Programming Mobile Devices

Remote Procedures and Web services

University of Innsbruck

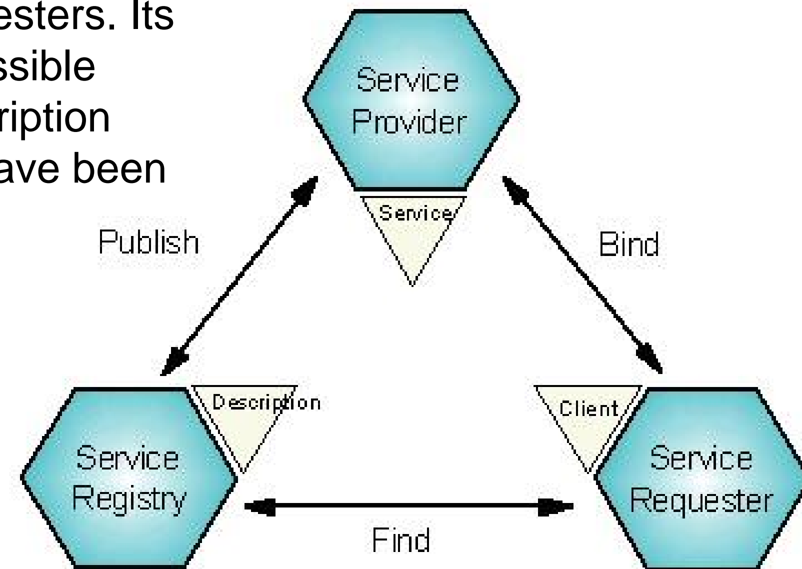
WS 2009/2010

thomas.strang@sti2.at

STI · INNSBRUCK

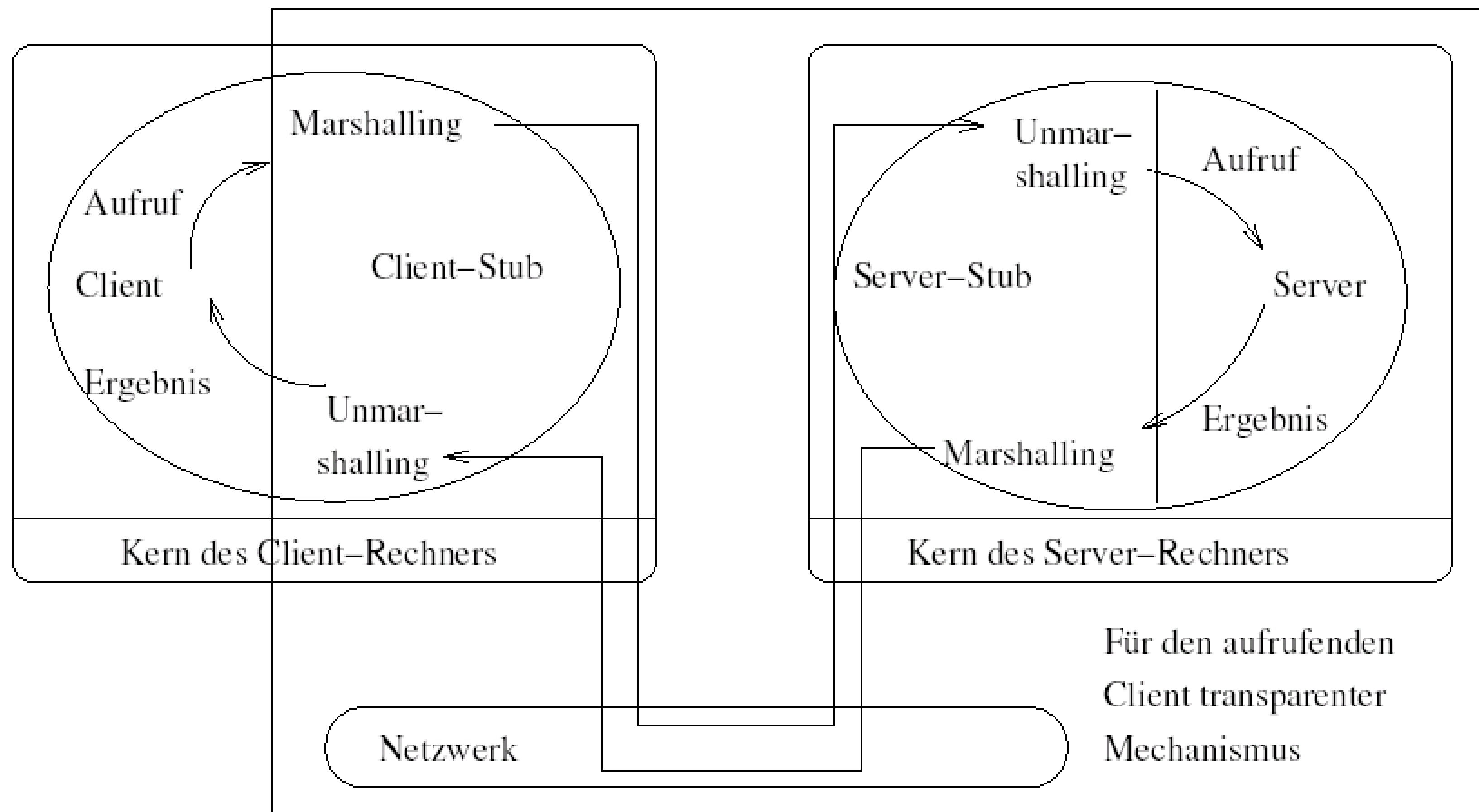
Distributed Service Interaction

- **Service**
  The application being provided for use by requesters. Its implementation is deployed on a network accessible platform. It is described through a service description language. Its description and access policies have been published to a registry.



- **Service Provider**
  From a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that provides access to the service.
- **Service Registry**
  This is a searchable repository of service descriptions where service providers publish their services and service requesters find services and obtain binding information for services.
- **Service Requestor**
  From a business perspective, this is the business that requires certain function to be fulfilled. From an architectural perspective, this is the application or client that is looking for and invoking a service.

The core: Remote Procedure Call (RPC)



RPC – marshalling/unmarshalling

- Only *serialized* objects may be transferred through the network
- Marshalling
  - serialization of input parameters or output parameters (result)
- Unmarshalling
  - de-serialization
- Client and server have to use a common serialization format (data structure) – this is where XML came in for Web Services

RPC in J2SE: RMI

- RMI stands for Remote Method Invocation
- RMI applications are comprised of two separate programs: a server and a client
  - Server – creates remote objects, makes references to them accessible and waits for clients to invoke methods on these remote objects
  - Client - gets a remote reference to one or more remote objects in the server and then invokes methods on them
- RMI application is sometimes referred to as a *distributed object application*

Distributed object applications

- *Locate remote objects*
  - the rmiregistry - register remote objects with RMI's simple naming facility,
  - the application can pass and return remote object references as part of its normal operation.
- *Communicate with remote objects* – everything is hidden – to the programmer remote communication looks like a standard operation invocation.
- *Load class bytecodes for objects that are passed around*

Dynamic Code Loading

- Download the *bytecode* of an object's class if the class is not defined in the receiver's virtual machine
- RMI passes objects by their true type – the behaviour of these objects is not changed when they are sent to another virtual machine
- Change the behaviour of an application dynamically, add new functionality in distributed application

Remote Interfaces, Objects, and Methods

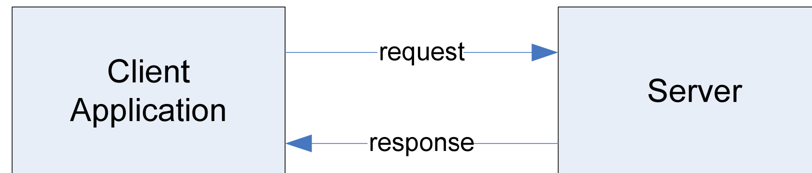
- A remote interface extends the interface `java.rmi.Remote`
- Each method of the interface declares
  `java.rmi.RemoteException`
- Instead of making a copy of the implementation object in the receiving virtual machine, RMI passes a remote *stub* for a remote object
- Stub acts as the local representative, or proxy, for the remote object
- The caller invokes a method on the local stub, which is responsible for carrying out the method call on the remote object.

Crash course - Creating Distributed Applications Using RMI

- Step 1 – design and implement the components of distributed application
  - Define the remote interfaces
  - Implement the remote objects
  - Implement the clients
- Step 2 – Compile sources and generate stubs
  - javac compiler to compile the source files
  - rmic compiler to create stubs for the remote objects
- Step 3 – Make classes network accessible
  - Use Web server to make available class files associated with the remote interfaces, stubs, and other classes that need to be downloaded to clients
- Step 4 – Start the application
  - Start the RMI remote object registry, the server, and the client

Writing an RMI Server

- The server accepts tasks from clients, runs the tasks, and returns any results
- The interface provides the definition for the methods that can be called from the client. We can call it also the client's view of the remote object.
- The class provides the implementation.



Implementing a Remote Interface

- Declare the remote interfaces being implemented
- Define the constructor for the remote object
- Provide an implementation for each remote method in the remote interfaces

Setup procedure

- The server needs to create and to install the remote objects. The setup procedure should
  - Create and install a security manager
  - Create one or more instances of a remote object
  - Register at least one of the remote objects with the RMI remote object registry (or another naming service such as one that uses JNDI), for bootstrapping purposes

UnicastRemoteObject

- It is a convenience class
- Remote Object does not have to extend UnicastRemoteObject
- Defined in the RMI public API
- This superclass supplies implementations for a number of java.lang.Object methods:
  - equals,
  - hashCode,
  - toString
- It includes constructors and static methods used to *export* a remote object, that is, make the remote object available to receive incoming calls from clients.

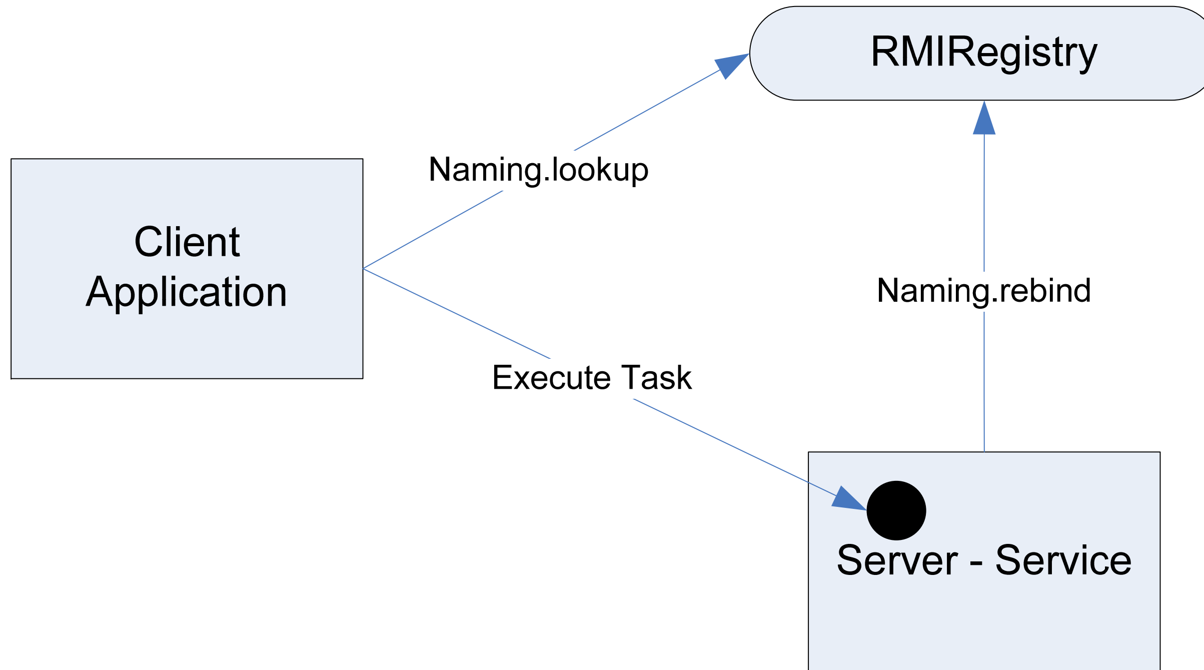
Security Manager

- Protects access to system resources from mistrusted downloaded code running within the virtual machine
- Determines whether downloaded code has access to the local file system or can perform any other privileged operations
- All programs using RMI must install a security manager, or RMI will not download classes
- `RMISecurityManager` – very strict about what is allowed (similar to applets); but there are other security managers which allow for more freedom

Make the Remote Object Available to Clients

- `InterfaceDef object = new InterfaceImpl();`
  - interface available to clients
- RMIRegistry – simple remote object name service; caller must first obtain a reference to the remote object
- `java.rmi.Naming` interface is used as a front-end API for binding, or registering, and looking up remote objects in the registry
  - `String name = "//host/SampleObject";`
  - `Naming.rebind(name, object);`

Client program



Web Services

*A **Web Service** is a software system identified by an URI whose public interfaces and bindings are defined and described by XML.* [Boot et al., 2003]

Web Services

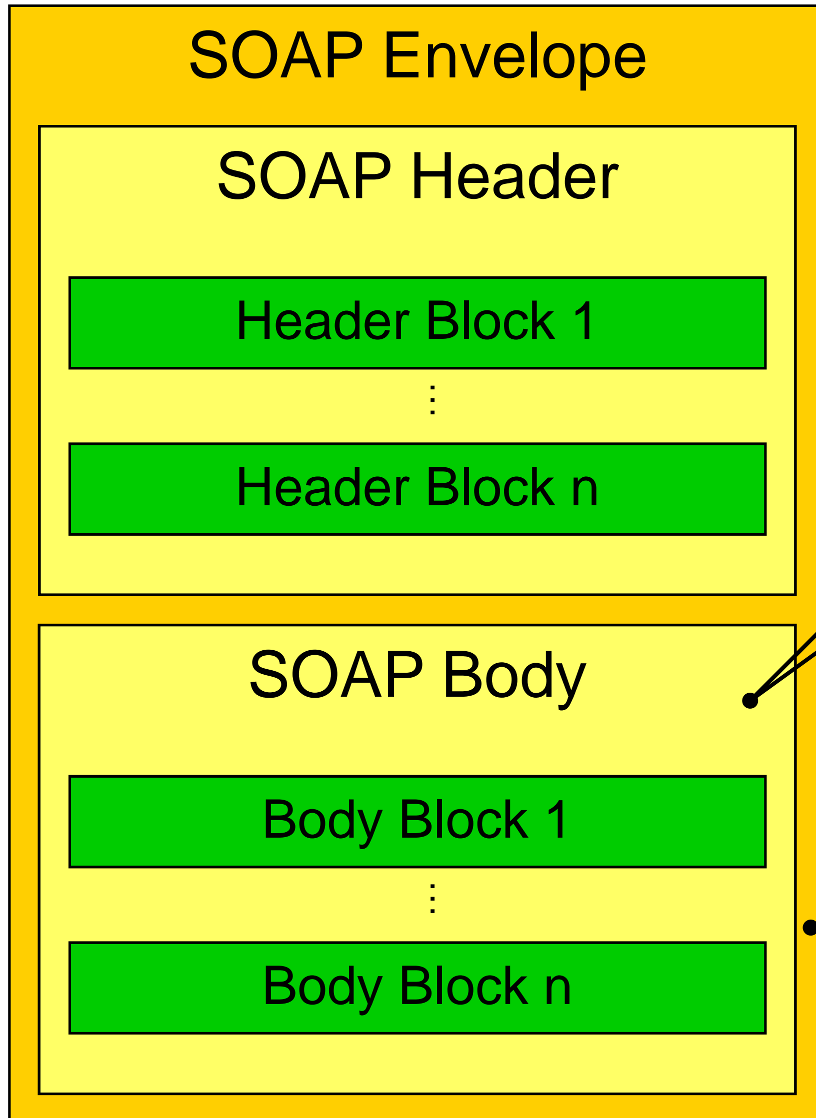
Web Services are self-contained, modular applications that can be:

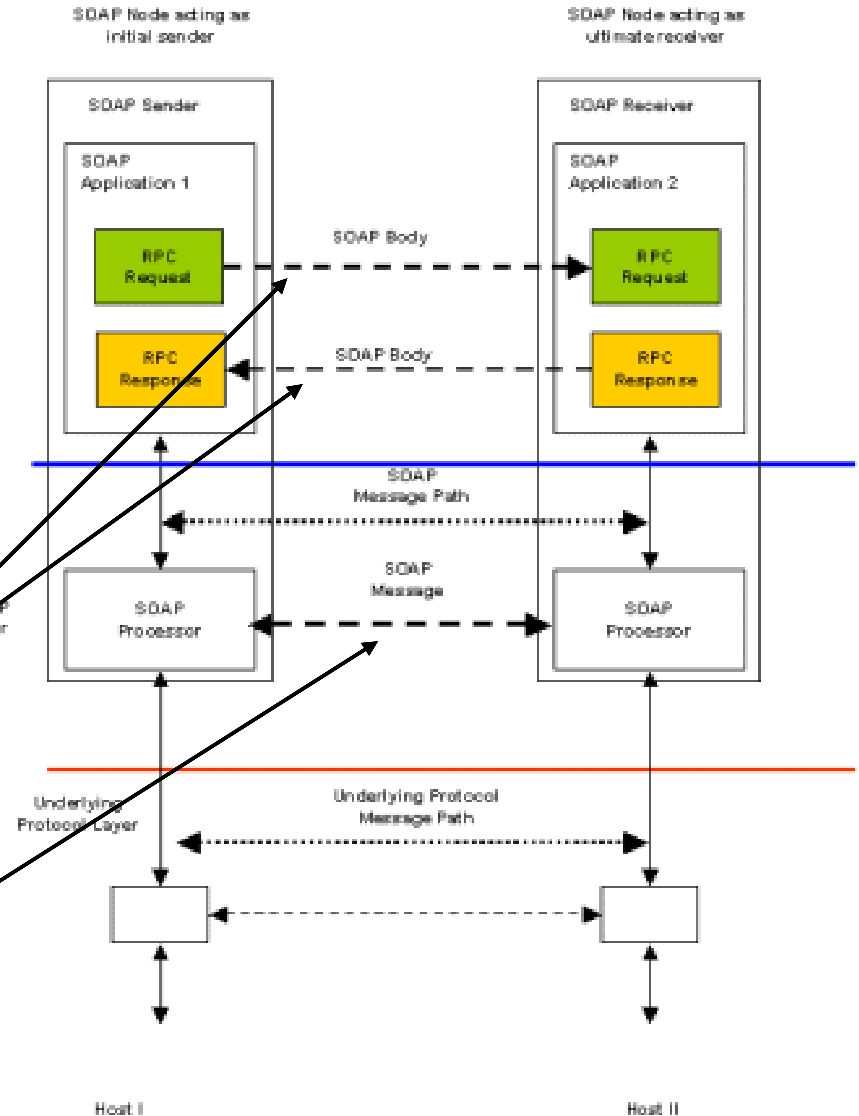
- **Described** using a service description language, i.e. WSDL (Web Services Description Language).
- **Published** by registering their descriptions and use policies with a well-known registry, i.e. the UDDI (Universal Description, Discovery and Integration) registry.
- **Found** by sending queries to that registry and receiving the binding details of the service(s) that fit the parameters of the query.
- **Bound** by using the information contained in the service description to create a callable service instance or proxy.
- **Invoked** over a network by using the information contained in the binding details of the service description.
- **Composed** with other services into new services and applications.

Applied to the Web: SOAP

- SOAP is successor of XML-RPC
- SOAP 0.9, 1.0, 1.1 – 2000
  - Simple Object Access Protocol
- SOAP 1.2 – June 2003
  - No longer acronym for
    Simple Object Access Protocol
- Marshalling/Unmarshalling → SOAP-Envelope
- Networking → SOAP-Binding (HTTP, Email)

SOAP-Envelope / RPC





Example

- Java Function

```
Integer getSunBurnTime(String lat, String lon, String hhmm_utc)
```

- SOAP Request

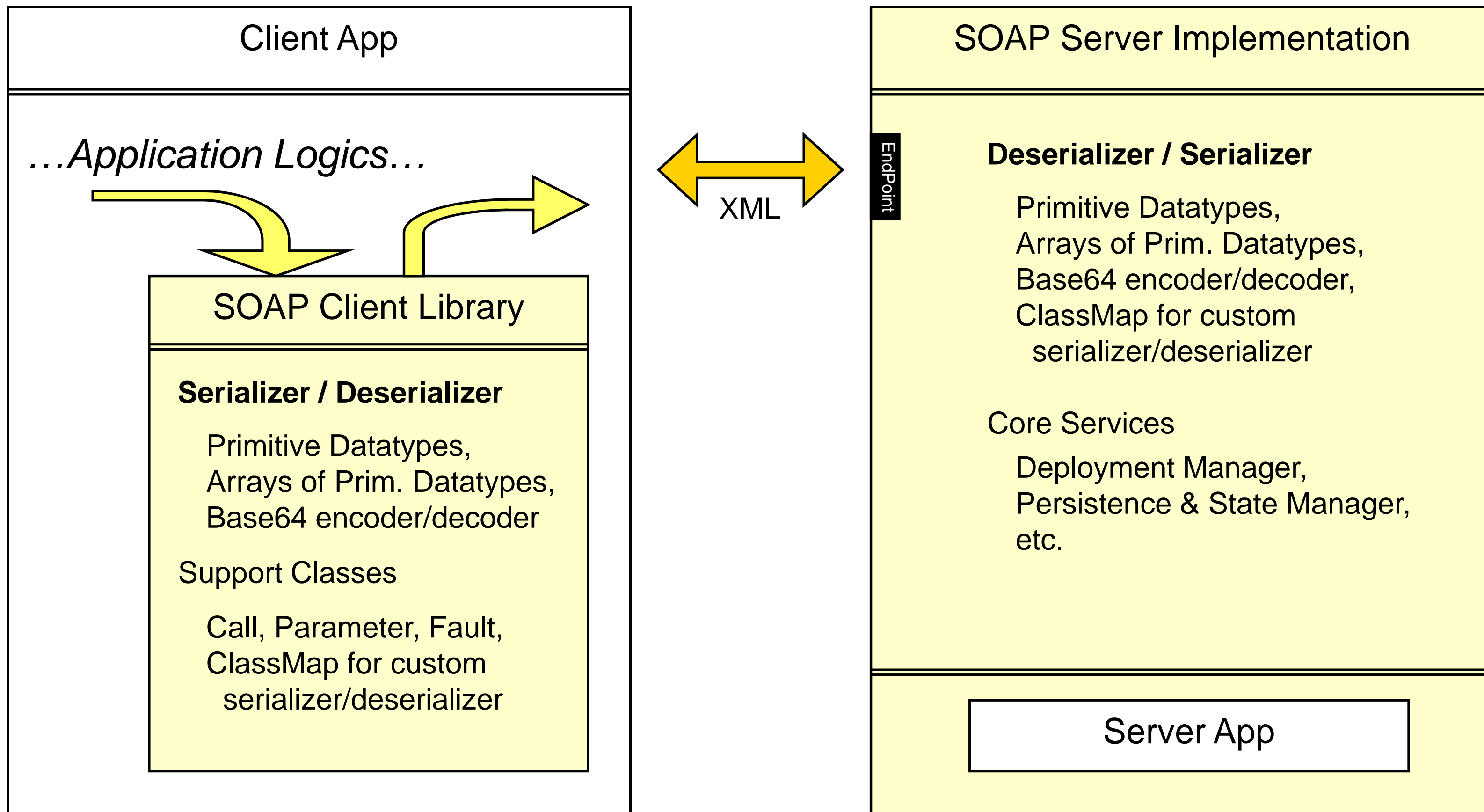
```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <ns1:getSunBurnTime xmlns:ns1="urn:heywow-uv-v4"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <lat xsi:type="xsd:string">48.0</lat>
      <lon xsi:type="xsd:string">11.7</lon>
      <hhmm_utc xsi:type="xsd:string"></hhmm_utc>
    </ns1:getSunBurnTime>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Example

- SOAP Response

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <ns1:getSunBurnTimeResponse xmlns:ns1="urn:heywow-uv-v4"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="xsd:int">45</return>
    </ns1:getSunBurnTimeResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Serializer/Deserializer (Marshaller/Unmarshaller)



Coding Steps (Example: J2SE + Apache SOAP)

- Step 1: Service Implementation

```
public class UVService
{
    public UVService() { /* .. */ } // unconditioned constructor must exist

    public Integer getSunBurnTime(String lat, String lon, String hhmm_utc)
    throws RuntimeException
    {
        // .. do some magic calculations
        return calculatedResult;
    }

    // you may test the service in main() if manually started
    public static void main( String args[] )
    {
        try {
            UVService service = new UVService();
            System.out.println("Today's SunBurnTime in Munich at noon: " +
                service.getSunBurnTime( "48.1", "11.5", "1200" ));
        } catch (Exception e) {
            System.out.println( e.getMessage() );
        }
    }
}
```

Coding Steps (Example: J2SE + Apache SOAP)

- Step 2: Deployment (e.g. through Web-Interface)

Apache SOAP Admin Tool - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Suchen Favoriten Medien

Adresse http://uv.heywow.com:8080/soap/admin/index.html Wechseln zu Links

Google Web-Suche Site-Suche 6 blockiert Optionen

Apache SOAP Admin

List

Deploy

Un-deploy

Deploy a Service

Service Deployment Descriptor Template

Property	Details
ID	urn:heywow-uv-v4
Scope	Request
Methods	getPosition getSunBurnTime getUVMatrix (Whitespace separated list of method names)
Provider Type	Java
	For User-Defined Provider Type, Enter FULL Class Name:
	Number of Options: Key Value
Java Provider	Provider Class com.heywow.uv.UVService Static? No

Internet

Methods to deploy

Implementation

Coding Steps (Example: J2SE + Apache SOAP)

- Step 3: Client Implementation (incl. Wrapper)

```
Integer getSunBurnTime(String lat, String lon, String hhmm_utc)
throws RuntimeException
{
    // Create and prepare the RPC.
    Call call = new Call();  // call object specific to J2SE Apache SOAP!
    call.setTargetObjectURI( "urn:heywow-uv-v4" );
    call.setMethodName( "getSunBurnTime" );

    Vector params = new Vector ();
    params.addElement ( new Parameter("lat", String.class, lat, null) );
    params.addElement ( new Parameter("lon", String.class, lon, null) );
    params.addElement ( new Parameter("hhmm_utc", String.class, "", null) );
    call.setParams ( params );

    // Invoke the RPC.
    URL endpoint = new URL( "http://app1.dlr-kn.de:8080/soap/servlet/rpcrouter" );
    String actionURI = "getSunBurnTime";
    Response resp = call.invoke ( endpoint, actionURI );

    // Check the response.
    if (resp.generatedFault ()) {
      Fault fault = resp.getFault(); throw new RuntimeException( fault.toString() );
    } else {
      Parameter result = resp.getReturnValue ();
      return (Integer)result.getValue();
    }
}
```

RPC in J2ME: kSOAP

- http://kSOAP.org
- following J2ME examples refer to kSOAP version 2
- binding to HTTP is encapsulated in class `HttpTransport`
- equivalent to `Call` class in J2SE is the `SoapObject` in kSOAP for J2ME

Same as Step 3 in previous example, here kSOAP

```
import org.ksoap.*;
import org.ksoap.transport.*;
import org.kobjects.serialization.*;

//…
SoapObject rpc = new SoapObject( "urn:heywow-uv-v4",
                                 "getSunBurnTime" );

rpc.addProperty( "lat", new ElementType(lat), lat);
rpc.addProperty( "lon", new ElementType(lon), lon);
rpc.addProperty( "hhmm_utc", new ElementType(hhmm_utc), hhmm_utc);

HttpTransport trans = new HttpTransport(
    "http://app1.dlr-kn.de:8080/soap/servlet/rpcrouter",
    "urn:heywow-uv-v4#getSunBurnTime");

ClassMap classMap = new ClassMap(true); // use 1999 schema
classMap.implicitTypes = true;
// possibly add mappings to own serializers/deserializers
trans.setClassMap( classMap );

Integer sbt = (Integer)trans.call( rpc );
```

"Primitive" Datatypes

- Datatypes with inherent SOAP encoding (existing serializer / deserializer):
  - Bean
  - Boolean
  - Byte
  - Calendar
  - Date
  - Decimal (BigInt)
  - Double
  - Float
  - Hashtable
  - Int
  - Long
  - Map
  - Mime
  - QName
  - Short
  - String

[according to Apache SOAP]

- Special serializer / deserializer
  - Array (also used for Vector)
  - Base64 (for binary encoding)

Own Datatypes require own Ser-/Deserializer

- Java Function

```
UVMatrix getUVMatrix(String lat, String lon, String hhmm_utc)
```

- SOAP Request

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <ns1:getUVMatrix xmlns:ns1="urn:heywow-uv-v4"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <lat xsi:type="xsd:string">48.0</lat>
      <lon xsi:type="xsd:string">11.7</lon>
      <hhmm_utc xsi:type="xsd:string"></hhmm_utc>
    </ns1:getUVMatrix>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Own Datatypes require own Ser-/Deserializer

- SOAP Response

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope <!-- ..namespaces.. --> >

  <SOAP-ENV:Body>
    <ns1:getUVMatrixResponse xmlns:ns1="urn:heywow-uv-v4"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="ns1:UVMatrix">
        <hhmm_utc xsi:type="xsd:string">1317</hhmm_utc>
        <delta_ss xsi:type="xsd:string">250</delta_ss>
        <skinVeryBright xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
            xsi:type="ns2:Array" ns2:arrayType="xsd:string[3]">
          <item xsi:type="xsd:string">36</item>
          <item xsi:type="xsd:string">90</item>
          <item xsi:type="xsd:string">241</item>
        </skinVeryBright>
        <skinBright ...> ... </skinBright>
        ...
      </return>
    </ns1:getUVMatrixResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Own Datatypes require own Ser-/Deserializer

- Serializer
  - (in this case) on SOAP server side
  - UVMatrix object → XML
- Deserializer
  - (in this case) on SOAP client side
  - XML → UVMatrix object
- What needs to be provided:
  - Mapping to Serializer (at server) and Deserializer (at client)
  - Own object classfile (in case of Java)
    - (at least) empty constructor
    - Special instantiation by Class.forName("NameOfObject")

Array Encoding Example

- Java Function

```
Position[] getPosition(String cityname)
```

- SOAP Request

```
<SOAP-ENV:Envelope xmlns:n0="urn:heywow-cityposition"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <n0:getPosition id="o0" SOAP-ENC:root="1">
      <cityname xsi:type="xsd:string">Breinig</cityname>
    </n0:getPosition>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Array Encoding Example

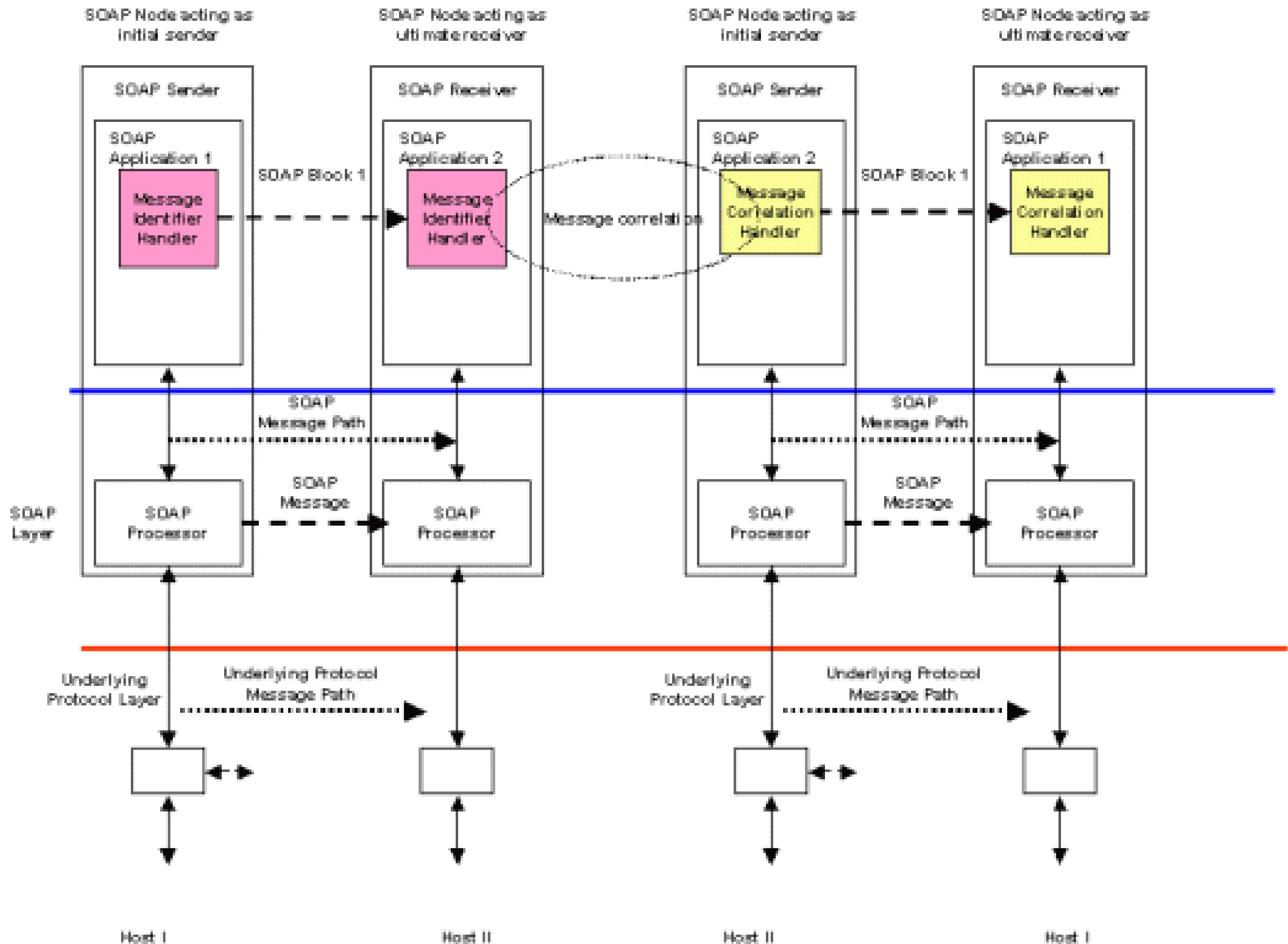
- SOAP Response

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <ns1:getPositionResponse xmlns:ns1="urn:heywow-cityposition"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
          xsi:type="ns2:Array" ns2:arrayType="ns1:Position[1]">
        <item xsi:type="ns1:Position">
          <id xsi:type="xsd:string">eu-id:214213</id>
          <name xsi:type="xsd:string">D-52223 Stolberg/Breinig</name>
          <lat xsi:type="xsd:string">50.7306</lat>
          <lon xsi:type="xsd:string">6.22054</lon>
          <timezone xsi:type="xsd:string">Europe/Berlin</timezone>
        </item>
      </return>
    </ns1:getPositionResponse>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

SOAP Message Exchange Pattern

- SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information
- SOAP defines two MEPs:
  - SOAP Request-Response MEP
    - RPC like
  - SOAP Response MEP
    - defines a pattern for the exchange of a non-SOAP message acting as a request followed by a SOAP message acting as a response

Message Correlation



SOAP Message Exchange Pattern

- Both MEPs are not session persistent
  - Additional session handling required, e.g. by adopting transport sessions (HTTP cookies)
- Receiving SOAP node may not be equal to the ultimate recipient of a SOAP message
  - in this case the receiving SOAP node is called a SOAP intermediate
  - message is said to be "routed" through one or more SOAP intermediaries which act in some way on the message (e.g. log, audit, modify etc.)
  - handling depends on specific "role" of a node

SOAP Headers

- The *Header* has three predefined optional attributes:
  - **`Role`** (called *actor* in SOAP 1.0 and 1.1): Determines if a node should process a particular header (*see next slide*)
  - **`mustUnderstand`**: If set to "true", the node must know how to process the header
    - new in SOAP 1.2: matching `env:NotUnderstood` Fault
  - **`Relay`**: Indicates whether or not an unprocessed header block should be forwarded
- Example use of Headers:
  - Session State Support: Many services require several steps and so will require maintenance of session state.
    - Equivalent to cookies in HTTP.
    - Put session identifier in the header.
  - Security: WS-Security places additional security information (like digital signatures and public keys) in the header.

Predefined Roles and Custom Roles of Nodes

- For a particular message, the node can act in one or more roles
- You can define your own roles, e.g.
  - “Log message” role
  - “Check authorization” role

Short-name	Role Label (URI)	Description
next	"http://www.w3.org/2003/05/soap-envelope/role/next"	Each SOAP intermediary and the ultimate SOAP receiver MUST act in this role.
none	"http://www.w3.org/2003/05/soap-envelope/role/none"	SOAP nodes MUST NOT act in this role. That is, the header block should not be directly processed. It may carry supplemental information.
ultimateReceiver	"http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"	The ultimate receiver MUST act in this role. If no role is specified in a header, it is treated as being in this role.

SOAP Bindings

- Generic protocol binding framework
- Default: SOAP HTTP Binding
  - Media type `application/soap+xml`
  - Use of HTTP GET and POST methods
  - `SOAPAction` processing shortcut
    - mandatory in SOAP 1.0, optional in SOAP 1.1
    - deprecated in SOAP 1.2
  - Broad use of HTTP error codes
- Alternative Bindings: Email

SOAP HTTP Binding Example

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: uv.heywow.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 513
SOAPAction: "getUVMatrix"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
…
```

SOAP Faults

- Always in the Body
- Extensible fault code
  - Version mismatch
  - Problems with understanding headers or data encodings
  - Generic sender/receiver problems
- Human-readable fault reason
- Detail element

SOAP Fault Example

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope>
  <env:Header>
    <env:NotUnderstood qname="t:transaction"
      xmlns:t="http://thirdparty.example.org/transaction" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">Header not understood</env:Text>
        <env:Text xml:lang="de">Kopfelement unbekannt</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Descriptive Repository

www.xmethods.net - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück · Suchen · Favoriten · Medien

Adresse http://www.xmethods.net/ · Wechseln zu · Links

Google · Web-Suche · Site-Suche · 6 blockiert · Optionen

X METHODS

Home · Interfaces · Tools · Implementations · Manage · Register · Tutorials · Mailing List · About

Welcome to XMethods.

Emerging web services standards such as SOAP, WSDL and UDDI will enable system-to-system integration that is easier than ever before. This site lists publicly available web services.

Programmatic Interfaces

Access XMethods through a variety of interfaces:

- UDDI v2
- WS-Inspection
- RSS
- SOAP
- DISCO

Read about the TRY IT feature.

Recent Listings [View the FULL LIST]

Publisher	Style		Service Name	Description	Implementation
findpeoplefree	DOC	Try It	Find People Free .co.uk	Lookup businesses and people in the UK	
msumerano	DOC	Try It	Bible XML Web Service	Full American Standard 1901 Bible with Dictionary and Random Verse	MS .NET
BosNewsLife	DOC	Try It	BosNewsLife Information Services	Search the latest news stories from the BosNewsLife archive	MS .NET
rhelena	DOC	Try It	Caribbean Tourism Statistics	Get data on tourist activity in the Caribbean for the last 10 years	MS .NET
walterjones	DOC	Try It	Real Time Quote and Market Data	This Web Service offers various sets of data designed for all market participants - from the individual investor to market professionals.	MS .NET
walterjones	DOC	Try It	USA Weather Forecast	Get one week weather forecast for valid zip code or Place name in USA	MS .NET
walterjones	DOC	Try It	London Gold And Silver Fixings (Real Time)	Real Time quotes for the London Gold and Silver Fixings	MS .NET

Internet

Exercise 12

- Go to www.xmethods.net and look for the service „*getRandomBushism*“
- Activate and use JSR-172 (WSA) on your WTK
- Implement a MIDlet acting as a web service client for the *getRandomBushism* service, showing the next random quote on the display