.NET-Networking 2
Windows Communication Foundation

Proseminar *Objektorientiertes Programmieren mit .NET und C#*

Fabian Raab

Institut für Informatik
Software & Systems Engineering

Agenda

- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

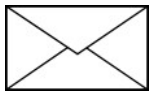
Grundproblem

Client

```
class Application {
  // ...
  service.func("Hello");
}
```

Netzwerk

?



Server

```
class Service {
  void func(string arg)
  {...}
}
```

Lösung durch die WCF

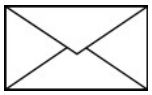
Client

```
class Application {
  // ...
  proxy.func("Hello");
}
```

```
class Proxy {
  void func(string arg)
  {...}
}
```

Netzwerk

Kanal



Server

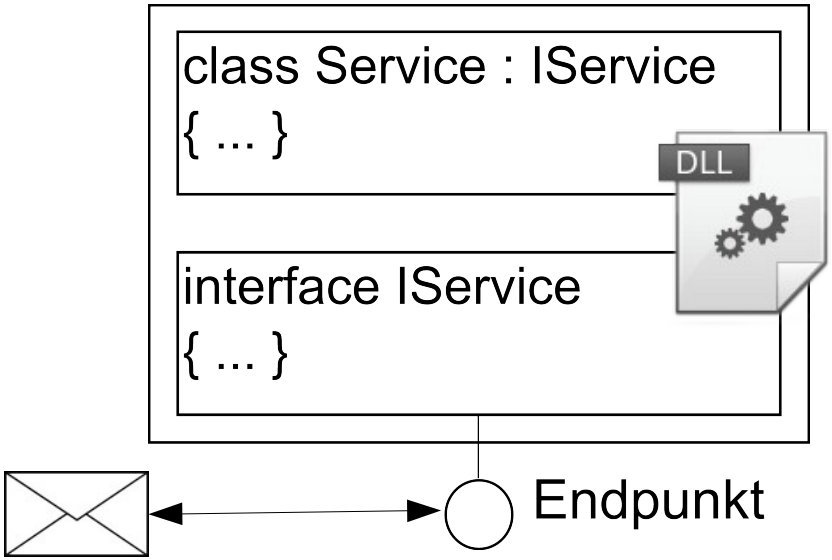
```
class Service {
  void func(string arg)
  {...}
}
```

WCF-Service

Agenda

- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Bestandteile des Services



Implementierung & veröffentlichte Schnittstelle (Library)

Service Host Application

Konfigurationsdatei (app.config)

Ausführbare .NET Andwendung


```
using System;
using System.Text;
using System.ServiceModel;
using EchoLibrary;
namespace ConsoleEchoService
{
  class Program
  {
    static void Main(string[] args)
    {
      using (ServiceHost selfHost = new ServiceHost(typeof(Echo)))
      {
          selfHost.Open();
          Console.WriteLine("WCF Service is ready.");
          Console.ReadLine(); // do not end the service
      }
    }
  }
}
```

WCF Assembly

Unsere Service Library

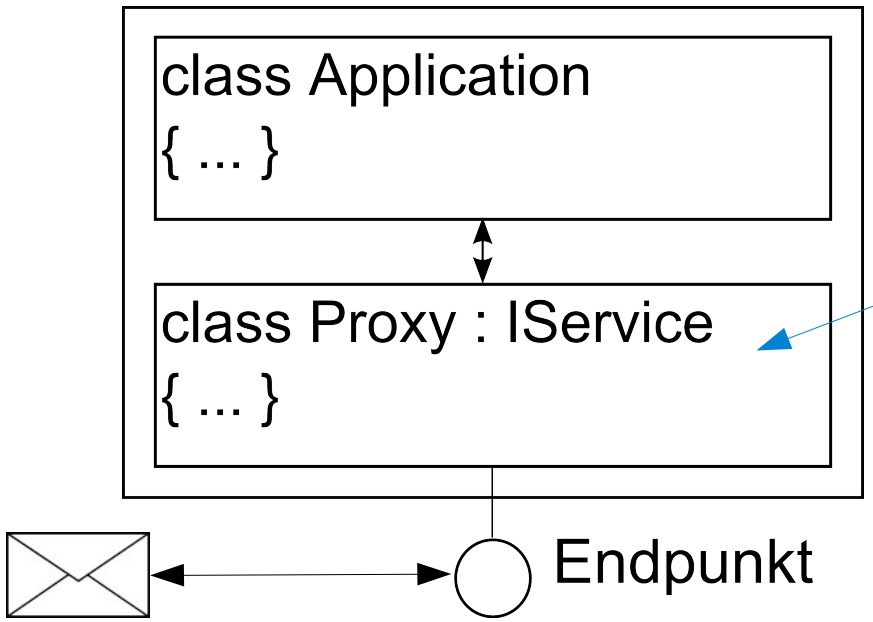
Neue Instanz des Services

Erlaubt Verbindungen

Agenda

- Grundproblem
- Bestandteile des Services
- Bestandteile des Clients
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Bestandteile des Clients

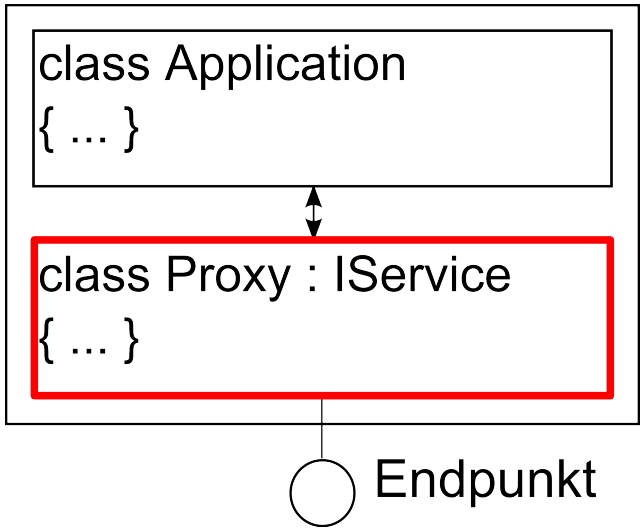


Programm & Proxy Class

Aus Metadaten mit dem *Service Metadata Utility Tool (svcutil.exe)* generiert

Konfigurationsdatei (app.config)

Proxy Class (Auszug)

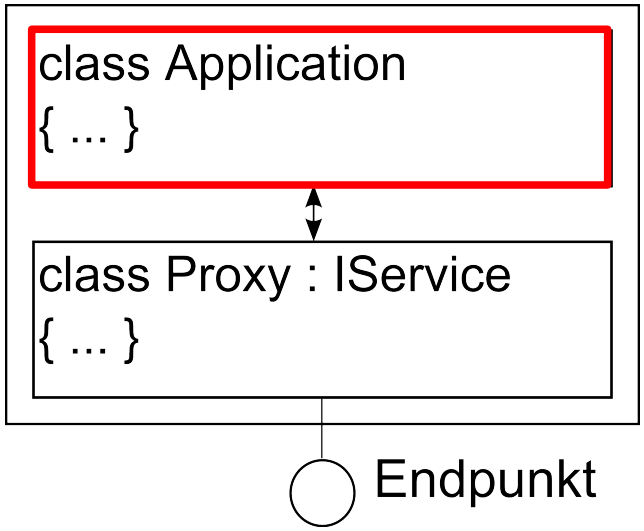



```
public partial class EchoClient :
System.ServiceModel.ClientBase<IEcho>, IEcho
{

    public EchoClient()
    {}
    // [...](many Contructors)

    public string echoString(string echoMsg)
    {
        return base.Channel.echoString(echoMsg);
    }
}
```

Konstruktoren

Lokal definierte Operation

Implementierung Client



Verbindung mithilfe des Proxys erstellen

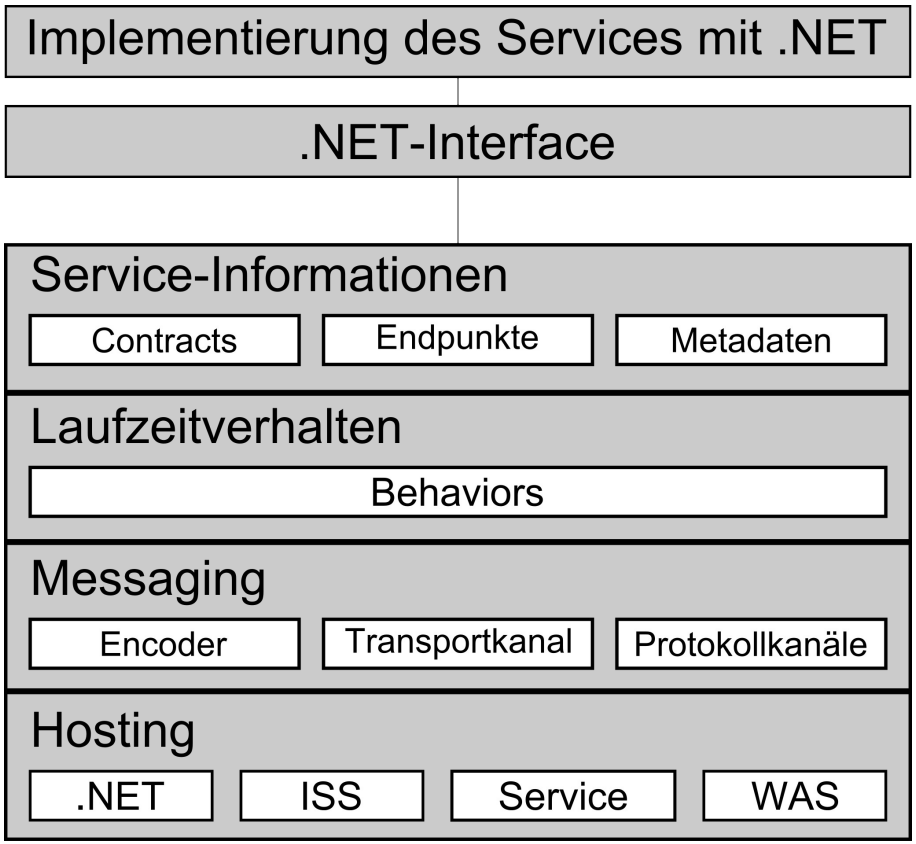
```
using System;
using System.Text;

namespace ConsoleEchoClient
{
    class Program
    {
        static void Main(string[] args)
        {
            EchoClient echoClient = new EchoClient();
            Console.WriteLine("Connecton established.");

            Console.WriteLine("Server response: \"{0}\"",
                        echoClient.echoString("Hello World!"));
            echoClient.Close();
        }
    }
}
```

Senden der Nachricht & Ausgabe der Antwort

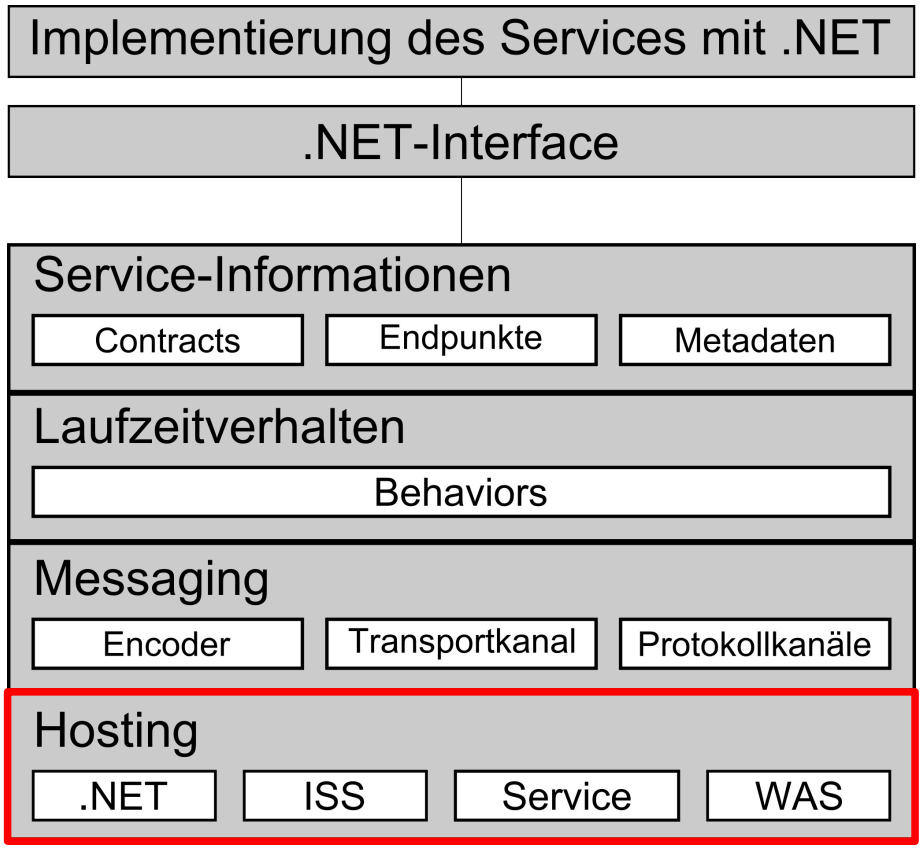
Agenda

- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Architektur der WCF



Hosting

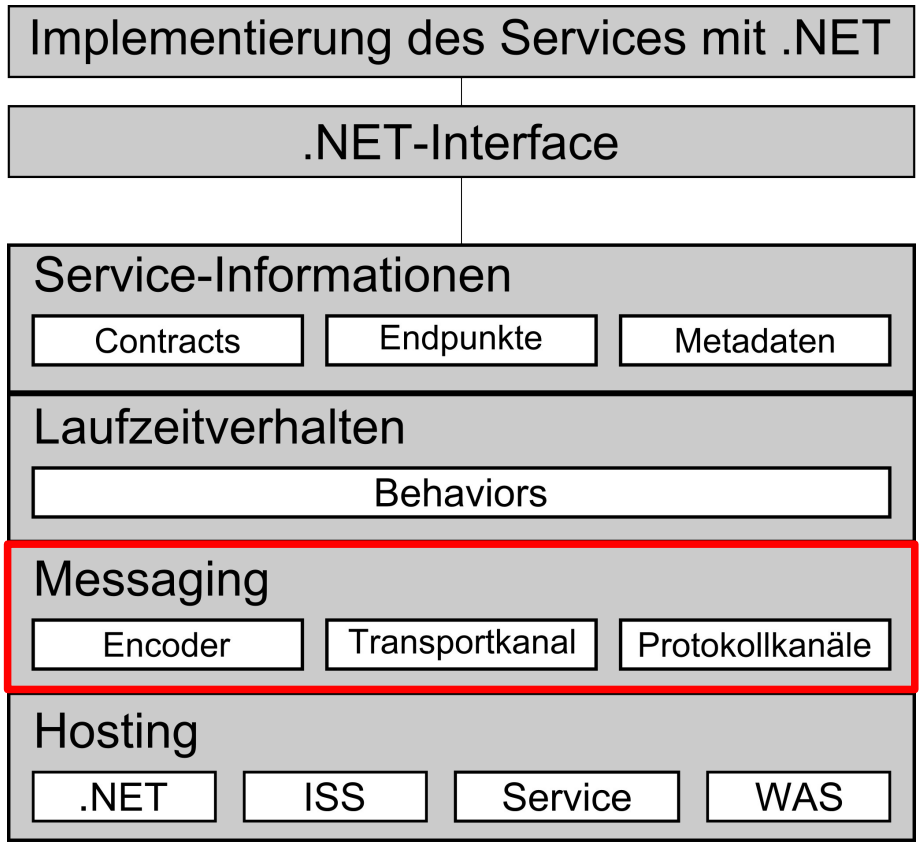


- Stellt den eigentlichen **Service bereit**
- Können auf folgende Arten **gehostet** werdenden:
  - Ausführbare .NET-Anwendung
  - Internet Information Service (ISS)
  - Windows Dienst
  - Windows Activation Service (WAS)

Agenda

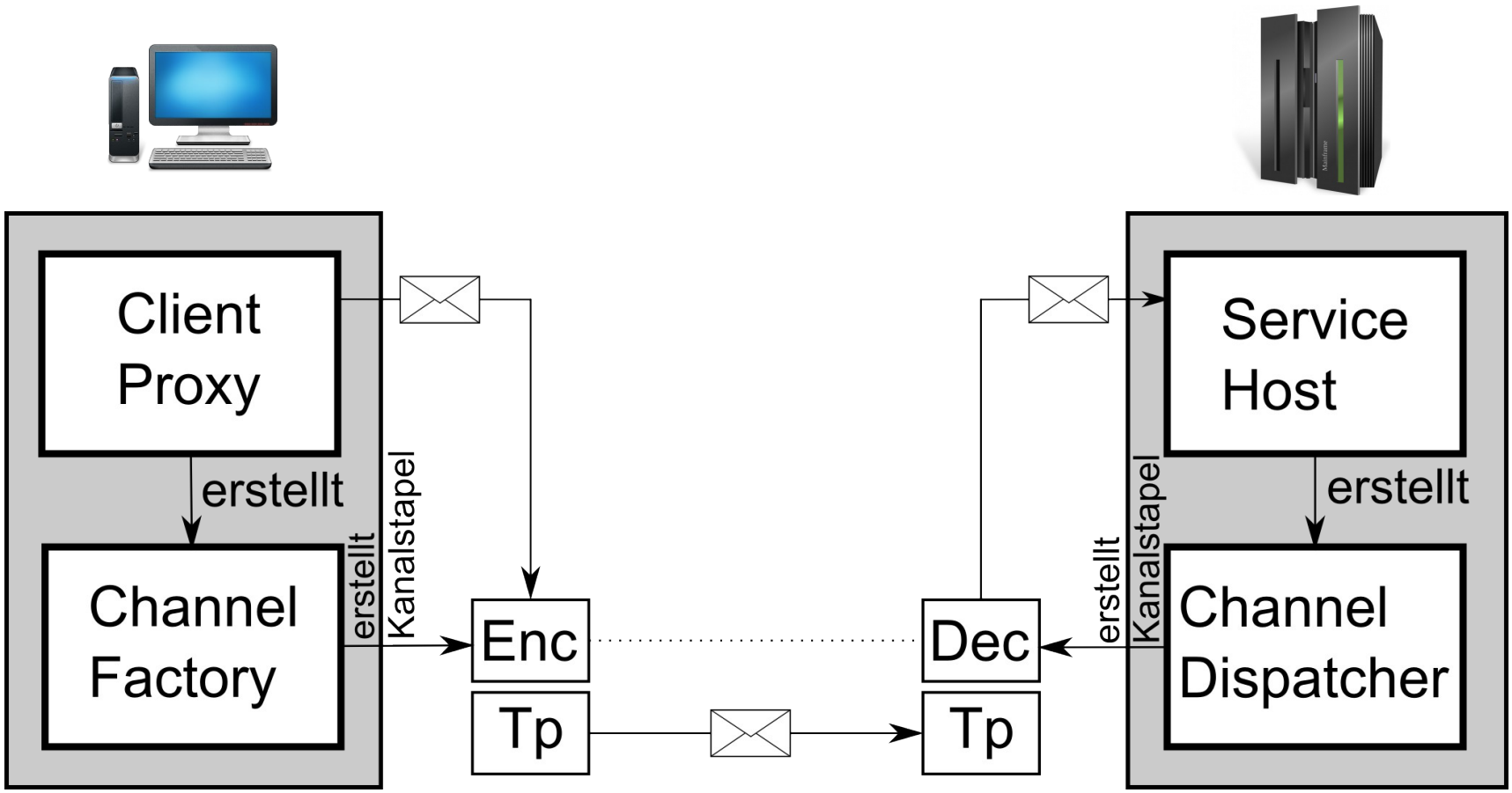
- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Messaging



- Vorgang des **Übertragens** von Nachrichten
- Wichtigster interoperable Web-Services Standard: **SOAP**
- Message Exchange Pattern
  - One-Way
  - Request-Reply
  - Duplex

Senden einer Nachricht (Client → Server)

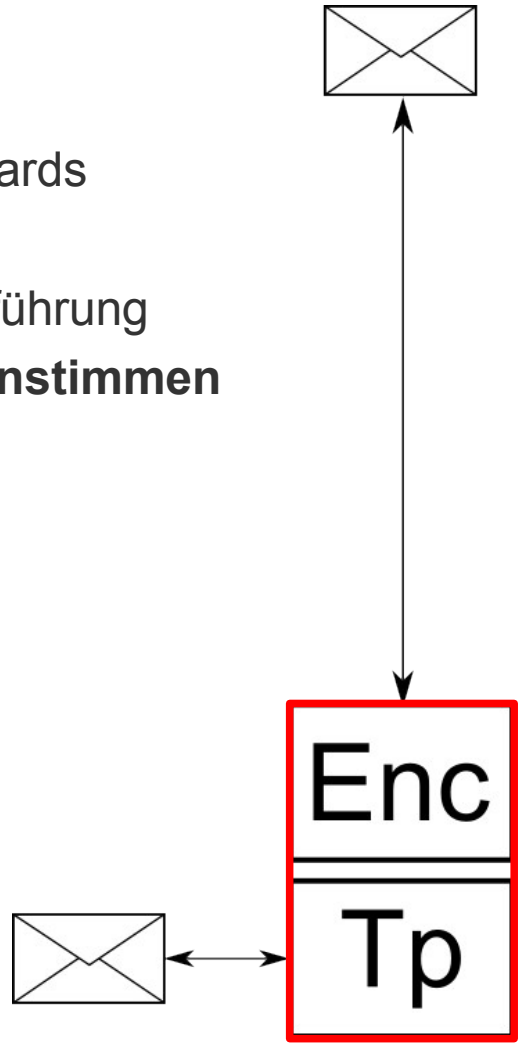


Transportprotokolle und Kodierungen

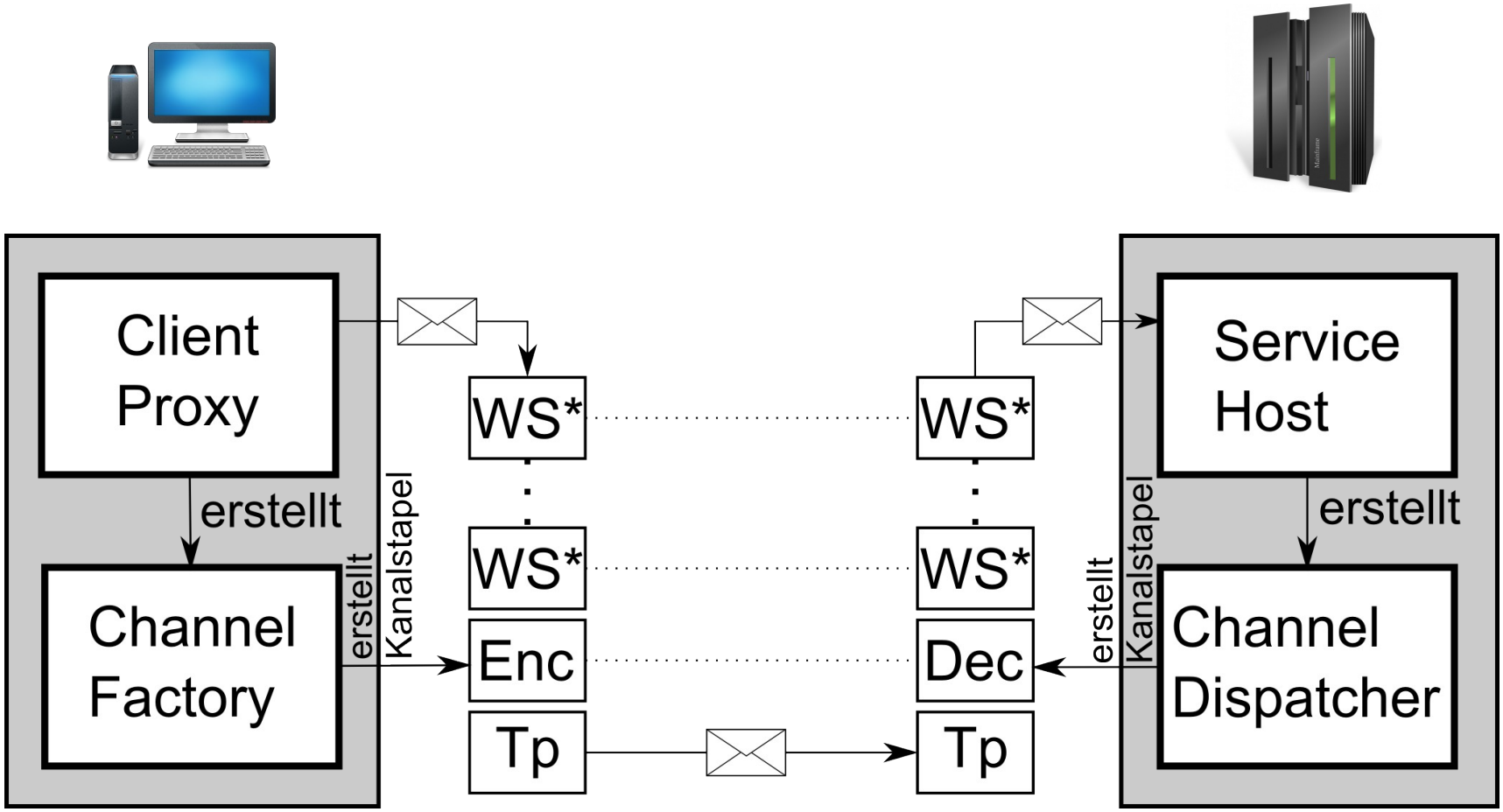
ProtokollPS	Kodierung	Vorteile	Nachteile
HTTP / HTTPS	XML nach WSDL	Maximale Interoperabilität	langsam & Kein Duplex bei HTTPS
TCP / Named Pipes	Binär	Sehr schnell	Client muss Kodierung kennen
Microsoft Message Queuing (MSMQ)			

WS-I Basic Profile

- Problem **viele unterschiedliche** Versionen der Standards
- Web Services Interoperability Organinzation (**WS-I**)
- **Spezifiziert** genau alle Standards in Version und Ausführung
- So müssen nur noch Version des Basic Profile **übereinstimmen**

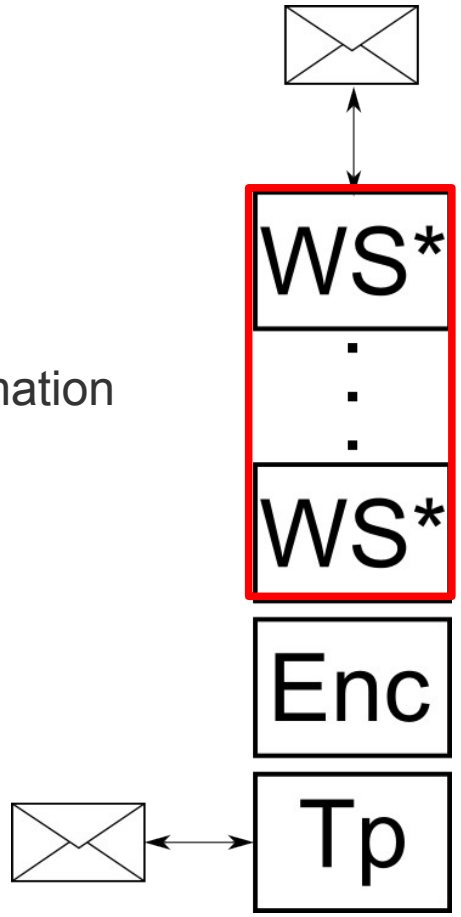


Senden einer Nachricht (Client → Server)



Web Service Enhancements (WS*)

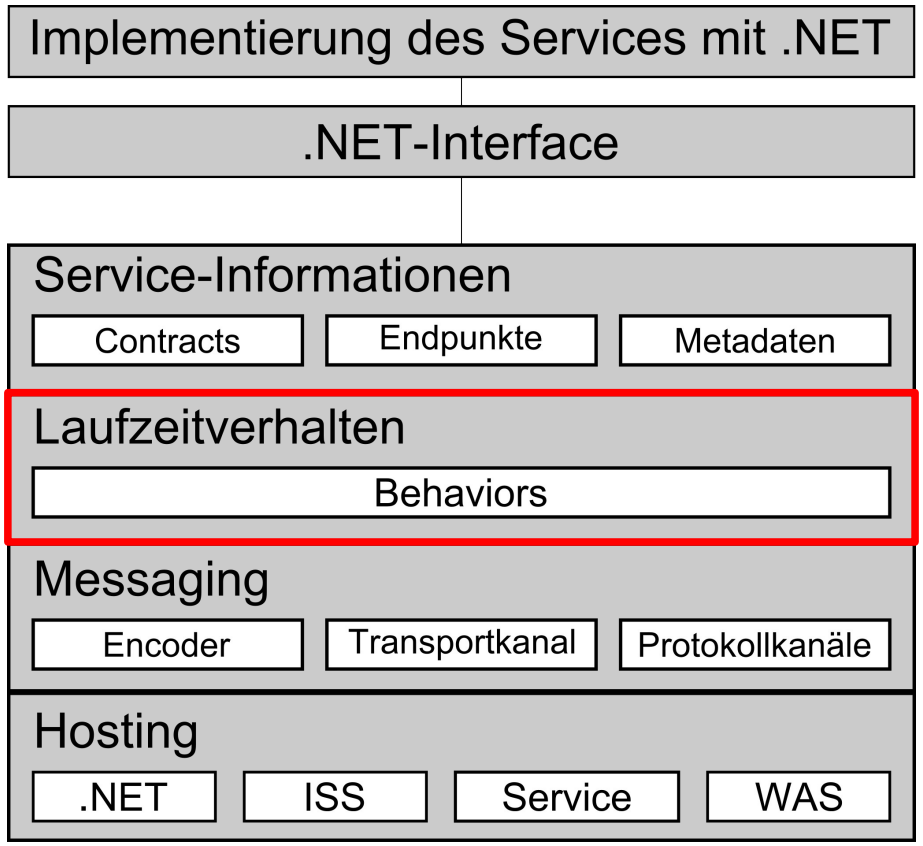
- Spezifizieren weitere optionale Standards wie z.B
  - Transaktionssicherheit
  - Authentifizierung
- Spezifiziert von:
  - World Wide Web Consortium (W3C)
  - Organization for the Advancement of Structured Information Standards (OASIS)



Agenda

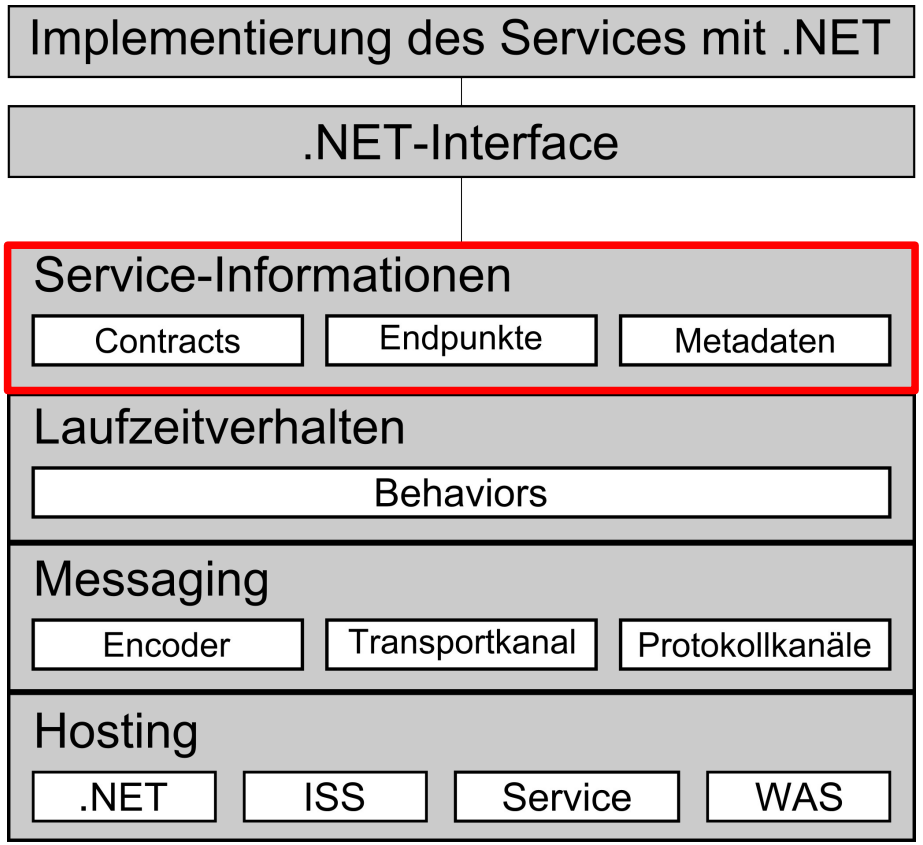
- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Laufzeitverhalten



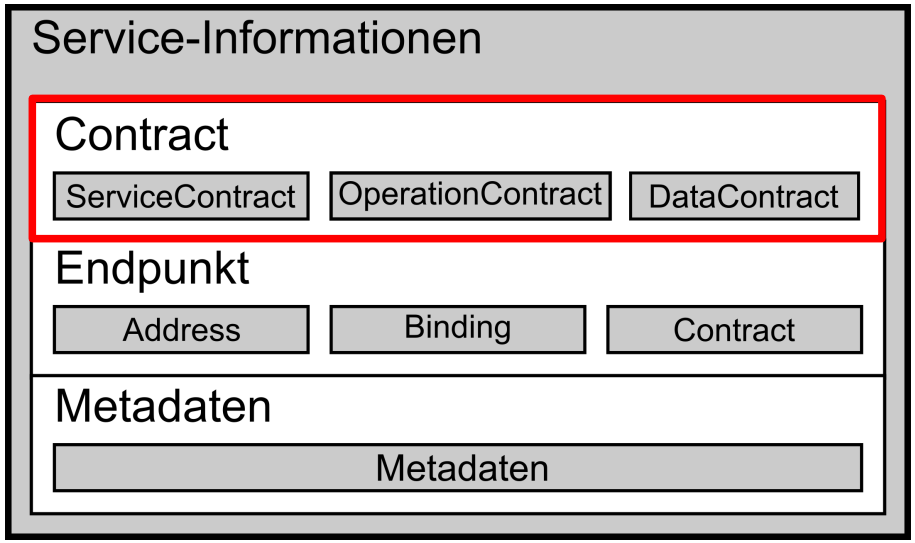
- Legt das **Verhalten** von Klient oder Service in bestimmten Situationen während der Laufzeit fest
- **Überwachen** die Nachrichten und geben gewisse **Regeln** vor (z.B. timeouts) oder **erweitern** um gewisse Funktionen (z.B. Logging der Nachrichten
- Diese werden **nicht** ausgetauscht, sondern legt jeder Klient und Service für sich selbst fest

Service-Informationen



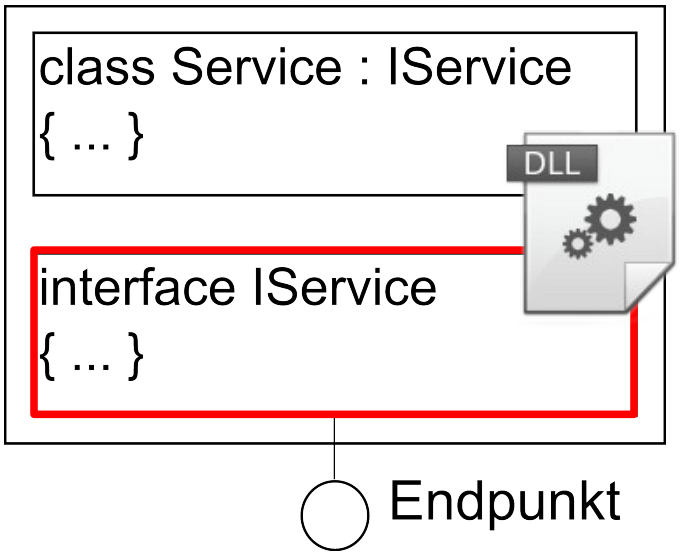
- Alle Informationen, die der **Client benötigt**

Contract



- Definiert die veröffentlichten Funktionalitäten
- ServiceContract
  - Veröffentlicht Klasse oder Interface
- OperationContract
  - Veröffentlicht Methode in einem ServiceContract
- DataContract
  - Veröffentlicht Typen

Contract (Beispiel)




```
using System;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace EchoLibrary
{
    [ServiceContract]
    public interface IEcho
    {
        [OperationContract(Name = "echoString")]
        string echo(string echoMsg);
    }
}
```

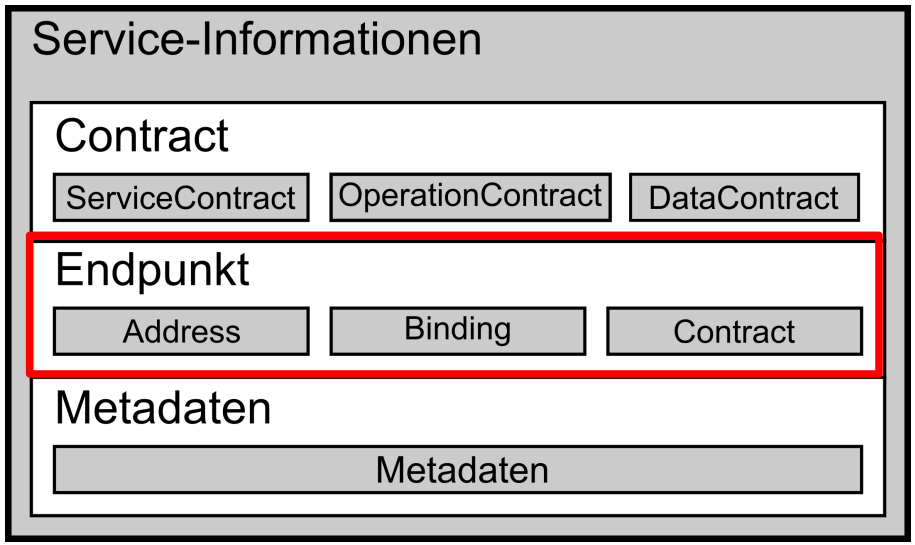
WCF Assembly

Veröffentlichen

Veröffentlichen

Eindeutiger Name für Client als Parameter

Endpunkte

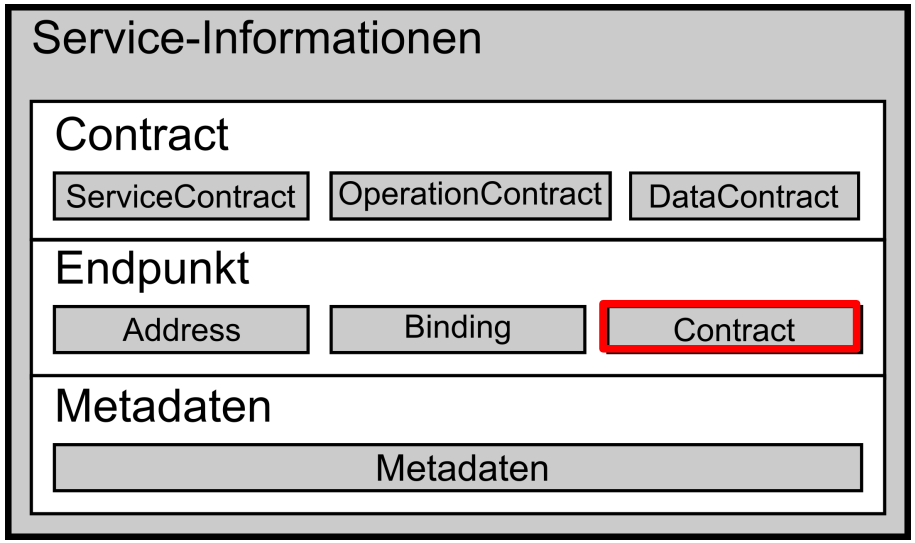


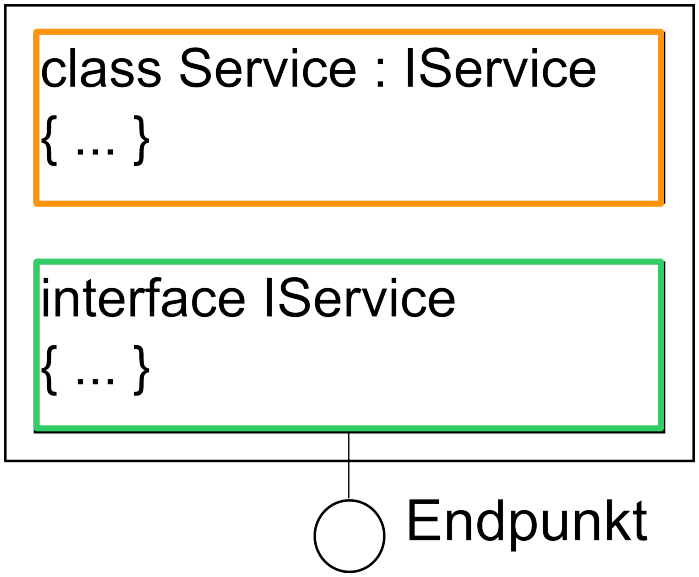
- Über Endpunkte können Nachrichten **ausgetauscht** werden
- Ein Service kann **mehrere** Endpunkte haben

```
<services>
  <service name="EchoLibrary.Echo">
    <endpoint address="http://localhost:8732/" binding="wsHttpBinding"
              contract="EchoLibrary.IEcho">
    </endpoint>
  </service>
</services>
```

(app.config)

Contract

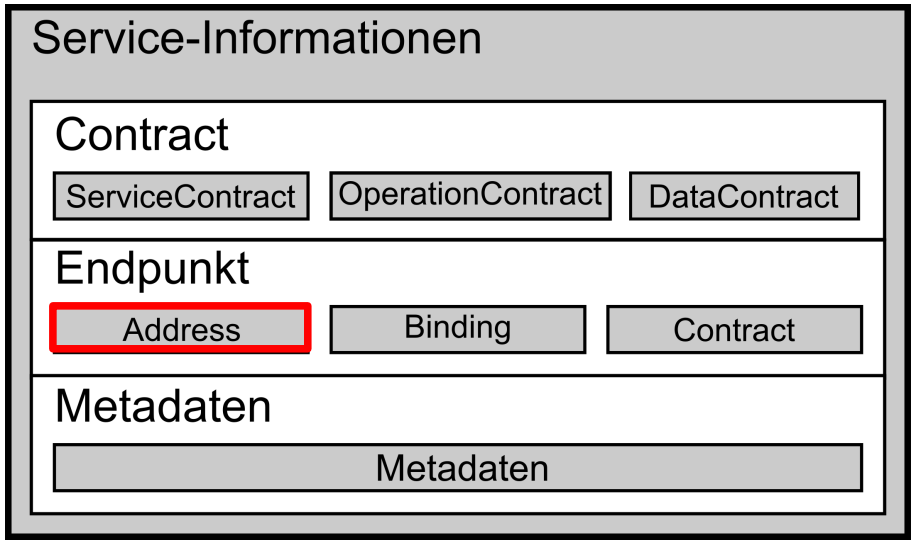





```
<service name="EchoLibrary.Echo">
  <endpoint address="http://localhost:8732/" binding="wsHttpBinding"
            contract="EchoLibrary.IEcho">
  </endpoint>
</service>
```

(app.config)

Adresse



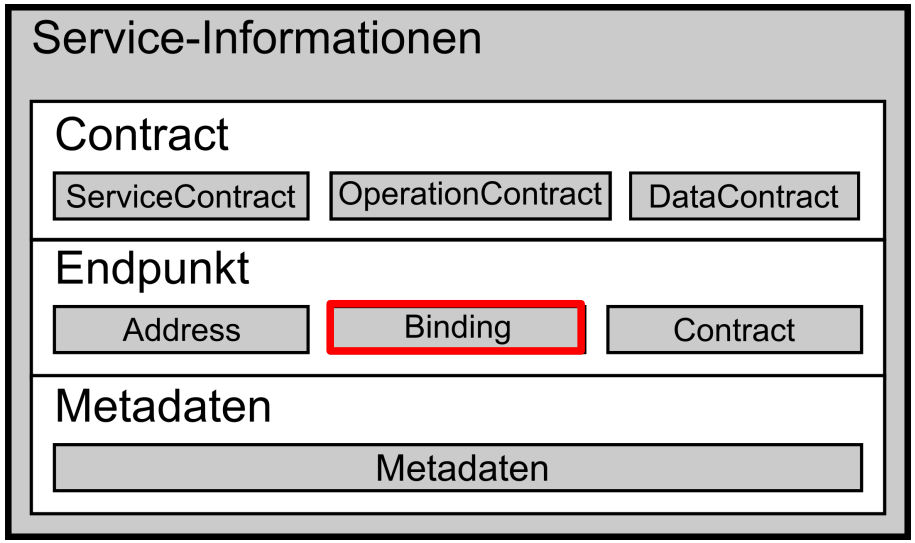
Transportprotokoll	URI-Schreibweise
HTTP	http
HTTPS	https
TCP	net.tcp
Named Pipes	net.pipe
MSMQ	net.msmq

Transportprotokoll://Domainname[:Portnummer][/Verzeichnis]

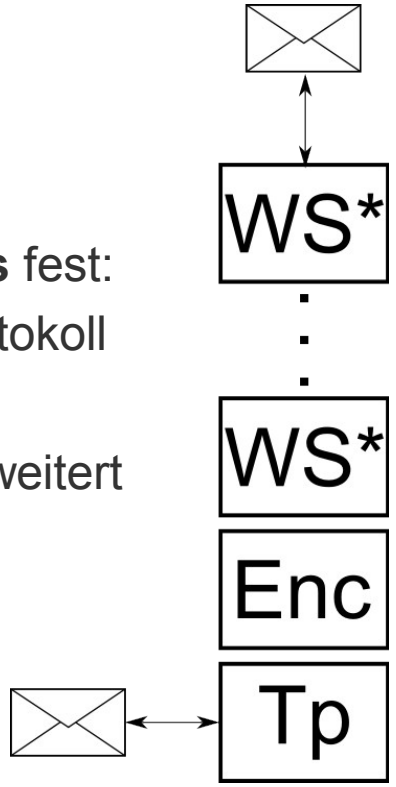
```
<service name="EchoLibrary.Echo">
  <endpoint address="http://localhost:8732/" binding="wsHttpBinding"
            contract="EchoLibrary.IEcho">
  </endpoint>
</service>
```

(app.config)

Binding

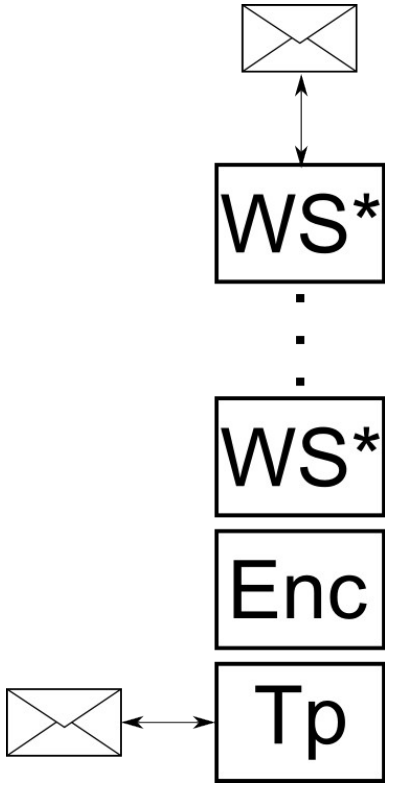


- Legt **mindestens** fest:
  - Transportprotokoll
  - Kodierung
- Kann beliebig erweitert werden (**WS*** Spezifikationen)



Vordefinierte Bindings

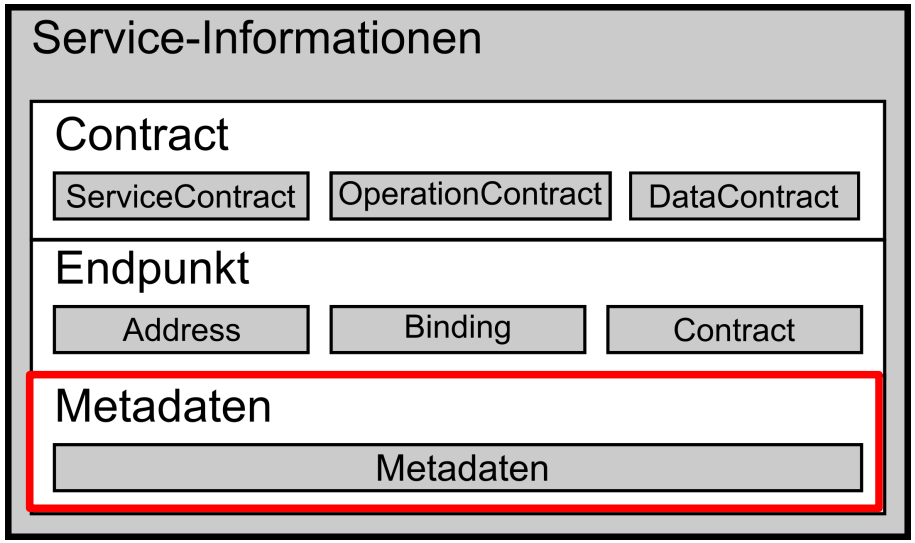
- BasicHttpBinding
  - http mit Standardport 80
  - xml Kodierung
- wsHttpBinding
  - Zusätzlich WS* Erweiterungen erlaubt




```
<service name="EchoLibrary.Echo">
  <endpoint address="http://localhost:8732/" binding="wsHttpBinding"
                    contract="EchoLibrary.IEcho">
  </endpoint>
</service>
```

(app.config)

Metadaten



- Enthalten alle **Service-Informationen**
- Können vom Klienten **abgerufen** werden
- Benötigt der Klient um zu wissen **wie** er mit dem Service Kommunizieren kann
- Können von Speziellen **Metadata Exchange (MEX)-Endpunkten** abgerufen werden oder per **HTTP-GET** (Behavior)

Agenda

- Grundproblem
- Bestandteile des Services
- Bestandteile des Klient
- Architektur der WCF
  - Hosting
  - Messaging
  - Laufzeitverhalten
  - Service Informationen
    - Contract
    - Binding
    - Medtadaten
- Zusammenfassung

Zusammenfassung

- WCF **veröffentlicht** Methoden und Klassen
- Beim Programmieren kann es als **Framework** eingebunden werden
- Funktionen werden automatisch und **transparent** serialisiert
- Architektur mit **unabhängig** konfigurierbaren Elementen, sodass diese **beliebig erweitert** werden können
- Fasst folgende Technologien **zusammen**:
  - .NET Remoting
  - .NET Enterprise (COM+)
  - Web Service Enhancements (WSE)
  - Microsoft Massage Queuing (MSMQ)
  - ASP.NET Webservices (ASMX) werde

Vielen Dank für ihre Aufmerksamkeit ...

Fragen?