Web Services

part 1

Agenda

Part 1

- What Are Web Services?
- Services are everywhere. Why?
- Web-services and SOA
- History of Web-services
- XML-RPC

What Are Web Services?

"Web service is a software system designed to support interoperable machine-to-machine interaction over a network"

By W3C

"Web Services are technology that allows *applications* to communicate with each other, regardless of the environment via *protocols* and *web interfaces*."

By Wiki

Web site vs Web service

Web Site

"Human-oriented".
Graphical user interface (GUI).

?

Web Service

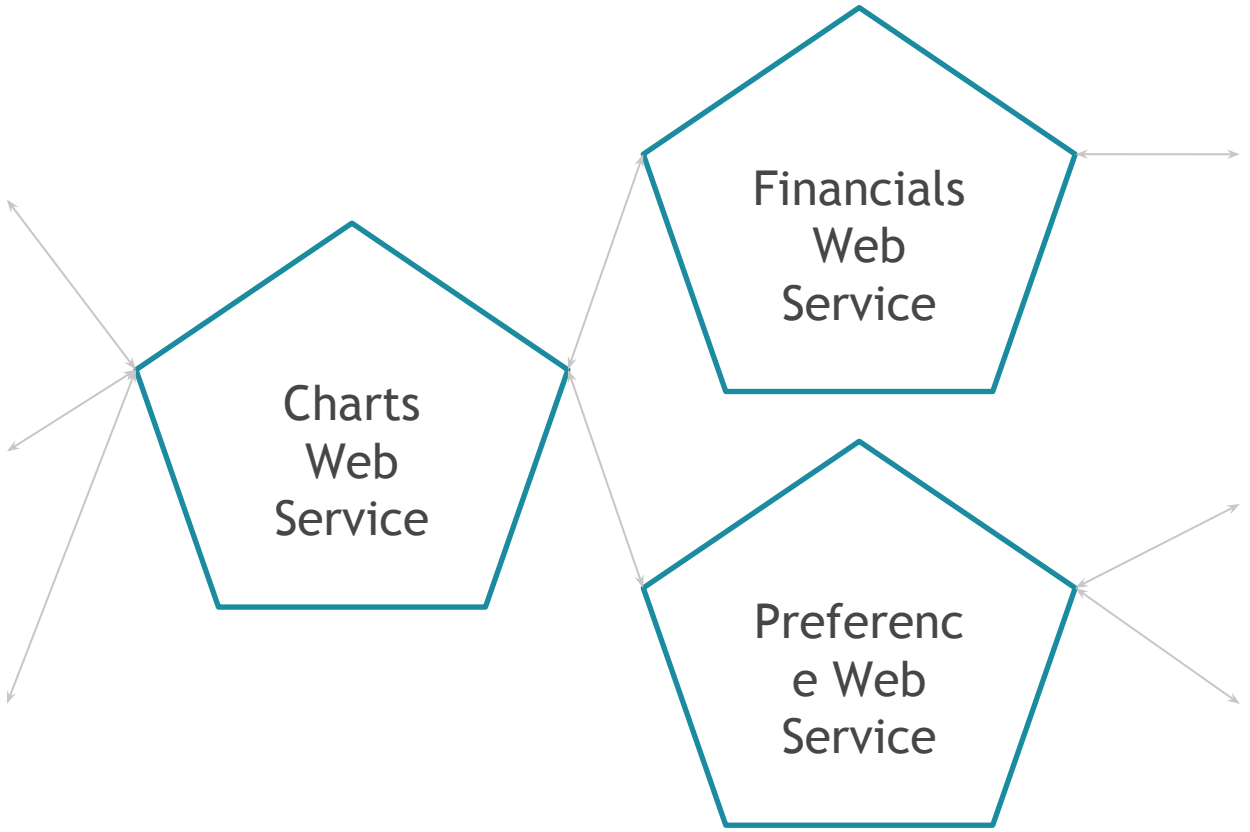
Software-oriented.
Thus, no GUI / visuals.

Web Service View

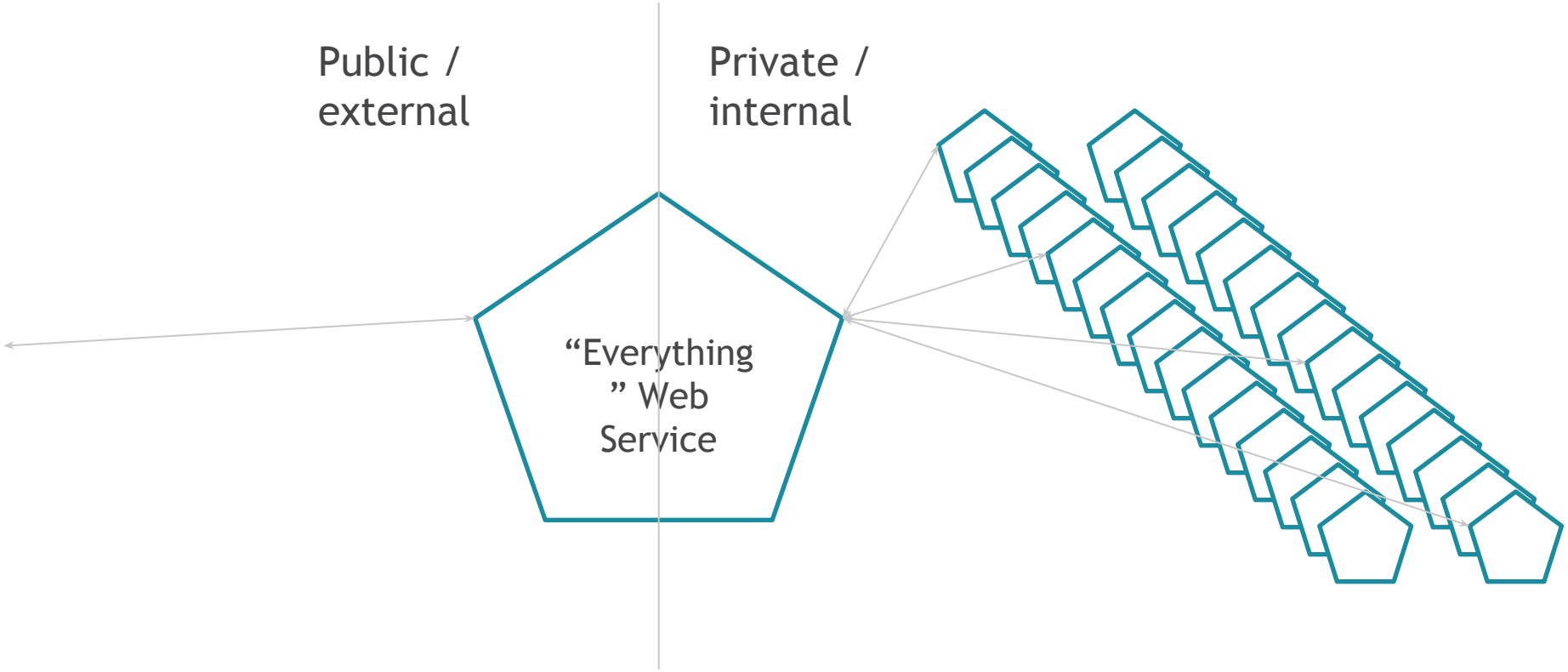
Get chart 300 x 200 px

Get chart 190 x 400 px

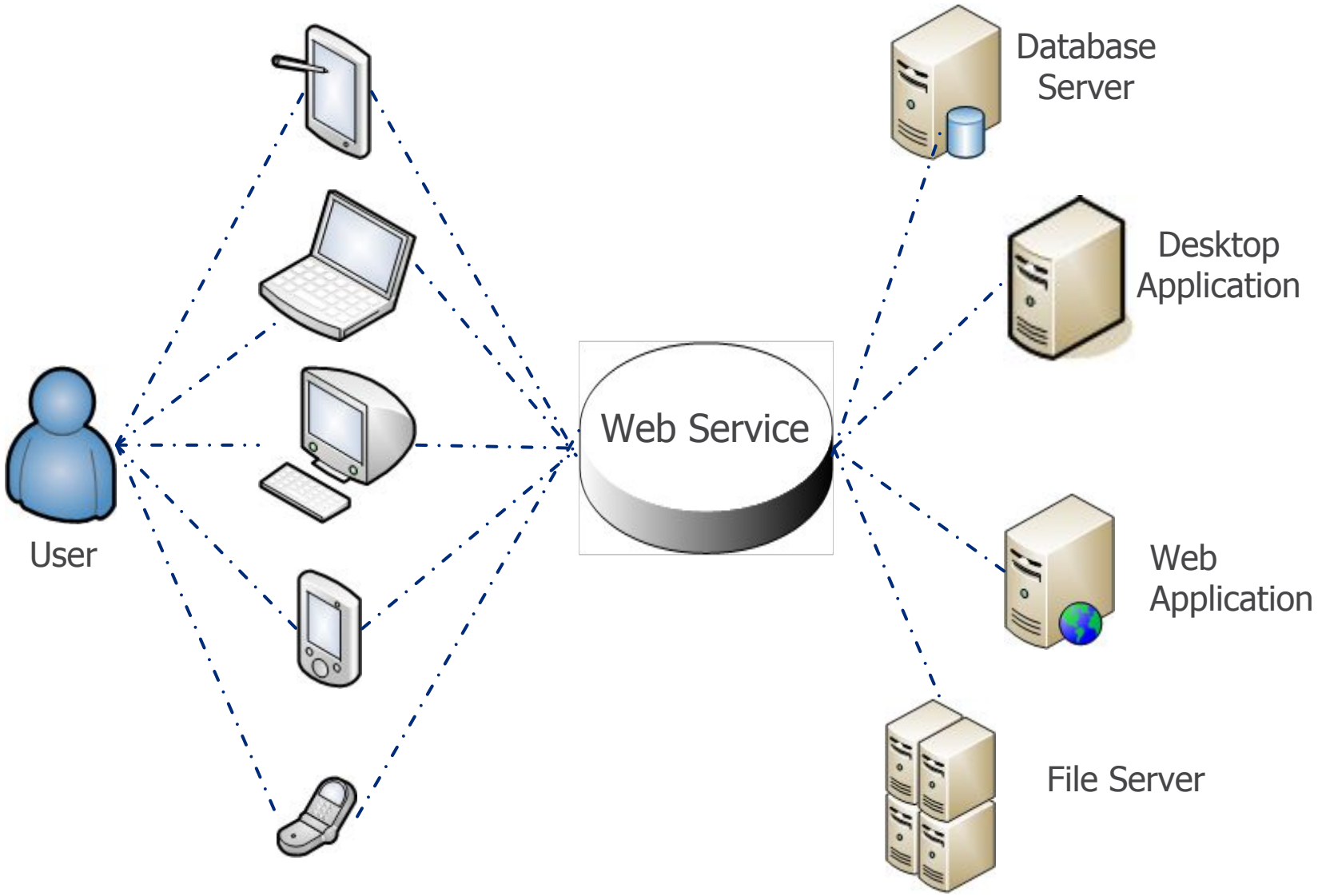
Get chart 250 x 400 px



Data aggregation services



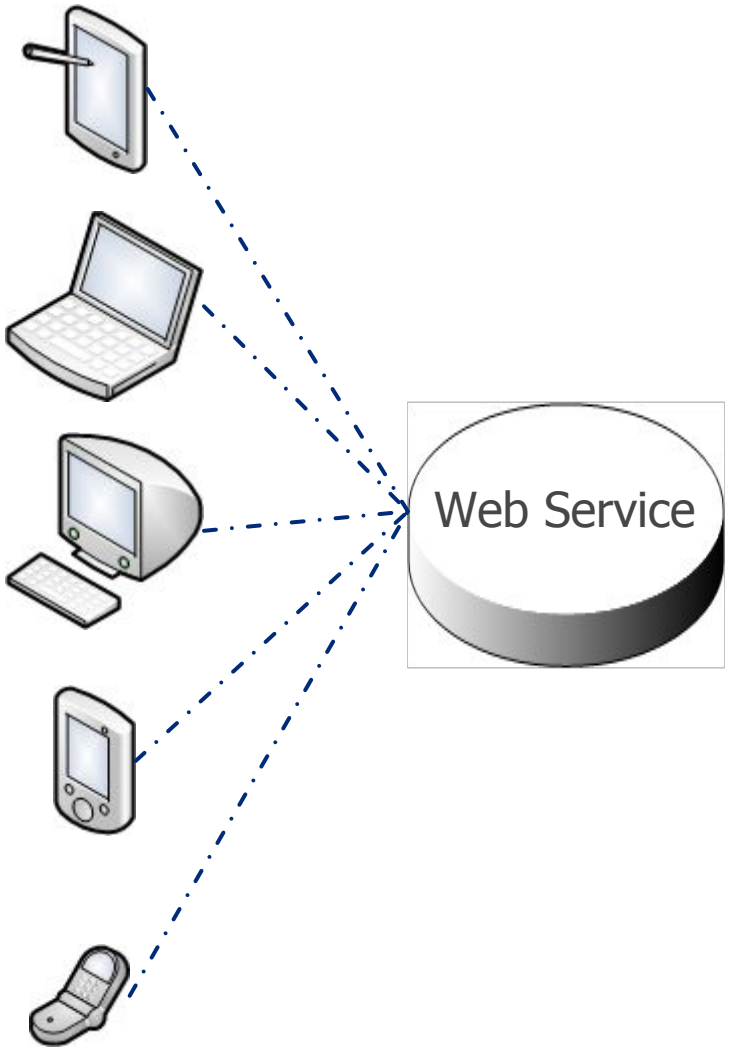
Services are everywhere. Why?



Reason #1: Common API

Web services are platform-independent.

Different (often incompatible) platforms can talk to each other via web service.



Reason #2: High compatibility

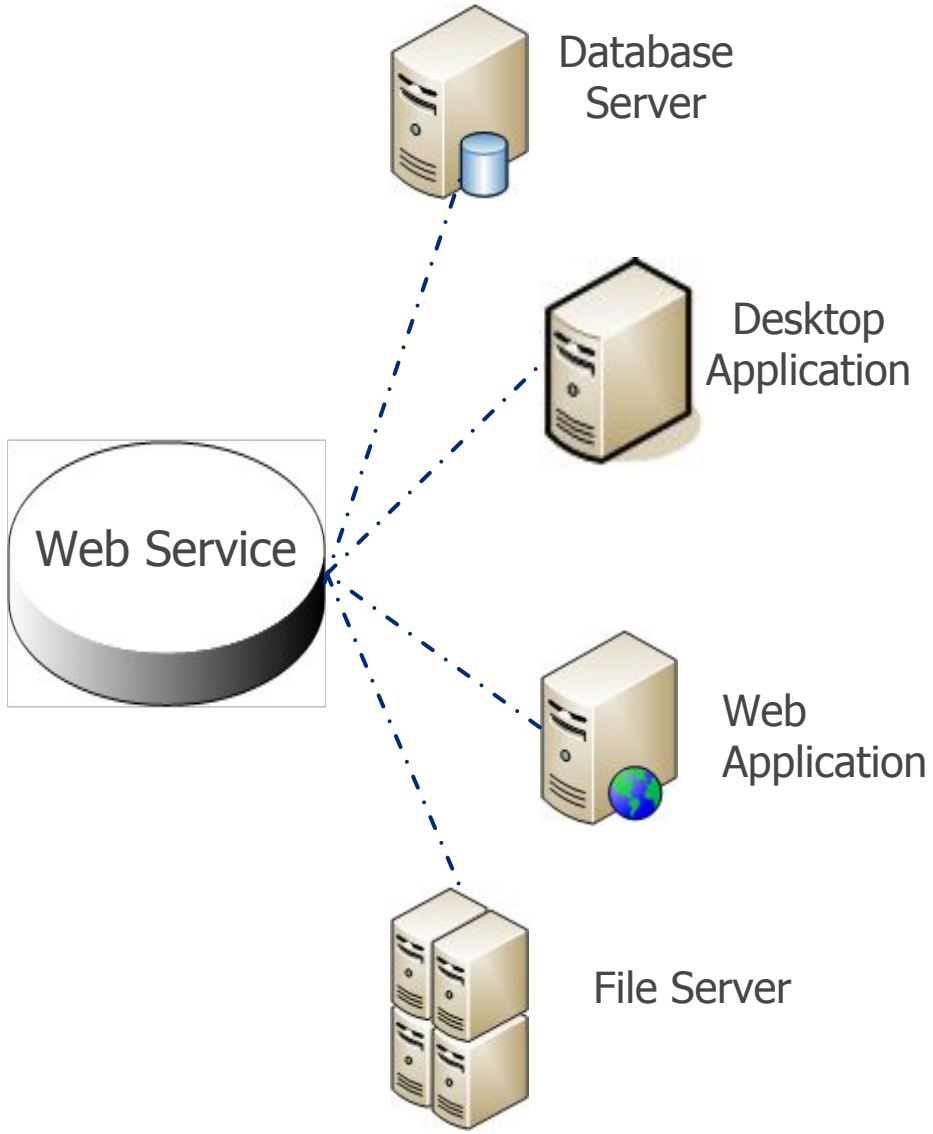
Web services often use simple trusted technologies – XML and HTTP.

HTTP (port 80) is often open even in high-security systems and firewalls.

Reason #3: Black box

Internal implementation is hidden from clients.

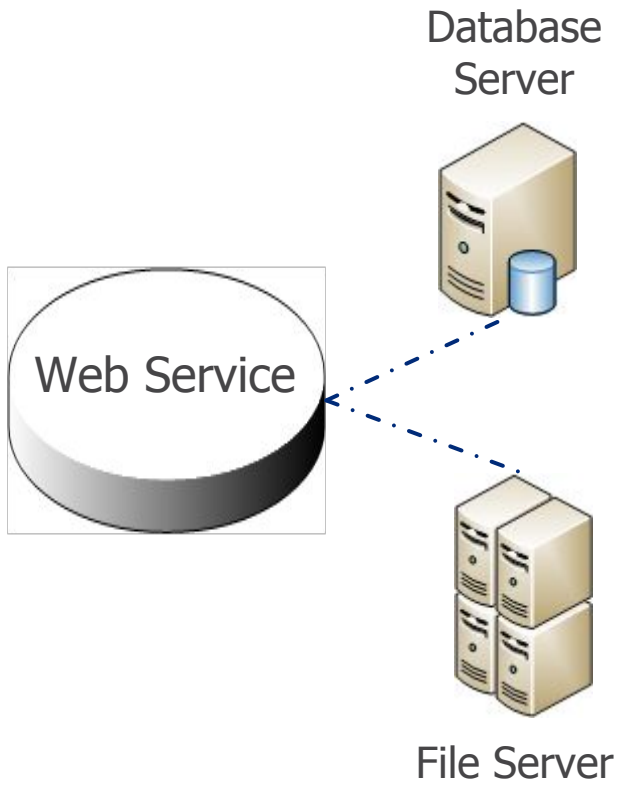
Internal system(s) may be developed, tested, upgraded and deployed separately.



Reason #4: Security

Web service API defines allowed manipulations.

This provides limited access to internal systems.



Benefits

- Open infrastructure
- Platform and language transparency
- Modular design

Integrating network-accessible services, which are interoperable because each has an interface that clearly defines the operations encapsulated in the service.

System, services as building block components may be characterized as *unassociated* and *loosely coupled*.

History of Web-services

early 1990s

DCE/RPC

```
/* echo.idl */
[uuid(2d6ead46-05e3-11ca-7dd1-426909beabcd), version(1.0)]
interface echo {
      const long int ECHO_SIZE = 512;
      void echo(
            [in]              handle_t h,
            [in,  string]     idl_char from_client[ ],
            [out, string]     idl_char from_server[ECHO_SIZE]
      );
}
```

History of Web-services

early 1990s — DCE/RPC

soon — MSRPC

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM
1998	XML-RPC

Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
  <methodName>stock.getPrice</methodName>
   <params>
     <param>
       <value><string>IBM</string></value>
     </param>
   </params>
</methodCall>
```

Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
   <params>
     <param>
       <value><double>34.5</double></value>
     </param>
   </params>
</methodResponse>
```

History of Web-services > XML-RPC > Datatypes

- Integer: <i4> or <int>
- Boolean: <boolean> with value of 0/1 or true/false
- String: <string>
- Double: <double>
- Date/time: <dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>
- Base64: <base64>
- Struct:

```
<struct>
  <member>
    <name>something</name>
    <value><i4>1</i4></value>
  </member>
</struct>
```

- Array:

```
<array>
  <data>
    <value><i4>1404</i4></value>
    <value><string>Some string</string></value>
  </data>
 </array>
```

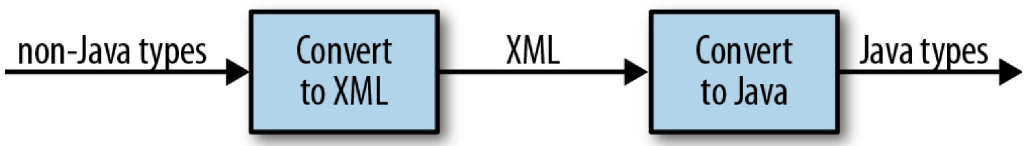
- Nil: <nil/>

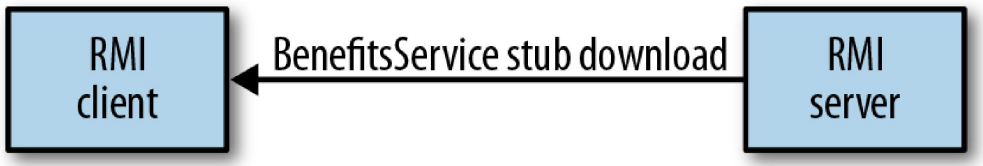
XML-RPC

- Text
- HTTP (later SMTP)

DCE/RPC

- Binary
- Any other





Benefits and drawbacks

Benefits

- Robust standard with long history and solid support from nearly all programming languages
- XML-RPC doesn’t need in general any additional “contract” like WSDL, WADL etc.
- XML-RPC supports all basic datatypes “out of box”

Drawbacks

- XML is too verbose (in comparison with JSON)
- Usage of custom datatypes is complicated

XML-RPC Demo

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM
1998	XML-RPC
Sep 13, 1999	SOAP 1.0

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM
1998	XML-RPC
Sep 13, 1999	SOAP 1.0
2000	REST

REST Request/Response Example

Request

```
POST /stock
Host: www.stocks.com
Authorization: Basic xxxxxxxxxxxxxxxxxx
Accept: application/json
Content-Length: nnn
Content-Type: application/json

{
  “name”: “IBM”,
  “price”: “34.5”
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: nnn

{
  “name”: “IBM”,
  “price”: “34.5”
}
```

Benefits and drawbacks

Benefits

- Can use any encoding (XML, JSON, etc.)
- Easy and relatively fast implementation
- Doesn’t require contract like WSDL
- Reuses HTTP protocol features instead of inventing new

Drawbacks

- Not a standard – limited support from programming languages
- As it doesn’t have schemas and formal contract might be bad documented

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM
1998	XML-RPC
Sep 13, 1999	SOAP 1.0
2000	REST
May 8, 2000	SOAP 1.1

History of Web-services

early 1990s	DCE/RPC
soon	MSRPC
October 1991	CORBA
1993	COM/OLE
1993	MSRPC + COM/OLE = DCOM
1998	XML-RPC
Sep 13, 1999	SOAP 1.0
2000	REST
May 8, 2000	SOAP 1.1
June 24, 2003	SOAP 1.2

SOAP Request/Response Example

Request

```
POST /stock HTTP/1.1
Host: www.stocks.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope
"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>

</soap:Envelope>
```

Benefits and drawbacks

Benefits

- Robust standard with long history and solid support from nearly all programming languages
- WSDL allows automatic method generation and serves as documentation even if there’s no documentation
- Allows much more functionality in comparison with XML-RPC
- Supports all basic datatypes “out of box”
- Allows usage of custom data (XSD-defined)

Drawbacks

- XML is too verbose (in comparison with JSON)
- Relatively time-consuming development (requires WSDL, XSD)
- Standard is too universal – most services don’t use all features

Web Services Types

By invocation target type:

- Method call (XML-RPC)
- Message transfer (SOAP)
- Resource manipulation (REST)

Each Web Service relies on at least two protocols:

- Data encoding protocol
  Determines how data is represented.
- Transport protocol
  Determines how data is transmitted.