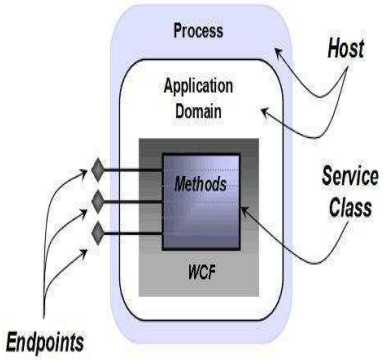
WCF
Windows Communication Foundation

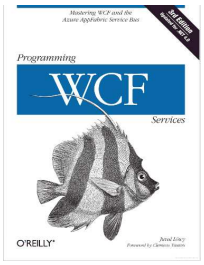


O que é o WCF

- **Unificação de várias tecnologias de suporte à distribuição**
  - **Objectos Distribuídos - .NET Remoting;**
  - **Web Services ASMX com extensões WSE (especificações WS-*);**
  - **Comunicação por mensagens MSMQ;**
  - **Enterprise Services COM+ (Queuing, Transactions, etc.);**
  - **Após definir um serviço é possível definir vários *endpoints*.**
  - **Um *endpoint* contém informação que define o caminho e condições de acesso ao serviço (*Address*, *Binding*, e *Contract*)**

Bibliografia

Microsoft Windows Communication Foundation 4.0 Cookbook for Developing SOA Applications,
By Steven Cheng
Packt Publishing, 2010;



Programming WCF Services, 3rd Edition
Mastering WCF and the Azure AppFabric Service Bus
By Juval Lowy
Publisher: O'Reilly Media, August 2010

ABC do WCF

endpoint

***A**ddress* – especifica para onde podem ser enviadas as mensagens (onde está localizado o serviço);

***B**inding* – descreve como enviar as mensagens: protocolo de transporte, segurança, estilo de codificação etc.;

***C**ontract* – descreve o que as mensagens podem conter (definindo operações, tipos de dados, mensagens e falhas).

- ✓ Os clientes necessitam saber o *ABC* do serviço para o poderem usar;
- ✓ Um *endpoint* actua como um *gateway* do serviço para o exterior;
- ✓ Os *endpoints* de um serviço são descritos em WSDL (Web Services Description Language).

Address

- **Formato dos endereços:**
  - `base address/[optional URI]`
    - `base address ⇒ [transport]://[machine or domain][:optional port]`
- **Exemplos:**
  - `http://localhost:8001`
  - `http://localhost:8001/MyService`
  - `net.tcp://localhost:8002/MyService`
  - `net.pipe://localhost/MyPipe`
  - `net.msmq://localhost/private/MyService`
  - `net.msmq://localhost/MyService`

Binding

http://msdn.microsoft.com/en-us/library/ms730879.aspx

Binding	Description
BasicHttpBinding	A binding that is suitable for communicating with WS-Basic Profile conformant Web services, for example, ASP.NET Web services (ASMX)-based services. This binding uses HTTP as the transport and text/XML as the default message encoding.
WSHttpBinding	A secure and interoperable binding that is suitable for non-duplex service contracts.
WSDualHttpBinding	A secure and interoperable binding that is suitable for duplex service contracts or communication through SOAP intermediaries.
WSFederationHttpBinding	A secure and interoperable binding that supports the WS-Federation protocol that enables organizations that are in a federation to efficiently authenticate and authorize users.
NetTcpBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications.
NetNamedPipeBinding	A secure, reliable, optimized binding that is suitable for on-machine communication between WCF applications.
NetMsmqBinding	A queued binding that is suitable for cross-machine communication between WCF applications.
NetPeerTcpBinding	A binding that enables secure, multiple machine communication.
MsmqIntegrationBinding	A binding that is suitable for cross-machine communication between a WCF application and existing Message Queuing applications.
BasicHttpContextBinding	A binding that is suitable for communicating with WS-Basic Profile conformant Web services that enables HTTP cookies to be used to exchange context.
NetTcpContextBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications that enables SOAP headers to be used to exchange context.
WebHttpBinding	A binding used to configure endpoints for WCF Web services that are exposed through HTTP requests instead of SOAP messages.
WSHttpContextBinding	A secure and interoperable binding that is suitable for non-duplex service contracts that enables SOAP headers to be used to exchange context.

Binding

http://msdn.microsoft.com/en-us/library/ms730879.aspx

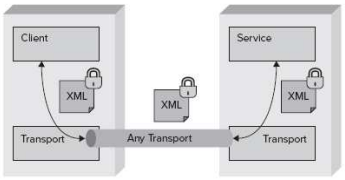
Binding	Interoperability	Security (Default)	Session (Default)	Transactions	Encoding (Default)
BasicHttpBinding	Basic Profile 1.1	(None), Transport, Message, Mixed	(None)	(None)	Text, (MTOM)
WSHttpBinding	WS	Transport, (Message), Mixed	(None), Reliable Session, Security Session	(None), Yes	(Text), MTOM
WSDualHttpBinding	WS	(Message), None	(Reliable Session), Security Session	(None), Yes	(Text), MTOM
WSFederationHttpBinding	WS-Federation	(Message), Mixed, None	(None), Reliable Session, Security Session	(None), Yes	(Text), MTOM
NetTcpBinding	.NET	(Transport), Message, None, Mixed	(Transport), Reliable Session, Security Session	(None), Yes	Binary
NetNamedPipeBinding	.NET	(Transport), None	None, (Transport)	(None), Yes	Binary
NetMsmqBinding	.NET	Message, (Transport), None	(None), Transport	None, (Yes)	Binary
NetPeerTcpBinding	Peer	(Transport)	(None)	(None)	
MsmqIntegrationBinding	MSMQ	(Transport)	(None)	None, (Yes)	n/a
BasicHttpContextBinding	Basic Profile 1.1	(None), Transport, Message, Mixed	(None)	(None)	Text, (MTOM)
NetTcpContextBinding	.NET	(Transport), Message, None, Mixed	(Transport), Reliable Session, Security Session	(None), Yes	Binary
WSHttpContextBinding	WS	Transport, (Message), Mixed	(None), Reliable Session, Security Session	(None), Yes	Text, (MTOM)

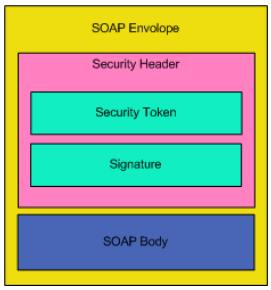
*Web Services Specifications WS-**

- Representam um conjunto de protocolos acordados na indústria (IBM, Microsoft, Sun, VeriSign etc.), introduzindo extensões ao SOAP.
- Inicialmente a Microsoft lançou o *framework* WSE (web services enhancements) , que expandia o *stack* de Web services (asmx) com a implementação das especificações WS-*.
- A última versão WSE 3.0, incluía a implementação dos protocolos:
  - WS – Security;
  - WS – Addressing;
  - WS – SecureConversation;
  - MTOM (Message Transmission Optimization Mechanism)
  - WS - ReliableMessaging.
- Em 2006, a Microsoft descontinuou o WSE, incorporando no WCF a experiência acumulada.

WS-Security

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlsn:a="http://www.w3.org/2005/08/addressing"
xmlsn:u="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <a:Action s:mustUnderstand="1" u:Id="_2">
       http://tempuri.org/IHelloWorld/Hello
    </a:Action>
    <a:MessageID u:Id="_3">
      urn:uuid:4dabcfb1-6939-402d-919b-8e8486206e1f
    </a:MessageID>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/
      wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="uuid-ea4659dc-85e5-4054-9b0c-c674f0128e7f-11">
        <u:Created>2009-07-27T20:03:36.830Z</u:Created>
        <u:Expires>2009-07-27T20:08:36.830Z</u:Expires>
      </u:Timestamp>
      <!-- Removed for simplicity -->
    </o:Security>
  </s:Header>
  <s:Body u:Id="_0">
    <e:EncryptedData Id="_1" Type="http://www.w3.org/2001/04/xmlenc#Content"
      xmlns:e="http://www.w3.org/2001/04/xmlenc#">
      <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
      </e:EncryptionMethod>
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-wssecurity-secext-1.0.xsd">
          <o:Reference ValueType="http://schemas.xmlsoap.org/ws/2005/02/sc/dk"
            URI="#uuid-ea4659dc-85e5-4054-9b0c-c674f0128e7f-10"></o:Reference>
        </o:SecurityTokenReference>
      </KeyInfo>
      <e:CipherData>
        <e:CipherValue>.....</e:CipherValue>
      </e:CipherData>
    </e:EncryptedData>
  </s:Body>
</s:Envelope>
```





MTOM – Message Transmission Optimization Mechanism

- **Standard do W3C, para transferir de forma eficiente dados binários (*attachments*) em mensagens SOAP.**

Mensagem SOAP sem MTOM

```
<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:ns1="http://catalog.mtom.company.com/book">
  <soapenv:Body>
    <ns1:chapter>
      <name>Introducing SOA Security</name>
      <content>
          JVBERi0xLjQNJeLjz9MN ...( base64 encoded sample chapter content,)
      </content>
    </ns1:chapter>
  </soapenv:Body>
</soapenv:Envelope>
```

conteúdos binários são convertidos em texto

extraído de:
http://www.devx.com/xml/Article/34797/0/page/1

Mensagem com MTOM

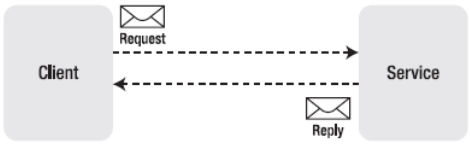
```
<soapenv:Envelope  ....  xmlns:ns1="http://catalog.mtom.company.com/book">
   <soapenv:Body>
      <ns1:chapter>
         <name>Introducing SOA Security</name>
         <content>
            <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
              href="cid:ac7744d3-5f6d-45d8-b9eb-79e9cd58702f@example.jaxws.sun.com">
            </xop:Include>
         </content>
      </ns1:chapter>
   </soapenv:Body>
</soapenv:Envelope>

Content-Type: application/octet-stream
Content-ID: <ac7744d3-5f6d-45d8-b9eb-79e9cd58702f@example.jaxws.sun.com>
Content-transfer-encoding: binary
```

... (binary stream for the sample chapter content)

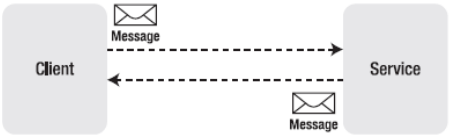
conteúdos binários são enviados em binário sobre http

Message Exchange Patterns



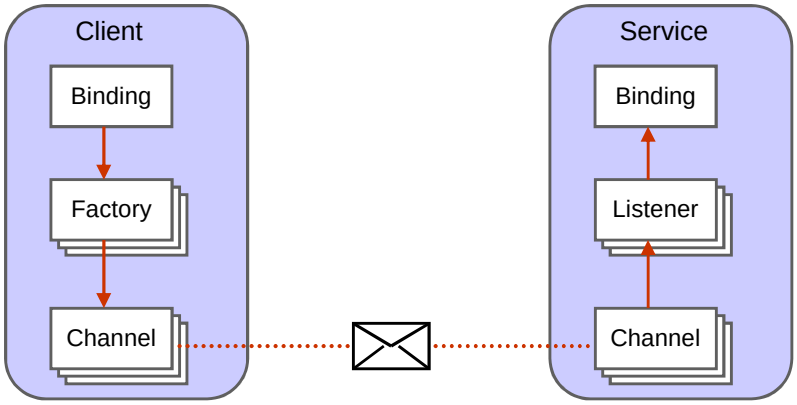
Request-Reply messaging

One-Way messaging



Duplex messaging

WCF Message Stack



- Um cliente pode aceder ao serviço através de um ou mais *channels* criados através do padrão *factory*. O serviço aceita mensagens de vários *channels*;
- Os *Channels* podem ser conectados em *pipeline* para aplicar as funcionalidades WS-* de *security, reliability, session state,* etc. para cada protocolo de transporte.

Contract

- A definição de contratos de serviços é feita com base em atributos
- O WCF permite quatro tipos de contrato:
  - *Service Contract* - define as funcionalidades (operações) do serviço;
  - *Data Contract* - define a estrutura e conteúdo da informação trocada entre o cliente e o serviço;
  - *Fault Contract* - define as *SOAP faults* que os clientes devem esperar quando acedem às operações do serviço;
  - *Message Contract* - permitem mapear tipos CLR com mensagens SOAP, proporcionando um controlo sobre os SOAP *headers* e a parte *body* das mensagens.

Tipos de Contrato

```
[ServiceContract]
public interface IMyContract{...}
[OperationContract]
void SomeOperation ();\
```

```
[DataContract]
public class SomeType { [DataMember] public int ID; };
```

```
[MessageContract]
public class MyRequest {
      [MessageHeader]public string field1;
      [MessageBody]public string field2;
}
```

```
[OperationContract]
[FaultContract(typeof(DivideByZeroException))]
void SomeOperation ();
```


Alojamento de Serviços

- *Managed (Self-Host)* – qualquer aplicação pode alojar serviços WCF

```
Uri addr = new Uri("http://localhost:8080/ServiceOla");
WSHttpBinding bind = new WSHttpBinding();
Type contrato = typeof(IServiceOla);

ServiceHost  svchost = new ServiceHost(typeof(ServiceOla));
svchost.AddServiceEndpoint(contrato, bind, addr);

svchost.Open();
```

- *Internet Information Service (IIS)*
- *Windows Activation Service (WAS) (IIS 7.0 – Vista)*
- *Managed Windows Service*

Behaviors

- *Behaviors* - permitem modificar como os serviços e os clientes operam em áreas como:
  - Instanciação, Concorrência e *throttling*
  - Manipulação de erros
  - Segurança
  - Transacções

Estrangular no sentido de limitar, regular

[ServiceBehavior] - Permite configurar comportamentos (*behaviors)* inerentes ao serviço e que afectam todos os *endpoints* do mesmo.
[OperationBehavior] - Pode ser aplicado a uma operação do contrato do serviço.

enforces SOAP MustUnderstand header processing

```
[ServiceBehavior(ValidateMustUnderstand=true)]
public class MyService : ISomeContract
```


Gestão de Instâncias

- *PerCall* - É criada uma nova instância por cada chamada
- *PerSession* *(default)* – Uma instância por cliente. Os clientes não partilham instâncias;
- *Single* (*Singleton*) – Uma instância é partilhada por todos os clientes

```
[ServiceBehavior(
      InstanceContextMode=InstanceContextMode.Single)]
public class MyService : ISomeContract
```

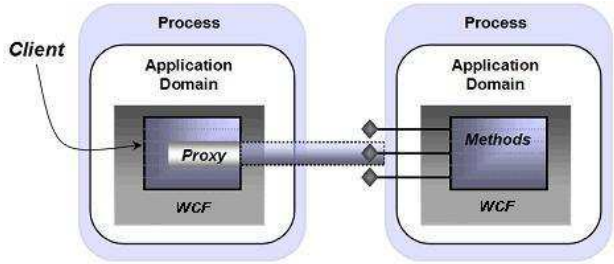
```
<throttling maxConcurrentCalls = "12"
      maxConnections = "34" maxInstances = "56" />
```

Concorrência

- O controlo de concorrência pode ser feito através do atributo [ServiceBehaviorAttribute], aplicado à classe que implementa o serviço, recorrendo à propriedade *ConcurrencyMode*, com um dos seguintes valores:
  - *Single* ***(default)*** – Apenas um pedido pode ser processado por instância de serviço (*single-thread*)
  - *Multiple* – Todos os pedidos realizados num serviço são processados em simultâneo (*multi-thread*) pela instância do serviço. Requer sincronização de estado do serviço (*thread-safe*).
  - *Reentrant* – Comporta-se como o modo *Single*, no entanto, permite libertar o *lock* quando é efectuado um pedido a outro serviço WCF ou é invocado um método de *callback.*

WCF - Clients

- Um cliente é um aplicação (ex: Console, Windows *Forms*, ASP .NET) ou outro serviço, que troca mensagens (SOAP) com um ou mais *Endpoints*
- O cliente acede ao serviço através de um *proxy* (que expõe o contrato do serviço) e que é gerado automaticamente a partir da *metadata* existente no WSDL do serviço.



Implementar um Serviço

```
[ServiceContract]
public interface IOrderService
{
    [OperationContract]
    void CreateOrder(int orderNumber);
    [OperationContract]
    void AddItemToOrder(int orderNumber, Item itm);
    [OperationContract]
    Order GetOrderDetails(int orderNumber);
}
```

```
public class OrderService : IOrderService
{
    void CreateOrder(int orderNumber){...}
    void AddItemToOrder(int orderNumber, Item itm){...}
    Order GetOrderDetails(int orderNumber){...}
}
```


Data Contract

```
using System.Runtime.Serialization;

[DataContract]
public class Order
{
      [DataMember]
      public int OrderNumber;
      [DataMember]
      public string ClientName;
      ...
};
```

Configuração do Serviço

Em Código

```
using System.ServiceModel;
...
ServiceHost myHost=new ServiceHost(typeof(OrderService));
Uri addUri = new Uri("http://localhost:8000/MyService");
WSHttpBinding binding = new WSHttpBinding();
myHost.AddServiceEndpoint(typeof(IOrderService), binding, addUri);
```

Em Ficheiros de Configuração

```
<system.serviceMode><services>
  <service>
     <endpoint address=http://localhost:8000/MyService
        binding="WSHttpBinding" contract="IOrderService"/>
  </service>
</services></system.serviceModel>
```


Exemplo: *WCFOlaMundo.zip*

Nota: Com objectivos pedagógicos, este exemplo faz as configurações do serviço e do cliente (*ClienteSimple*) programaticamente. Como veremos as configurações devem ser feitas nos ficheiros de configuração.

- ✓ Este exemplo, ilustra os conceitos fundamentais do WCF:
  - Definição de um *Service Contract*
  - Definição de um *Data Contract*
  - Definição de um *FaultContract* (operação *division*)
  - Alojamento (*hosting*) de um serviço
  - Possibilidade de testar diferentes *bindings* e *Behaviors*
  - 1 cliente que partilha o *assembly* com os contratos;
  - 1 cliente WCF que usa um *proxy* por adição de um <Service Reference>
  - 1 cliente com *proxy* gerado por adição de <Web Reference>, compatível com *Basic Profile 1.1* (O *binding* tem de ser *BasicHttpBinding*)

Sugere-se o estudo aprofundado deste exemplo, nomeadamente na análise do WSDL do serviço, perante a utilização de: diferentes *bindings*; protocolo MTOM; *behaviors* relacionados com modos de instanciação e controlo de concorrência e utilização de SOAP *faults*.

Contrato do serviço

```
[ServiceContract]
public interface IServiceOla
{
  [OperationContract]
  string olaSimples(string nome);
  [OperationContract]
  string olaPessoa(Pessoa pes);
  [OperationContract]
  Pessoa getPessoa(string firstName, string lastName);
  [OperationContract]
  void changeState(int val);
  [OperationContract]
  int getState();
  [OperationContract]
  string[] teste(string[] str);
  [OperationContract]
  byte[] getDataMTOM(byte[] arg);
  [OperationContract]
  [FaultContract(typeof(DivideByZeroException))]
  int division(int a, int b);
}
```


Contrato de Dados

```
[DataContract] //(Namespace = "http://ADEETC.ISEL")]
    public class Pessoa
    {
        string firstName;
        string lastName;

        [DataMember] //(Name="FirstName")]
        public string FirstName
        {
            get { return firstName; }
            set { firstName = value; }
        }
        [DataMember] //(Name = "LastName")]
        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }
    }
```

Utilização de MTOM (Message Transmission Optimization Mechanism)

- **Chamada à operação** `public byte[] getDataMTOM(byte[] arg)` **com o cliente *WCFclient***
- **O *binding* no *HostService* deve ser criado com suporte para MTOM;**

```
WSHttpBinding bind = new WSHttpBinding();
bind.MessageEncoding = WSMessageEncoding.Mtom;
```

- **Para analisar as mensagens SOAP, sugere-se a utilização do programa *TCP Viewer* para interceptar os pedidos ao serviço. Na criação do *proxy* no *WCFclient* definimos como *endpoint* o porto 8081, quando na realidade o serviço está no porto 8080.**

```
IServiceOla prx = new ServiceOlaClient(
                                    "WSHttpBinding_IServiceOla",
                                    "http://localhost:8081/ServiceOla"
                                  );
```

- **Nos slides seguintes ilustra-se a mensagens SOAP de envio de um *array* de bytes com protocolo MTOM e a mensagem de resposta com *SOAP fault*, indicando a não conformidade do *endpoint*, devido à alteração do porto no cliente.**

Mensagem SOAP com MTOM

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IServiceOla/getDataMTOM</a:Action>
    <a:MessageID>urn:uuid:66d4c7a8-dc0a-4305-b229-180221dfae1c</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://localhost:8081/ServiceOla</a:To>
  </s:Header>
  <s:Body>
    <getDataMTOM xmlns="http://tempuri.org/">
      <arg>
        <xop:Include href="cid:http://tempuri.org/1/634417607422120000"
                     xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
      </arg>
    </getDataMTOM>
  </s:Body>
</s:Envelope>
Content-ID: <http://tempuri.org/1/634417607422120000>
Content-Transfer-Encoding: binary
Content-Type: application/octet-stream
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU . . .
. . .
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
```

```
byte[] arg = new byte[1024];
for (int j = 0; j < 1024; j++)
    arg[j]=0x55;
byte[] buf = prx.getDataMTOM(arg);
```

Mensagem *SOAP fault*, devido a inconformidade do *endpoint*

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">
              http://www.w3.org/2005/08/addressing/soap/fault
    </a:Action>
    <a:RelatesTo>urn:uuid:66d4c7a8-dc0a-4305-b229-180221dfae1c</a:RelatesTo>
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Sender</s:Value>
        <s:Subcode>
          <s:Value>a:DestinationUnreachable</s:Value>
        </s:Subcode>
      </s:Code>
      <s:Reason>
        <s:Text xml:lang="pt-PT">
          The message with To 'http://localhost:8081/ServiceOla' cannot be processed
          at the receiver, due to an AddressFilter mismatch at the EndpointDispatcher.
          Check that the sender and receiver's EndpointAddresses agree.
        </s:Text>
      </s:Reason>
    </s:Fault>
  </s:Body>
</s:Envelope>
```


Exemplo: Operações *OneWay message*

Contrato

```
[ServiceContract]
public interface IService1
{
   [OperationContract(IsOneWay = false)]
   void DoWork(DataMessage m);

   [OperationContract(IsOneWay = true)]
   void DoWorkAsOneWay(string str);
}


[MessageContract(WrapperName="DataMessage“, WrapperNamespace=“SD")]
public class DataMessage
{
  [MessageHeader]
  public string TextContent { get; set; }
  [MessageBodyMember]
  public byte[] ImageData { get; set; }
}
```

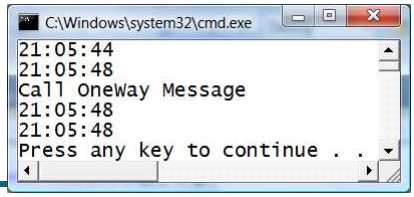

Serviço

```
public class Service1 : IService1
{
    void IService1.DoWork(DataMessage m)
    {
        System.Threading.Thread.Sleep(4 * 1000);
    }

    void IService1.DoWorkAsOneWay(string str)
    {
        System.Threading.Thread.Sleep(4 * 1000);
    }
}
```

Cliente

```
class Program
{
  static void Main(string[] args)
  {
    Service1Client prx = new Service1Client();
    Console.WriteLine(System.DateTime.Now.ToLongTimeString());
    DataMessage m = new DataMessage();
    m.TextContent="Text"; m.ImageData=new byte[]{1,2,3,4,5};
    prx.DoWork(m.TextContent, m.ImageData);
    Console.WriteLine(System.DateTime.Now.ToLongTimeString());
    Console.WriteLine("Call OneWay Message");
    Console.WriteLine(System.DateTime.Now.ToLongTimeString());
    prx.DoWorkAsOneWay("Luis");
    Console.WriteLine(System.DateTime.Now.ToLongTimeString());
  }
}
```



Request da operação *DoWork* (IsOneWay = false)

```
05-01-2011 17:02:13.423; 500: Client to Server (242 bytes)
POST /Service1.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/IService1/DoWork"
Host: localhost:6524
Content-Length: 131
Expect: 100-continue
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
05-01-2011 17:02:13.426; 500: Server to Client (126 bytes)
HTTP/1.1 100 Continue
Server: ASP.NET Development Server/10.0.0.0
Date: Wed, 05 Jan 2011 17:02:13 GMT
Content-Length: 0
05-01-2011 17:02:13.454; 500: Client to Server (131 bytes)
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
      <Action s:mustUnderstand="1"
              xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
         http://tempuri.org/IService1/DoWork
      </Action>
    <h:TextContent xmlns:h="http://tempuri.org/">Text</h:TextContent>
  </s:Header>
  <s:Body>
    <DataMessage xmlns="SD">
        <ImageData xmlns="http://tempuri.org/">12345</ImageData>
    </DataMessage>
  </s:Body>
</s:Envelope>
```

Response da operação *DoWork* (IsOneWay = false)

```
05-01-2011 17:02:17.456; 500: Server to Client (233 bytes)
HTTP/1.1 200 OK
Server: ASP.NET Development Server/10.0.0.0
Date: Wed, 05 Jan 2011 17:02:17 GMT
X-AspNet-Version: 4.0.30319
Content-Length: 139
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Connection: Close
05-01-2011 17:02:17.461; 500: Server to Client (139 bytes)
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
      <DoWorkResponse xmlns="http://tempuri.org/" />
  </s:Body>
</s:Envelope>
```

Note que embora o retorno da operação seja *void*
Existe mensagem SOAP de resposta

Request da operação *DoWorkAsOneWay* (IsOneWay = true)

```
05-01-2011 17:02:17.471; 500: Client to Server (226 bytes)
POST /Service1.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/IService1/DoWorkAsOneWay"
Host: localhost:6524
Content-Length: 139
Expect: 100-continue
Accept-Encoding: gzip, deflate
05-01-2011 17:02:17.478; 500: Server to Client (126 bytes)
HTTP/1.1 100 Continue
Server: ASP.NET Development Server/10.0.0.0
Date: Wed, 05 Jan 2011 17:02:17 GMT
Content-Length: 0
05-01-2011 17:02:17.480; 500: Client to Server (139 bytes)
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <DoWorkAsOneWay xmlns="http://tempuri.org/">
      <str>Luis</str>
    </DoWorkAsOneWay>
  </s:Body>
</s:Envelope>
```

Response da operação *DoWorkAsOneWay* (IsOneWay = true)

```
05-01-2011 17:02:17.482; 500: Server to Client (198 bytes)
HTTP/1.1 202 Accepted
Server: ASP.NET Development Server/10.0.0.0
Date: Wed, 05 Jan 2011 17:02:17 GMT
X-AspNet-Version: 4.0.30319
Cache-Control: private
Content-Length: 0
Connection: Close
```

Note que não existe mensagem SOAP de resposta