Distributed computing technologies and protocols

2004 CERN School of Computing,

Vico Equense

Andreas Pfeiffer

CERN, PH / SFT

Distributed computing technologies and protocols

Distributed computing technologies and protocols

- Definition of Web Services
- Architecture of Web Services
- XML-RPC
- SOAP
- WSDL

Distributed computing technologies and protocols

- Will use generic term “Web services”
  - Although there is a more specialized definition from W3C
    - Requires SOAP and WSDL
- Allow for cross platform interoperability
  - “The Internet is the platform”

Web Services

- Web/network interface to application
  - Independent of language of implementation
- Using XML for information exchange
  - For both: methods and data
- Kind of “Remote Procedure Call” using XML
- SOAP needs a rather complex “infrastructure”
  - Where, what and how to find
- XML-RPC is more simple, less heavy

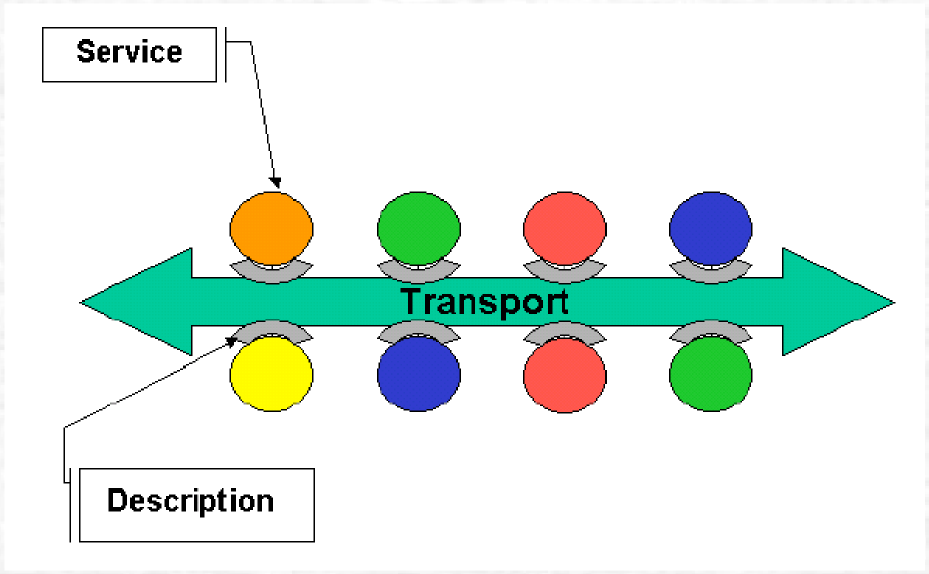
W3C on Web Services

- *"Definition: A Web service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."*

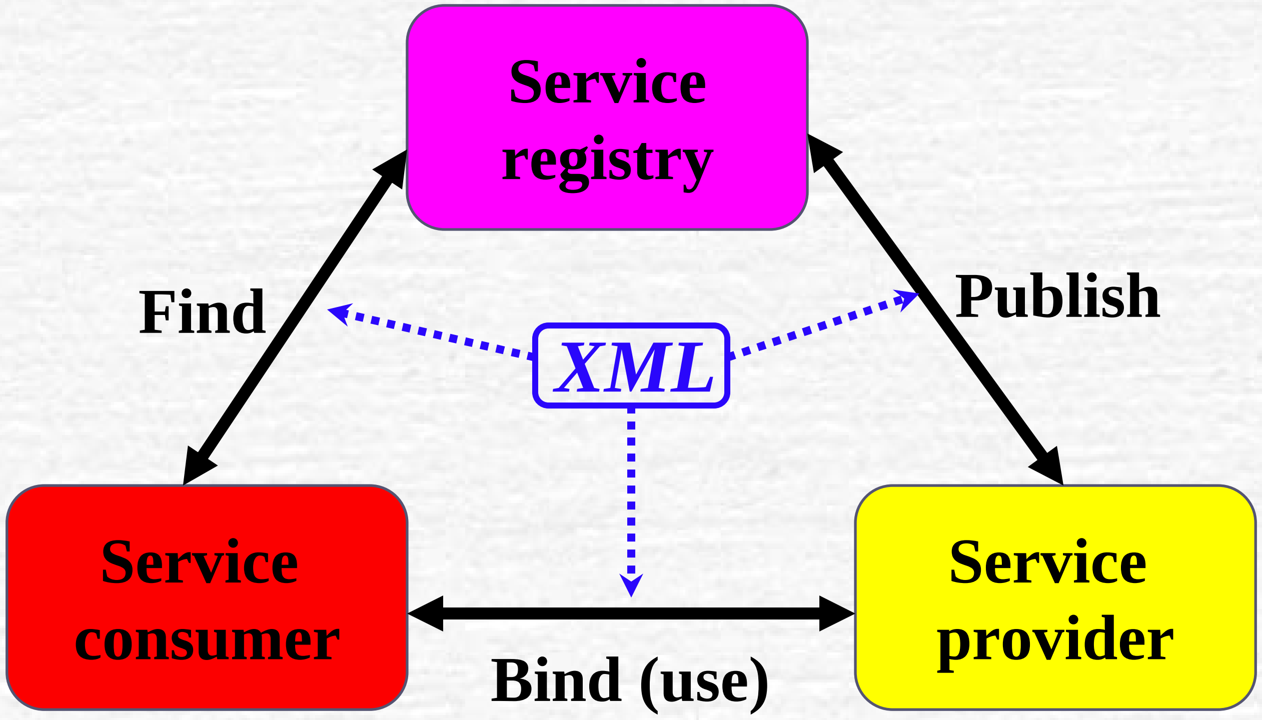
Agents and Services

- A distributed system, consists of discrete software agents that must work together to implement some intended functionality
- Agents implement a service

Generic Service Oriented Architecture Diagram



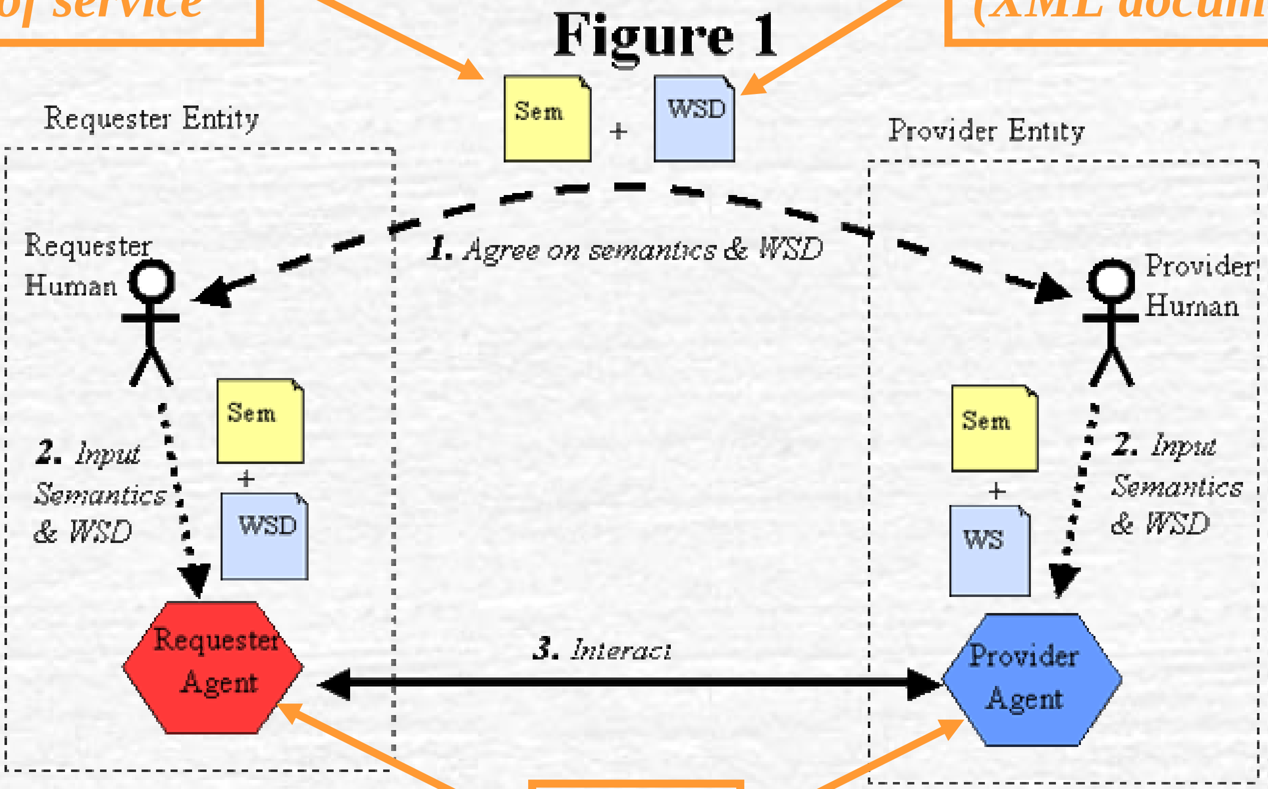
Architecture of Web Services (I)



Architecture of Web Services (II)

Semantics of service

Web Service Description (XML document)



Agents

Roles of the agents

- Service requestor
- Service provider
- Discovery agency
- Are not fixed, a given agent can “play” several roles

Calling a procedure on a remote system

- Needs
  - A procedure (with agreed semantics)
  - Arguments to the procedure
  - Return values from the procedure
  - Remote system where the procedure is implemented/running
  - An agreement on how to communicate

Remote procedure calls

- RPC
  - Since early 1980’s in unix world
  - eXternal Data Representation (XDR) to communicate values
  - Specific server/client models
  - CORBA and DCOM
- Enter XML
  - **XML-RPC**
  - **SOAP**
  - Late 1990’s (parallel development)

Will be discussed in more detail later

XML-RPC

- **http://www.xmlrpc.org/**
- *“It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.”*

XML-RPC

- Is a Remote Procedure Call protocol
  - Working over the Internet
- Using HTTP as the transport layer
  - An XML-RPC message is an HTTP-POST request
- And XML as the encoding
  - The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML.
  - Procedure parameters can be scalars, numbers, strings, dates, etc.; and can also be complex record and list structures.

XML-RPC goals

- Discoverability
  - *“We wanted a clean, extensible format that's very simple. It should be possible for an HTML coder to be able to look at a file containing an XML-RPC procedure call, understand what it's doing, and be able to modify it and have it work on the first or second try. “*
- Easy to implement
  - *“We also wanted it to be an easy to implement protocol that could quickly be adapted to run in other environments or on other operating systems.”*

From: http://www.xmlrpc.org/spec

XML-RPC example

HTTP POST request

Content-length must be correct

Body of the request

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
 <methodCall>
   <methodName> examples.getStateName </methodName>
     <params>
       <param> <value> <i4> 41 </i4> </value> </param>
     </params>
 </methodCall>
```

XML-RPC Basic Types

Tag	Type	Example
`<i4> or <int>`	Four-byte signed integer	42
`<boolean>`	0(false) or 1(true)	1
`<string>`	string	Hello world
`<double>`	Double-precision signed	-3.14.15926
`<dateTime.iso8601>`	Date/time	20030716T09:53:42
`<base64>`	Base64-encoded binary	eW91IGNhbid0IHJlYWQgdGhpcyE=

XML-RPC <struct>

```
<struct>
  <member>
    <name> lowerBound </name>
    <value> <i4> 18 </i4> </value>
  </member>
  <member>
    <name> upperBound </name>
    <value> <i4> 139 </i4> </value>
  </member>
</struct>
```

structs contain members,
members have name and value

- <struct>s can be recursive, any <value> may contain a <struct> (or <array>)

XML-RPC <array>

```
<array>
  <data>
    <value> <i4> 42 </i4> </value>
    <value> <string> Egypt </string> </value>
    <value> <boolean> 0 </boolean> </value>
    <value> <i4> -31 </i4> </value>
  </data>
</array>
```

arrays contain data,
data contains value(s),
array elements have no names

- <array>s can be recursive, any <value> may contain an <array> (or <struct>)

Response example

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value> <string>South Dakota</string> </value>
      </param>
    </params>
  </methodResponse>
```

Fault-Response example

```
[HTTP header …]
<?xml version="1.0"?>
  <methodResponse>
    <fault>
      <value>
       <struct>
         <member>
           <name>faultCode</name>
           <value> <int>4</int></value>
         </member>
         <member>
            <name>faultString</name>
            <value><string>Too many parameters.</string></value>
         </member>
       </struct>
     </value>
   </fault>
 </methodResponse>
```

fault contains a value, which is a struct with two elements:

- one int member named faultCode and

- one string member named faultString

XML-RPC extensions

- Multicall
  - Problem with HTTP round-trip times (latency)
  - Solution: group requests/responses in arrays and use only one call (“boxcarring”)
    - Proposal to add to XML-RPC by Eric Kidd
- Server side introspection
  - system.listMethods
  - system.methodSignature
  - system.methodHelp

SOAP

- Developed in parallel to XML-RPC
  - Started by UserLand and Microsoft developers (1998)
  - Now mainly Microsoft and IBM
- SOAP vs. XML-RPC
  - User defined data types
  - Able to specify the recipient
  - Message specific processing control
- Extensive use of namespaces and attribute specification tags in almost every element of a message

SOAP data types (I)

- Same basic types as for XML-RPC
  - int, boolean, double, string, date/time, base64
- References (to the same object in memory)

  -
    ```
    <value xsi:type="xsd:int" id="v1"> 42 </value>
    <value href="#v1" />
    ```
- Structs
  - SOAP structs define a set of name value pairs. Structs can be named.

SOAP Arrays

- SOAP arrays define a grouping of elements with no limitation mixing data types like integers and strings within the same array. Arrays can be named.
  - Access by ordinal position in the group (structs by name)
  - ArrayType attribute to specify which types occur where in the array
  - Multidimensional arrays possible
  - Handling of sparse arrays

SOAP Array Examples

1-dim, 3 entries

```
<someArray xsi:type=“SOAP-ENC:Array”
                     SOAP-ENC:arrayType=“se:string[3]”>
  <se:string> Joe   </se:string>
  <se:string> John  </se:string>
  <se:string> Louis </se:string>
</someArray>
```

2-dim, sparse: 2 entries

```
<names xsi:type=“SOAP-ENC:Array”
                     SOAP-ENC:arrayType=“xsd:string[10,10]”>
  <name SOAP-ENC:position=“[2,5]”> Guido </name>
  <name SOAP-ENC:position=“[4,2]”> Jim   </name>
</names>
```

SOAP data types (II)

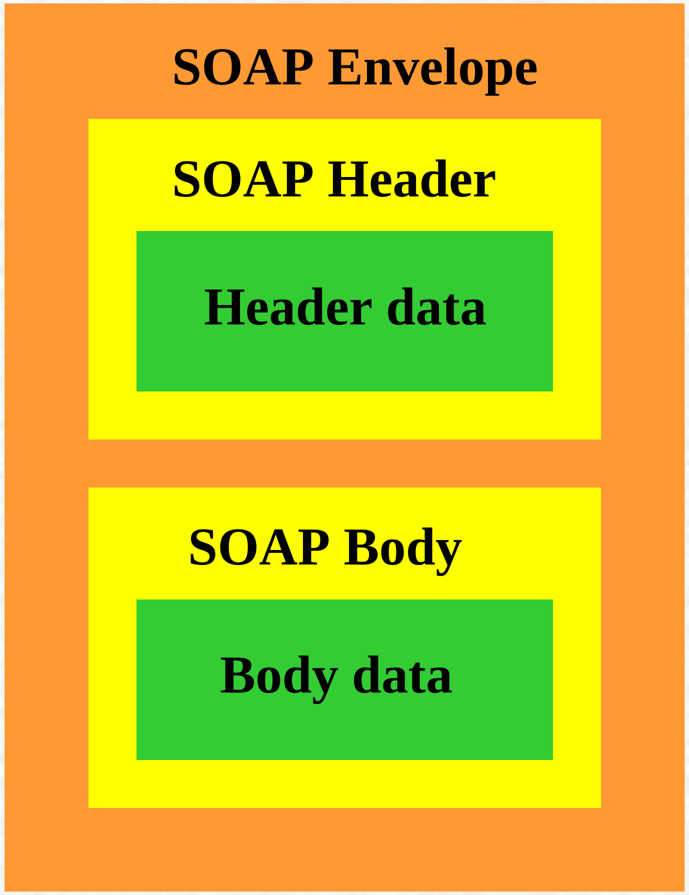
- Array of Bytes
  - Rules for an array of bytes are similar to those for a string.
  - Containing element of the array of bytes value MAY have an "id" attribute. Additional accessor elements MAY then have matching "href" attributes."
- Enumerations
  - A list of distinct values appropriate to the base type
  - All simple types except boolean.
  - "XML Schema Part 2: Datatypes" **http://www.w3.org/TR/xmlschema-2/**

SOAP data types (III)

- Polymorphic Accessors
  - An accessor "...that can polymorphically access values of several types, each type being available at run time. A polymorphic accessor instance MUST contain an "xsi:type" attribute that describes the type of the actual value."
    - <cost xsi:type="xsd:float">29.95</cost>
- User Defined Data-Types
  - Developers can define their own simple, or complex, data types.

SOAP envelope

- Structure of a SOAP message
- Header
  - Optional
  - Information on how the message is to be processed
- Body
  - Required
  - Contains actual message to be delivered



SOAP example

```
<env:Envelope xmlns:env=“http://www.w3.org/2003/05/soap-envelope”>
  <env:Header>
   <n:alertcontrol xmlns:n=“http://example.org/alertcontrol”>
     <n:priority>1</n:priority>
     <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m=“http://example.org/alert”>
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

SOAP additional features

- Control of routing
  - “role”s in headers, “mustUnderstand” flags
  - Nodes may modify the header blocks (or add new ones)
  - Allows for encryption/authentication of messages
- Bindings to various protocols
  - HTTP
    - Post *and* Get methods
  - E-mail
  - RPC

WSDL

- **W**eb **S**ervice **D**escription **L**anguage
- Describes the abstract interface of a web service and the details how a specific web service has implemented it
  - *"WSDL defines an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication."*

WSDL Service (I)

- Services are defined using six major elements:
  - **types**, which provides data type definitions used to describe the messages exchanged.
  - **message**, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
  - **portType**, which is a set of abstract operations. Each operation refers to an input message and output messages.

WSDL Service (II)

- **binding**, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
- **port**, which specifies an address for a binding, thus defining a single communication endpoint.
- **service**, which is used to aggregate a set of related ports.

WSDL Interface

```
<definitions …>
  <wsdl:message name="sayHello_IN">
    <part name="name" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="sayHello_OUT">
    <part name="greeting" type="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="HelloWorldInterface">
    <wsdl:operation name="sayHello">
      <wsdl:input  message="tns:sayHello_IN" />
      <wsdl:output message="tns:sayHello_OUT" />
    </wsdl:operation>
  </wsdl:portType>
</definitions>
```

WSDL Binding the Interface to an Implementation

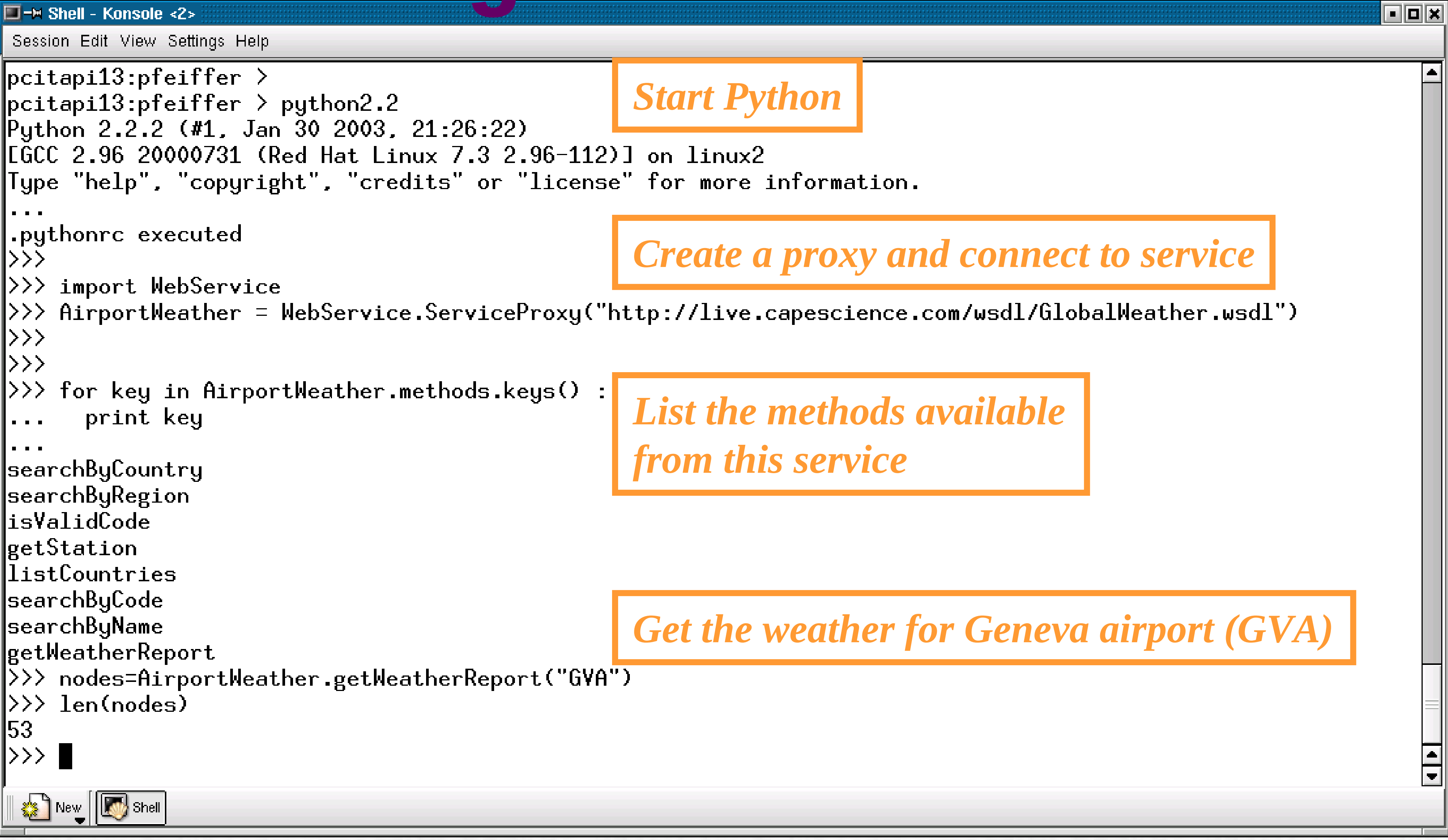
```
<wsdl:binding name="HelloWorldBinding"
              type="tns:HelloWorldInterface">
  <soap:binding style="rpc"
        transport=http://schemas.xmlsoap.org/soap/http/>

  <wsdl:operation name="sayHello">
    <soap:operation soapAction="urn:Hello" />
    <wsdl:input>
      <soap:body use="encoded"
            namespace="…" encodingStyle="…" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="encoded"
            namespace="…" encodingStyle="…" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

WSDL Linking the Binding to a network address

```
<wsdl:service name="HelloWorldService">

  <wsdl:port name="HelloWorldPort"
        binding="tns:HelloWorldBinding">
    <soap:address location="http://localhost:8080" />
  </wsdl:port>

  <wsdl:port name="HelloWorldPort_Java"
        binding="tns:HelloWorldBinding">
    <soap:address
          location="http://localhost/soap/servlet/rpcrouter" />
  </wsdl:port>

</wsdl:service>
```

Multiple instances of the same server

Using a Web Service



```
pcitapi13:pfeiffer >
pcitapi13:pfeiffer > python2.2
Python 2.2.2 (#1, Jan 30 2003, 21:26:22)
[GCC 2.96 20000731 (Red Hat Linux 7.3 2.96-112)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
...
.pythonrc executed
>>>
>>> import WebService
>>> AirportWeather = WebService.ServiceProxy("http://live.capescience.com/wsdl/GlobalWeather.wsdl")
>>>
>>>
>>> for key in AirportWeather.methods.keys() :
...   print key
...
searchByCountry
searchByRegion
isValidCode
getStation
listCountries
searchByCode
searchByName
getWeatherReport
>>> nodes=AirportWeather.getWeatherReport("GVA")
>>> len(nodes)
53
>>>
```

Start Python

Create a proxy and connect to service

List the methods available from this service

Get the weather for Geneva airport (GVA)

Web services in HEP

- Distributed analysis (reconstruction)
  - E.g. Clarens
    - CMS distributed data server for remote analysis
    - Python with XML-RPC (and SOAP)
    - Interfacing to Grid services
    - **http://clarens.sourceforge.net/**
  - Similar activities at SLAC
    - Using Java and Agents
- Just starting ...

Summary

- Web/network interface to application
  - Independent of language of implementation
  - “The Internet is the platform”
- Using XML for information exchange
  - Methods and data
- SOAP needs a rather complex “infrastructure”
  - WDSL, UDDI
- XML-RPC is more simple, less heavy
  - But follows development of SOAP

Links

- WWW consortium
  - **http://www.w3.org/**
- XML-RPC
  - **http://www.xmlrpc.org/**
- SOAP
  - **http://www.w3.org/TR/2003/REC-soap12-part0-20030624/**

Optional slides

UDDI

- WSDL provides all the info on how to interact with a service to the consumer
- How to find what services are there ?

- ➔ **U**niversal **D**escription, **D**iscovery and **I**ntegration project
  - Two parts
    - A registry of all metadata of a web service
    - A set of WSDL port type definitions for manipulating and searching that registry

UDDI Registry

- **`<businessEntity>`**
  - representing the provider of a web service
    - Information on the company
      - Contact information, ...
    - List of services provided
- **`<businessService>`**
  - represents a specific web service provided by that businessEntity
    - How to bind to the service
    - What type of service it is
    - Uses binding templates (for each implementation)

UDDI Features

- Global network of linked registries
  - Alternatively private ones
    - For communication between selected companies or industry group
- UDDI Interfaces
  - Publisher IF
  - Inquiry IF
- Toolkits for using the UDDI IFs
  - Registration programs
  - Tools to locate services
  - Generating UDDI from WSDL