# Mono Infocard project::

**Atsushi Eno**
atsushi@ximian.com

October 24, 2006

Novell®

# Scope

Implement "Infocard" (Windows CardSpace, WCS) functionality through "Indigo" (Windows Communication Foundation, WCF)

**This talk includes**

- Introduction to Indigo (WCF) and Infocard (WCS)
- our Development status and plans
- Short demo

# WCF (Indigo): abstract

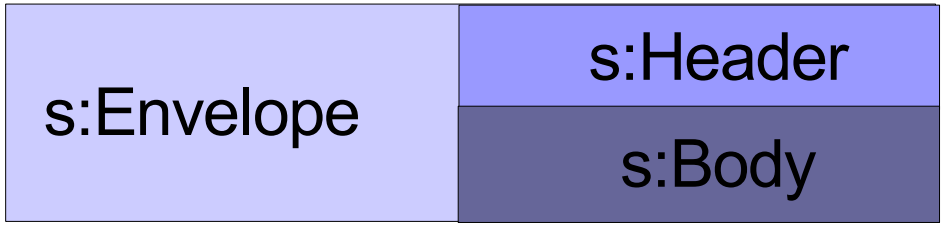
A messaging framework:

- contract based data serialization
- less channel dependent messages
- simplex and duplex messaging
- SOAP and non-SOAP
- session support
- transaction support
- WS-Security (encryption / sign)

# SOAP

## SOAP is the messaging framework for WS

- Envelope, Header and Body
- SOAP Action (represents an operation)



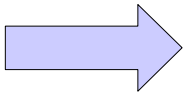

## How a SOAP message is typically integrated

- A method in a class becomes a SOAP Action
- The method's parameters becomes the SOAP request Body.
- The method's return value is created from the SOAP response body.

# Contract-based Data Serialization

- "XML" based serialization
- serializes only explicitly-specified members
  - binary serialization: [NonSerialized]
  - XML serialization: [XmlIgnore]
  - [DataContract] to indicate to use contract based serialization, [DataMember] to indicate the member is serialized.

```
public class Banana {
  public double Weight;
  public string Secret;
}
```



```
[DataContract]
public class Banana {
  [DataMember]
  public double Weight;
  public string Secret;
}
```

# Contract-based Services

## Service Contract and Operation Contract

- describes a service contract: SOAP actions, session mode, security requirement.
- Supports Data Contract, but XML serialization and custom message creation are also supported.

```
[ServiceContract]
public interface IMonkey
{
  [OperationContract]
  string EatBanana (Banana banana);
}
```

# WCF typical messaging how to

## Service:

- implement the service contract interface and compile into a library dll.
- create a file [yourservice].svc which just contains:
  <%@ServiceHost Service="serviceclass, yourservice.dll" %>
- Create "bin" directory and move dll file there, and run xsp2

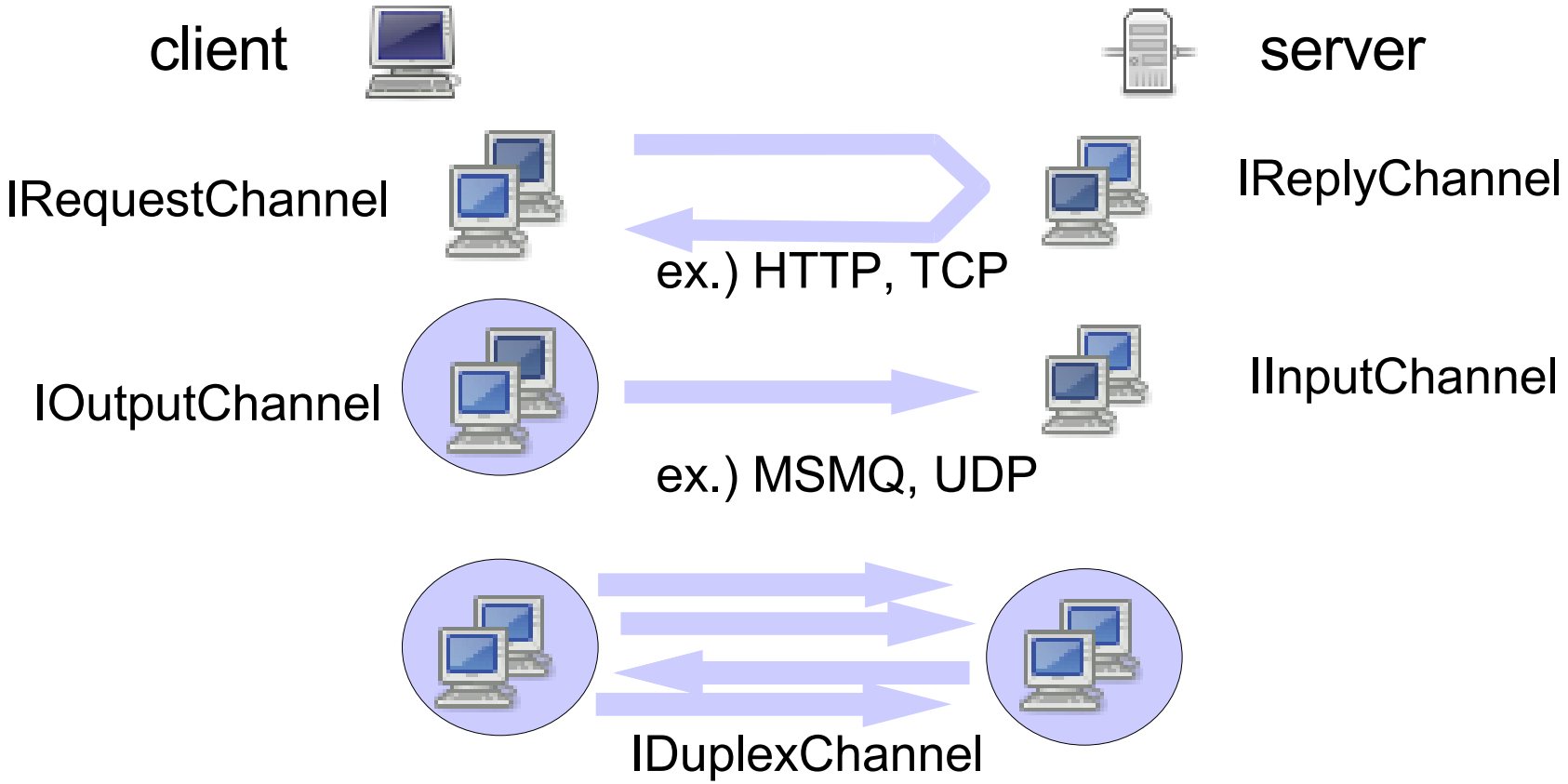
## Client:

- access the service with svcutil to create a client proxy:
  $ svcutil http://yoursite.com/[yourservice].svc?wsdl
- compile and use the generated proxy class in your application.

# WCF bindings and transport options

- Binding: a set of communication protocol requirements.
- Transport binding elements:
  - self hosting HTTP(S)
  - ASP.NET HTTP(S)
  - TCP
  - P2P
  - named pipe
  - ~~MSMQ~~
  - ~~COM+~~

# WCF channel types




channel type support is dependent on transport types.

# Security support in WCF

Secure messaging

- signing
- encryption

A couple of supported security specifications:

- WS-Security
- WS-SecurityPolicy
- WS-SecureConversation
- WS-Trust
- SAML

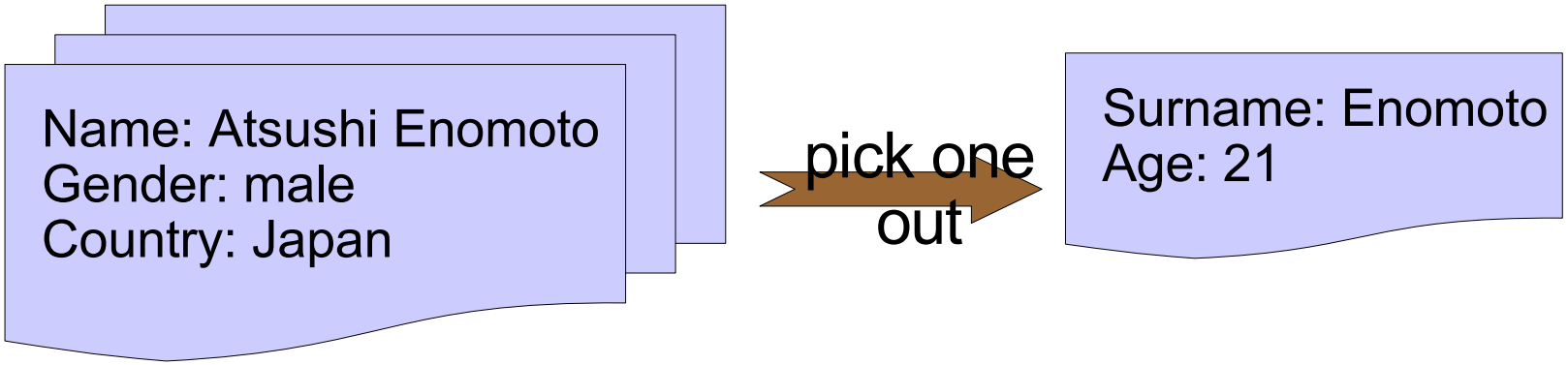
# WCS (Infocard): abstract

- an authentication framework
  - trustworthy services for users
  - trustworthy users for services
- based on the "Identity Metasystem" concept
- "claim based" authentication
- based on WS-* stack

It is somewhat similar to Web application authentication APIs such as flickr's, but WCS is designed for SOAP web services as well as for html-based web applications.

# WCS: cards

- a secure service asks its user to provide "claims" (information) which the user possesses.
- A set of claims are summarized into a "card".
- A user just selects a card to provide claims.
- Some predefined claim types such as name, email, gender, phone#, X509 thumbprint,  DNS, webpage.



# WCS: claim authorization

For service providers, some claims must be trustworthy.
("I can make this payment with this credit card")

- Identity Provider (IP)
- Security Token Service (STS)
- WS-Trust

different from MS-Passport: It is not Microsoft who can authorize a credit card validity.

# WCS: web applications usage

## Infocard in web browser

- Web applications could be Cardspace-enabled by embedding <object> element for WCS, which includes claim requirements.
- When a Cardpace-enabled web browser found it, it tries to retrieve security tokens via certain WCS API.
- Internet Explorer 7 supports it.

```
<object type="application/x-informationcard" name="xmlToken" id="xmlToken">
  <param name="tokenType"  value="
    http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1" />
  <param name="requiredClaims" value="
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier
    " />
</object>
```

# Mono Efforts

# Development History and Status

- Started from the end of Sep. 2005
- Duncan Mak, Ankit Jain, Atsushi Eno
- almost no development since Nov. 2005 until Jun. 2006 except for API updates for every beta/CTP
- Nearly 900 commits, 120000 lines of code.
- 10000 of classes/members at beginning
  -> 1000 missings and 1000 MonoTODOs now
- nothing is stable (or even usable) yet

# Tools we need

svcutil.exe

- Client proxy generation from WSDL with WS policies
  WSDL/policy generation from service contracts.

infocard.exe

- store and manage cards
- show card selector UI
- communicate with issuer Security token services

sts.exe

- Security token service implementation
- does not exist in .net 3.0 - active directory

web browser extensions for infocard.exe

configuration editor

# Current Status

## What we can do now

- Data contract serialization/deserialization.
- client and service can work with simple service contracts and data contracts on HTTP.
- XSP hosting or its own hosting (ServiceHost)
- Import and export simple WSDLs with ServiceHost.
- Some of the configuration system.

# Development Plans

## Security support

- It needs strict signing/encryption procedures described in WS-Security and WS-SecurityPolicy
- WS-SecureConversation support
- finish SAML support

## Implement our own STS

## Implement card selector UI

## There is a lot of small or big tasks to do

- Duplex channel support, Session support, WS-Transactions, more configuration support, TCP transport, P2P transport, Policy export/import, non-text Xml ...