Programming Indigo

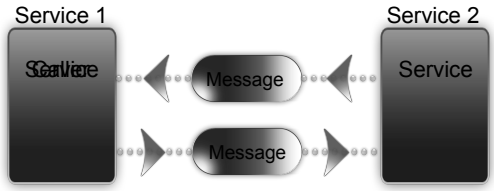
Payam Shodjai
Product Manager
Web Services Strategy
Microsoft Corporation
bix@microsoft.com

Agenda

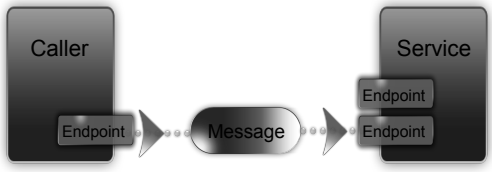
- Core Concepts
- Contracts
- Bindings
- Behaviors

Core Concepts

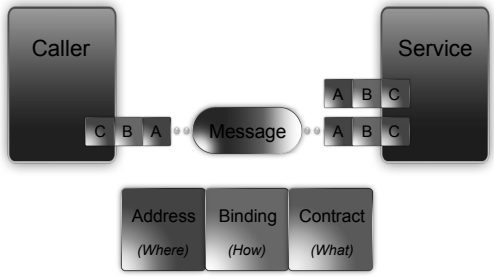
Service Communication



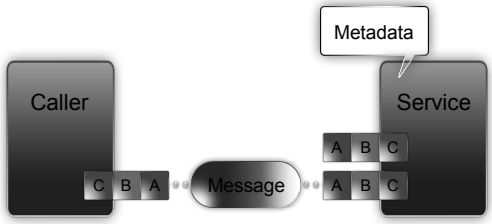
Endpoints



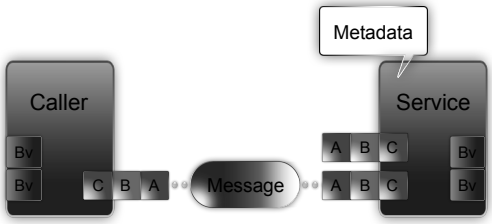
Address, Binding, Contract



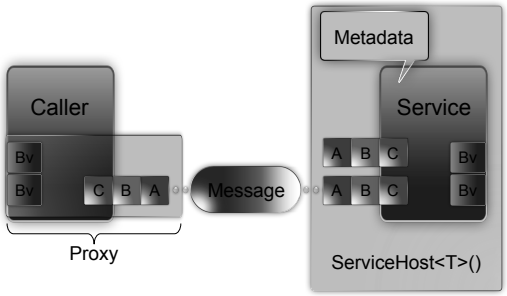
Metadata



Behaviors



Runtime



Demo

indigo

Contracts

Contracts: Overview

Service	Describes the operations a service can perform. Maps CLR types to WSDL.
Data	Describes a data structure. Maps CLR types to XSD.
Message	Defines the structure of the message on the wire. Maps CLR types to SOAP messages.

Service Contract

```
[ServiceContract]
public interface ICalculator
{
    [OperationContract]
    ComplexProblem SolveProblem (ComplexProblem p);
}
```

Service Contract: OneWay

```
[ServiceContract]
public interface IOneWayCalculator
{
    [OperationContract(IsOneWay=true)]
    void StoreProblem (ComplexProblem p);
}
```

Service Contract: Versioning

```
[ServiceContract]
public interface ICalculator2 : ICalculator
{
    [OperationContract(IsOneWay=true)]
    void SolveAndStoreProblem (ComplexProblem p);
}
```

Service Contract: Duplex

```
[ServiceContract(Session=true,
        CallbackContract=typeof(ICalculatorResults)]
public interface ICalculatorProblems
{
    [OperationContract(IsOneWay=true)]
    void SolveProblem (ComplexProblem p);
}

public interface ICalculatorResults
{
    [OperationContract(IsOneWay=true)]
    void Results(ComplexProblem p);
}
```

Service Contract: Faults

```
[ServiceContract(Session=true)]
public interface ICalculator
{
    [OperationContract]
    [FaultContract(typeof(DivideByZero))]
    ComplexProblem SolveProblem (ComplexProblem p);
}
```

```
try {
    return n1 / n2;
}
catch (DivideByZeroException e) {
    DivideByZero f = new DivideByZero (“Calculator”);
    throw new Fault<DivideByZero>(f);
}
```

Service Contract: Streams

```
[ServiceContract]
public interface IStreamCalculator
{
    [OperationContract]
    Stream Fibonacci(int iterations);
}
```

Data Contract

```
[DataContract]
public class ComplexNumber
{
    [DataMember]
    public double Real = 0.0D;
    [DataMember]
    public double Imaginary = 0.0D;

    public ComplexNumber(double r, double i)
    {
        this.Real = r;
        this.Imaginary = i;
    }
}
```

Data Contract: Names

```
[DataContract(Name="Complex",
              Namespace="http://BigMath.Samples")]
public class ComplexNumber
{
    [DataMember(Name="RealPart")]
    public double Real = 0.0D;
    [DataMember(Name="ImaginaryPart")]
    public double Imaginary = 0.0D;

    public ComplexNumber(double r, double i)
    {
        this.Real = r;
        this.Imaginary = i;
    }
}
```

Data Contract: Versioning

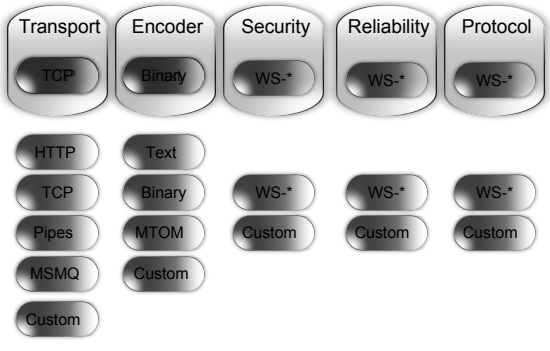
```
[DataContract(Name="Complex",
              Namespace="http://BigMath.Samples")]
public class ComplexNumber
{
    [DataMember(Name="RealPart")]
    public double Real = 0.0D;
    [DataMember(Name="ImaginaryPart")]
    public double Imaginary = 0.0D;
    [DataMember(VersionAdded=2, Name="BasePart",
            MustUnderstand=false, IsOptional=true)]
    public int Base = 10;

    public ComplexNumber(double r, double I, int b)
    {
        this.Real = r;
        this.Imaginary = i;
        this.Base = b;
    }
}
```

Message Contract

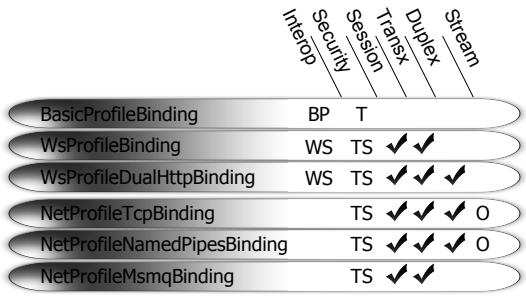
```
[MessageContract]
public class ComplexProblem
{
    [MessageHeader]
    public string operation;
    [MessageBody]
    public ComplexNumber n1;
    [MessageBody]
    public ComplexNumber n2;
    [MessageBody]
    public ComplexNumber solution;

    // Constructors…
}
```

Bindings

Elements of Binding

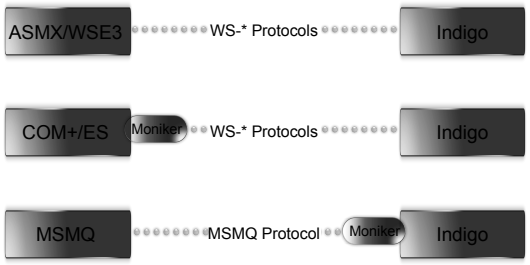


Standard Bindings



T = Transport Security | S = WS-Security | O = One-Way Only

Interop With Existing Technologies



Binding in Config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.serviceModel>
    <services>
      <service serviceType="CalculatorService">
        <endpoint address="Calculator"
                  bindingSectionName="basicProfileBinding"
                  contractType="ICalculator" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Configuring Bindings

```
<endpoint address="Calculator"
          bindingSectionName="basicProfileBinding"
          bindingConfiguration="Binding1"
          contractType="ICalculator" />


<bindings>
  <basicProfileBinding>
    <binding configurationName="Binding1"
             hostnameComparisonMode="StrongWildcard"
             transferTimeout="00:10:00"
             maxMessageSize="65536"
             messageEncoding="Text"
             textEncoding="utf-8"
    </binding>
  </basicProfileBinding>
</bindings>
```

Custom Bindings

```
<bindings>
   <customBinding>
      <binding configurationName="Binding1">
         <reliableSession bufferedMessagesQuota="32"
            inactivityTimeout="00:10:00"
            maxRetryCount="8"
            ordered="true" />
         <httpsTransport manualAddressing="false"
            maxMessageSize="65536"
            hostnameComparisonMode="StrongWildcard"/>
         <textMessageEncoding maxReadPoolSize="64"
            maxWritePoolSize="16"
            messageVersion="Default"
            encoding="utf-8" />
      </binding>
   </customBinding>
</bindings>
```

Behaviors

Behaviors: Overview

- Behaviors are all local
- Developers care about some behaviors
  - Concurrency
  - Instancing
- Deployers care about other behaviors
  - Throttling
  - Metadata exposure

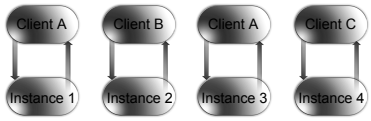
Instancing

- Per call
- Singleton
- Private Session
- Shared Session

Instancing

Scenario:
1. A accesses service
2. B accesses service
3. A accesses service again
4. C accesses service

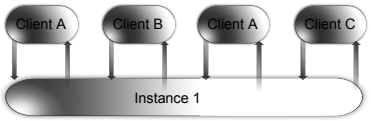
Per Call



Instancing

Scenario:
1. A accesses service
2. B accesses service
3. A accesses service again
4. C accesses service

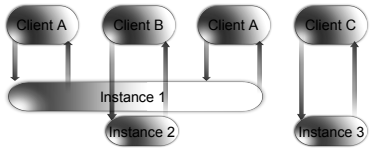
Singleton



Instancing

Scenario:
1. A accesses service
2. B accesses service
3. A accesses service again
4. C accesses service

Private Session



Instancing & Concurrency

```
[ServiceBehavior(ConcurrencyMode=ConcurrencyMode.Multiple,
                 InstanceMode = InstanceMode.Singleton)]
public class Calculator : ICalculator
{
   public ComplexProblem SolveProblem (ComplexProblem p)
   {
      switch (p.operation)
      {
         case "Add":
            p.solution.Real = p.n1.Real + p.n2.Real;
            p.solution.Imaginary =
                     p.n1.Imaginary + p.n2.Imaginary;
            return p;
            break;
         // …
      }
   }
}
```

Throttling

```
<service
   serviceType="Calculator"
   behaviorConfiguration="CalculatorBehavior">
   <!-- endpoint definitions /-->
</service>

<behaviors>
  <behavior configurationName="CalculatorBehavior" >
    <throttling maxConcurrentCalls="10"
                maxConnections="10"
                maxInstances="10"
                maxPendingOperations="10" />
  </behavior>
</behaviors>
```

Metadata

```
<service
   serviceType="Calculator"
   behaviorConfiguration="CalculatorBehavior">
   <!-- endpoint definitions /-->
</service>

<behaviors>
  <behavior configurationName="CalculatorBehavior" >
    <metadataPublishing enableGetWsdl="true"
                        enableHelpPage="true"
                        enableMetadataExchange="true"
     />
  </behavior>
</behaviors>
```

Demo

Summary

- Broad range of functionality for building distributed applications
- Functionality provided through several fundamental concepts
  - Addresses
  - Bindings
  - Behaviors
  - Contracts

payam@microsoft.com

Microsoft®