageBodyMember]
    public string Response {
      get { return localResponse; }
      set { localResponse = value; }
    }
    [MessageHeader]
    public string ExtraValues {
      get { return extra; }
      set { this.extra = value; }
   }
  }
}
```