

Web Services – Setting the Stage

A simple, live, interactive demo illustrating Web Services as XML document exchange between a client and a server


Version 14

A simple live interactive scenario

- Stock quote example
 - Supply a symbol
 - Get a price quote back

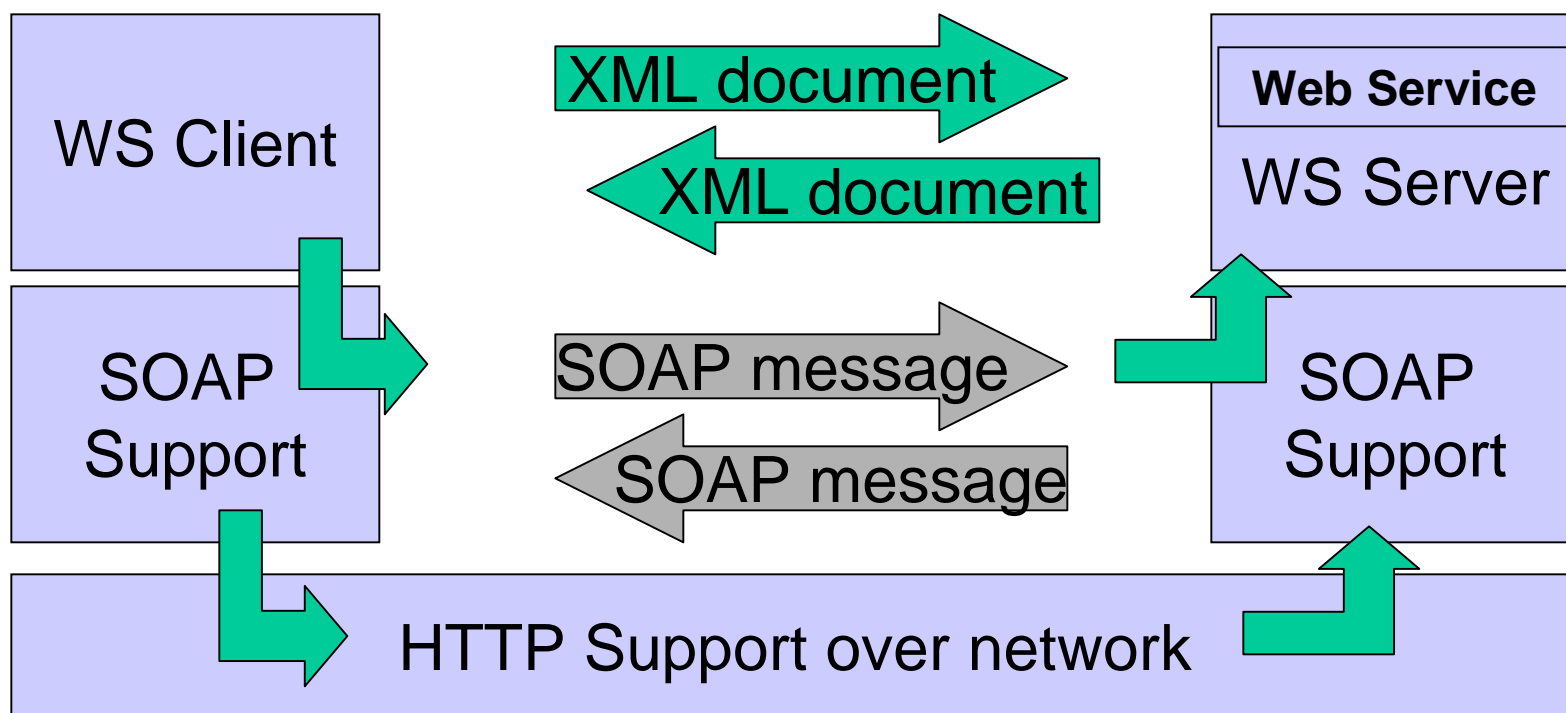
- How ?
 - Build up the Web Service XML request document
 - Invoke the service, supplying the document
 - Review the XML response document

Let's go – using a simple environment

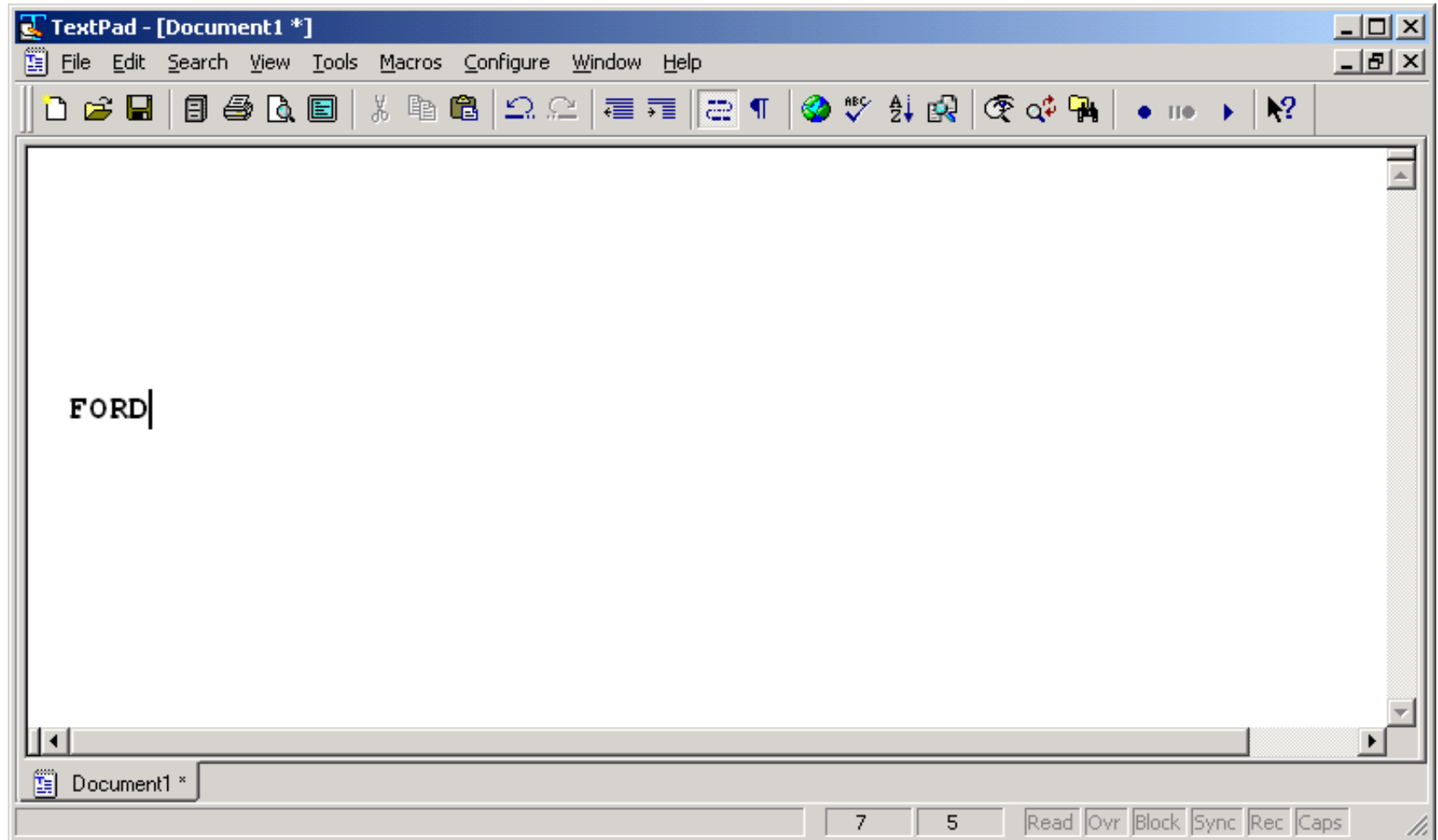
- We could use any number of tools for this demo but to show the simplicity of WS, we will use:
 - A simple text editor (Text Pad) to construct the XML document
 - A small part of the popular XMLSPY tool: to invoke the service and review the results
 - A remote, independent Web Service server to return the quotes
-  The **SOAP protocol** will carry the message to the service, and the reply back to the client

How will all of this work?

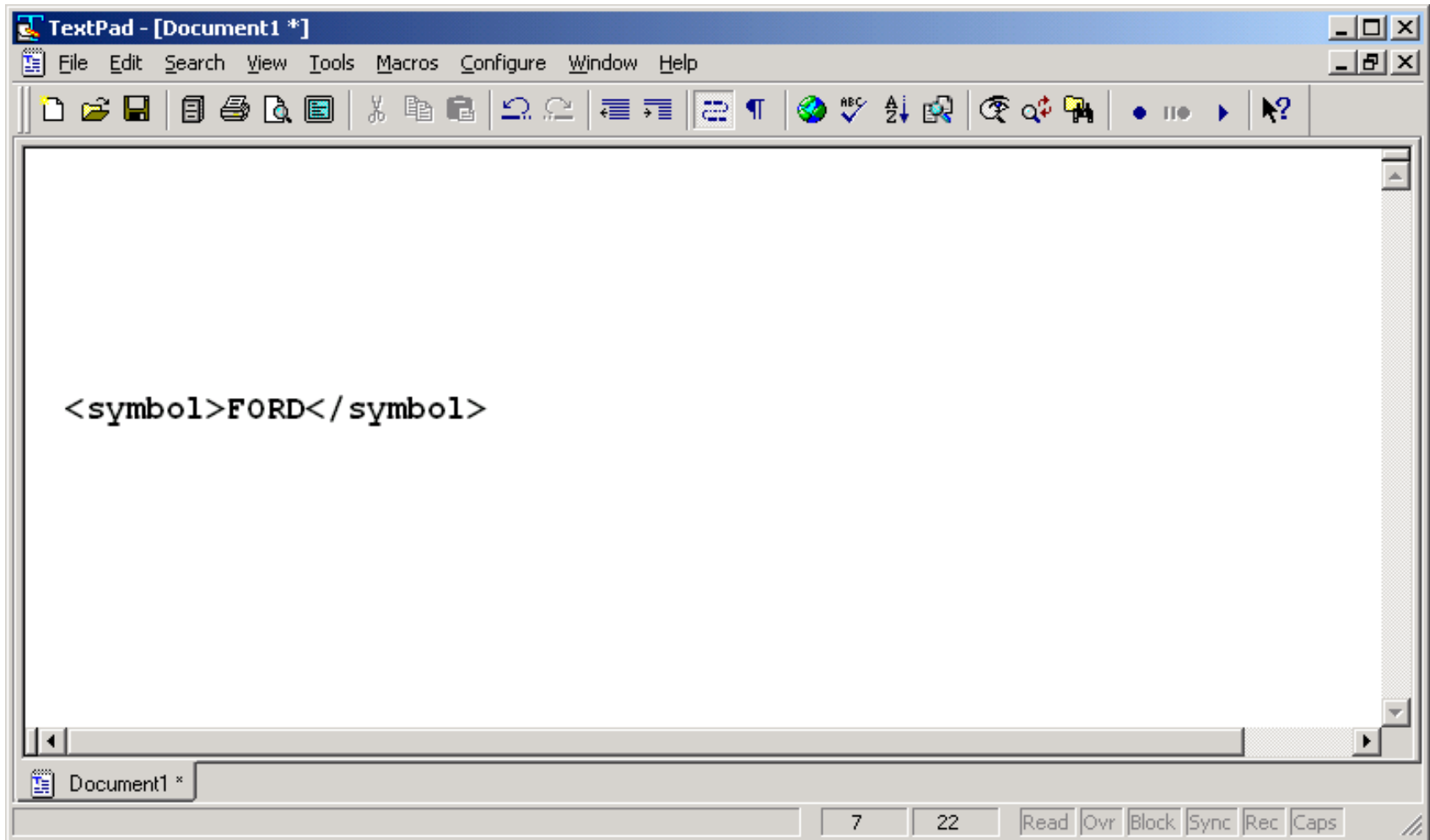
A Web Services Infrastructure



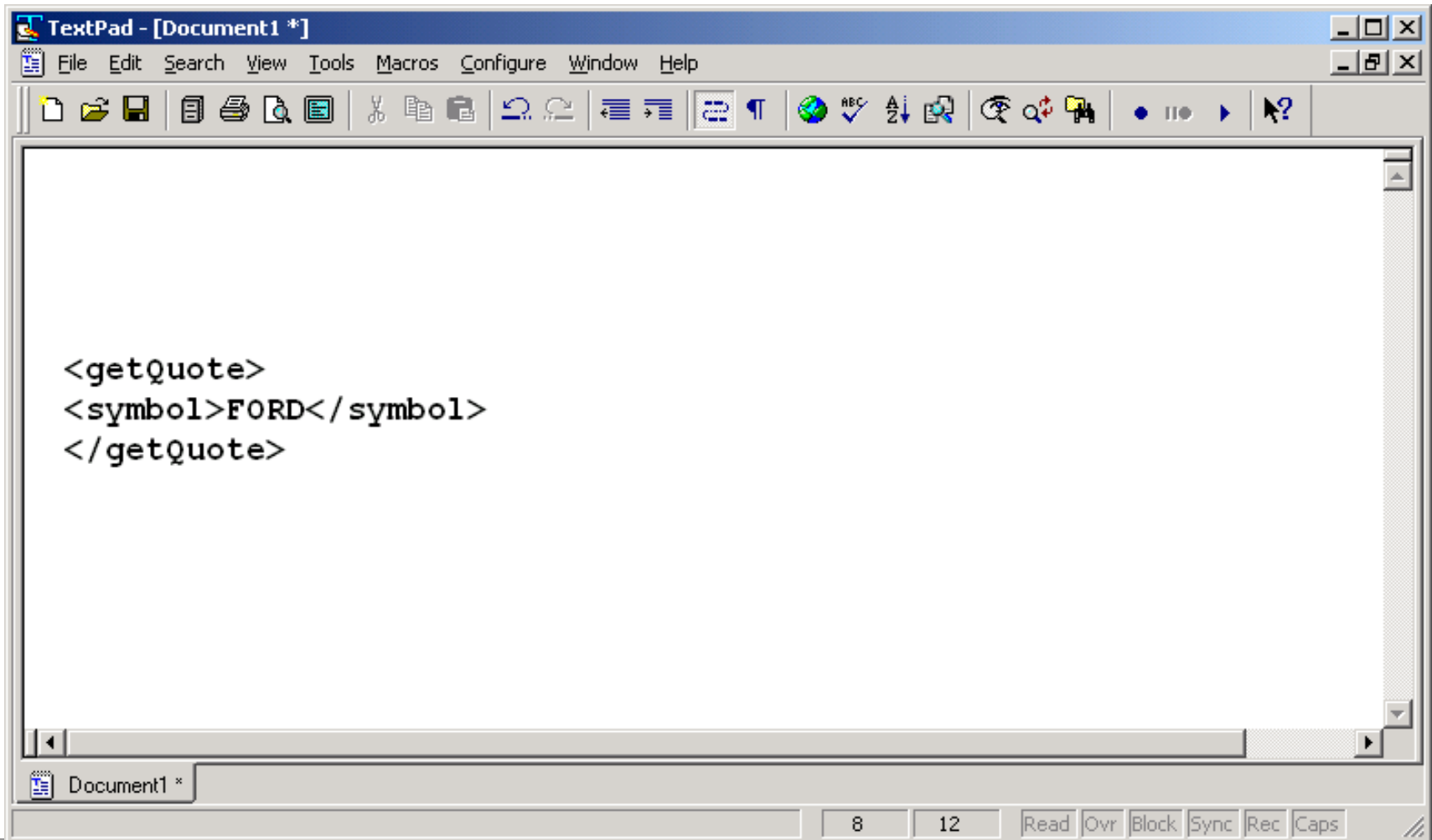
The simplest of data: a stock symbol!



Self-identify as a stock symbol



Extend our XML document to indicate the type of message (request)

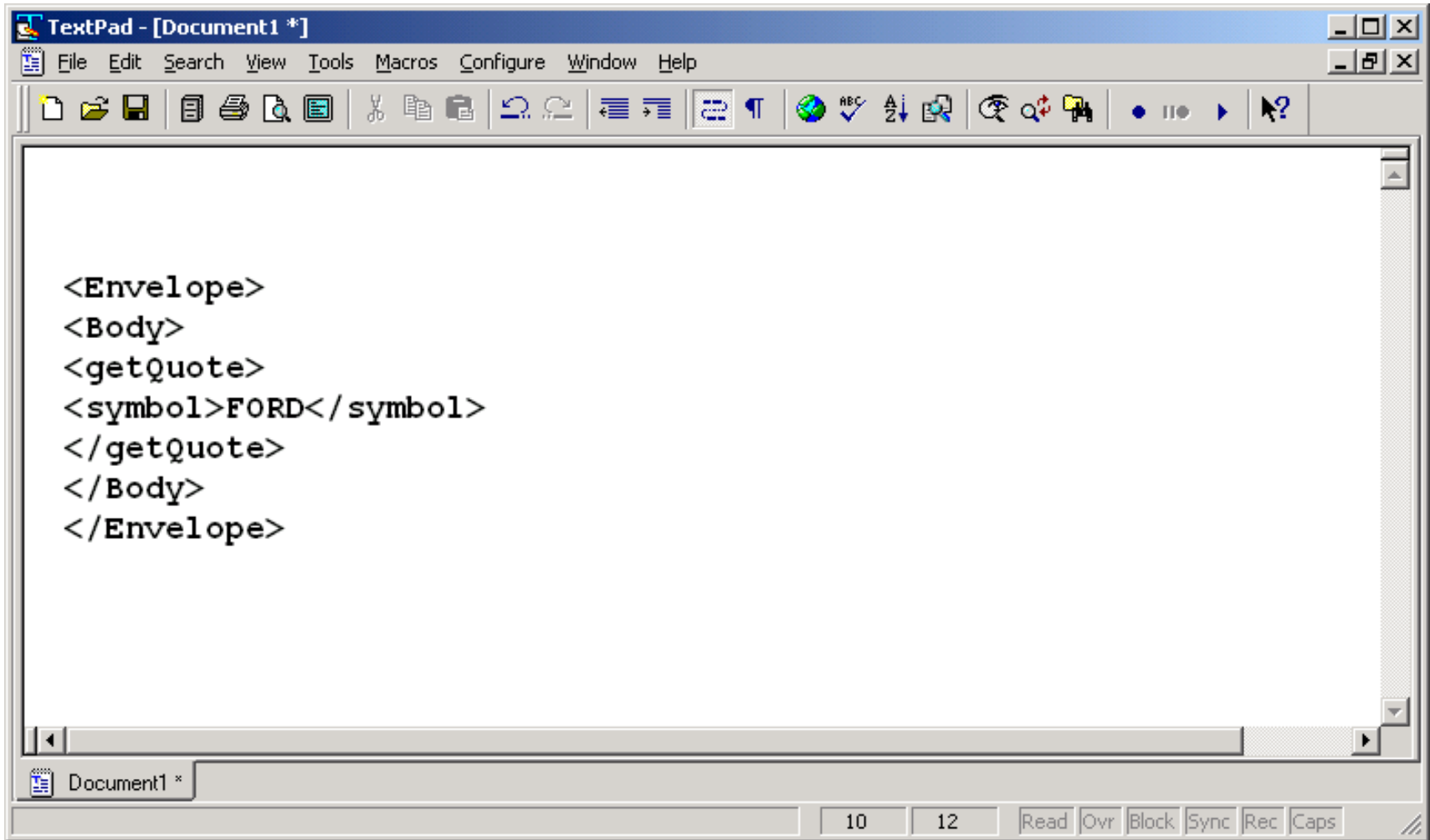


The screenshot shows a TextPad window titled "TextPad - [Document1 *]". The menu bar includes File, Edit, Search, View, Tools, Macros, Configure, Window, and Help. The toolbar contains various icons for file operations, editing, and navigation. The main text area contains the following XML code:

```
<getQuote>  
<symbol>FORD</symbol>  
</getQuote>
```

The status bar at the bottom shows "Document1 *" and a set of keyboard shortcuts: 8, 12, Read, Ovr, Block, Sync, Rec, Caps.

Wrap the document within a SOAP message (envelope and body)

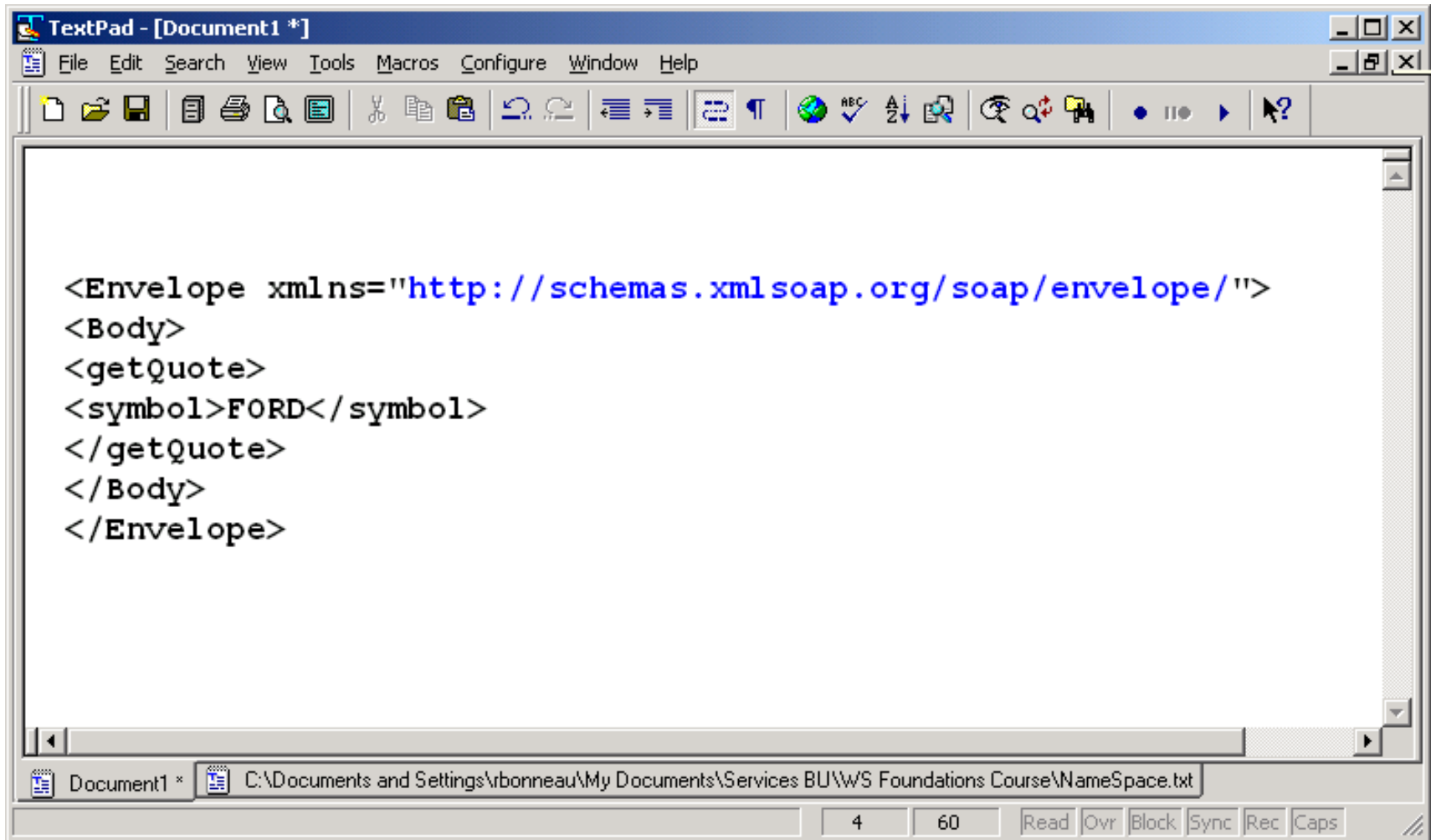


The image shows a screenshot of a TextPad window titled "TextPad - [Document1 *]". The window contains the following XML code for a SOAP message:

```
<Envelope>  
<Body>  
<getQuote>  
<symbol>FORD</symbol>  
</getQuote>  
</Body>  
</Envelope>
```

The window also shows a menu bar with "File", "Edit", "Search", "View", "Tools", "Macros", "Configure", "Window", and "Help". A toolbar with various icons is visible below the menu bar. The status bar at the bottom of the window displays "Document1 *", "10", "12", and several status indicators: "Read", "Ovr", "Block", "Sync", "Rec", and "Caps".

One “technical detail” re namespaces



The screenshot shows a TextPad window titled "TextPad - [Document1 *]". The menu bar includes File, Edit, Search, View, Tools, Macros, Configure, Window, and Help. The toolbar contains various icons for file operations, editing, and navigation. The main text area contains the following XML code:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
<getQuote>
<symbol>FORD</symbol>
</getQuote>
</Body>
</Envelope>
```

The status bar at the bottom shows the file path: C:\Documents and Settings\rbonneau\My Documents\Services BU\WS Foundations Course\Namespace.txt. The status bar also displays the page number 4, the total number of pages 60, and several status indicators: Read, Ovr, Block, Sync, Rec, and Caps.

Transmit to a waiting Web Service Server

We'll use XMLSPY to transmit the SOAP message

The screenshot shows the XMLSpy v5.4.0 interface. The main window displays a SOAP message in XML format:

```
<Envelope xmlns="http://schemas.xmlsoap.org
<Body>
<getQuote>
<symbol>FORD</symbol>
</getQuote>
</Body>
</Envelope>
```

A context menu is open over the message, with the following options:

- Create new SOAP request
- Send request to server
- Change SOAP request parameters
- Soap Debugger Session
 - Go
 - Single Step
 - Break on next Request
 - Break on next Response
 - Stop the proxy server
- Soap Debugger Options

The status bar at the bottom indicates the current file is 'Untitled1.xml' and the cursor is at 'Ln 7, Col 12'. The XMLSpy logo is visible in the bottom right corner.

Response message from server

The screenshot shows the XMLSpy v5 interface. The main window displays the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <n:getQuoteResponse xmlns:n="urn:xmethods-delayed-quotes">
      <Result xsi:type="xsd:float">7.25</Result>
    </n:getQuoteResponse>
  </soap:Body>
```

The value **7.25** is circled in black, and an arrow points from the circle to the right.

XMLSpy v5 rel. 4 U Registered to Richard Bonneau (IONA) ©1998-2003 Altova GmbH & Altova, Inc. Ln 1, Col 1

Change the symbol to “IBM” and re-send. Here’s the response:

The screenshot shows the XMLSpy interface with the following XML content displayed in the main editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <n:getQuoteResponse xmlns:n="urn:xmethods-delayed-quotes">
      <Result xsi:type="xsd:float">98.99</Result>
    </n:getQuoteResponse>
  </soap:Body>
</soap:Envelope>
```

The interface includes a menu bar (File, Edit, Project, XML, DTD/Schema, Schema design, XSL, Authentic, Convert, View, Browser, WSDL, SOAP, Tools, Window, Help), a toolbar, and a Project Explorer on the left showing a tree structure of examples like Org-Chart, Datasheet, Expense Report, etc., with 'Web Services Foundation' expanded to show 'StockQuoteComplexResponse'.

At the bottom, the status bar indicates: Making XMLSPY v5 rel. 4 U Registered to Richard Bonneau (IONA) ©1998-2003 Altova GmbH & Altova, Inc. Ln 1, Col 1

Here's a more complex document returned from a *different* stock quote service

The screenshot shows the XMLSpy v5 interface. The main window displays an XML document titled 'StockQuoteComplexResponse'. The document structure is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetQuoteResponse
      xmlns="http://www.xignite.com/services/">
      <GetQuoteResult>
        <Outcome>Success</Outcome>
        <Identity>IP</Identity>
        <Delay>0.703125</Delay>
        <Name>IONA TECH </Name>
        <Exchange>NasdaqNM</Exchange>
        <Quote>
          <Symbol>IONA</Symbol>
          <Previous_Close>7.25</Previous_Close>
          <Open>N/A</Open>
          <High>7.35</High>
```

The left sidebar shows a project tree with 'Examples' expanded, and 'Web Services' containing several 'StockQuo' entries. The status bar at the bottom indicates 'Ln 3, Col 13'.

How did we know what structure of XML to send; and what we'd get back in response?

- The **WSDL (Web Services Description Language)** file specified the Stock Quote service:
 - Operations (names and parameters)
 - Message formats
 - Endpoint URL
- Let's look quickly at the WSDL file for the original simple Stock Quote service

Screen shot of WSDL file

The screenshot shows the XMLSpy application window titled "XMLSPY - [http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl]". The project tree on the left shows a folder named "Web Services Foundation" containing several files, including "http://services.xmethc" and "http://www.xignite.co". The main editor displays the following XML code:

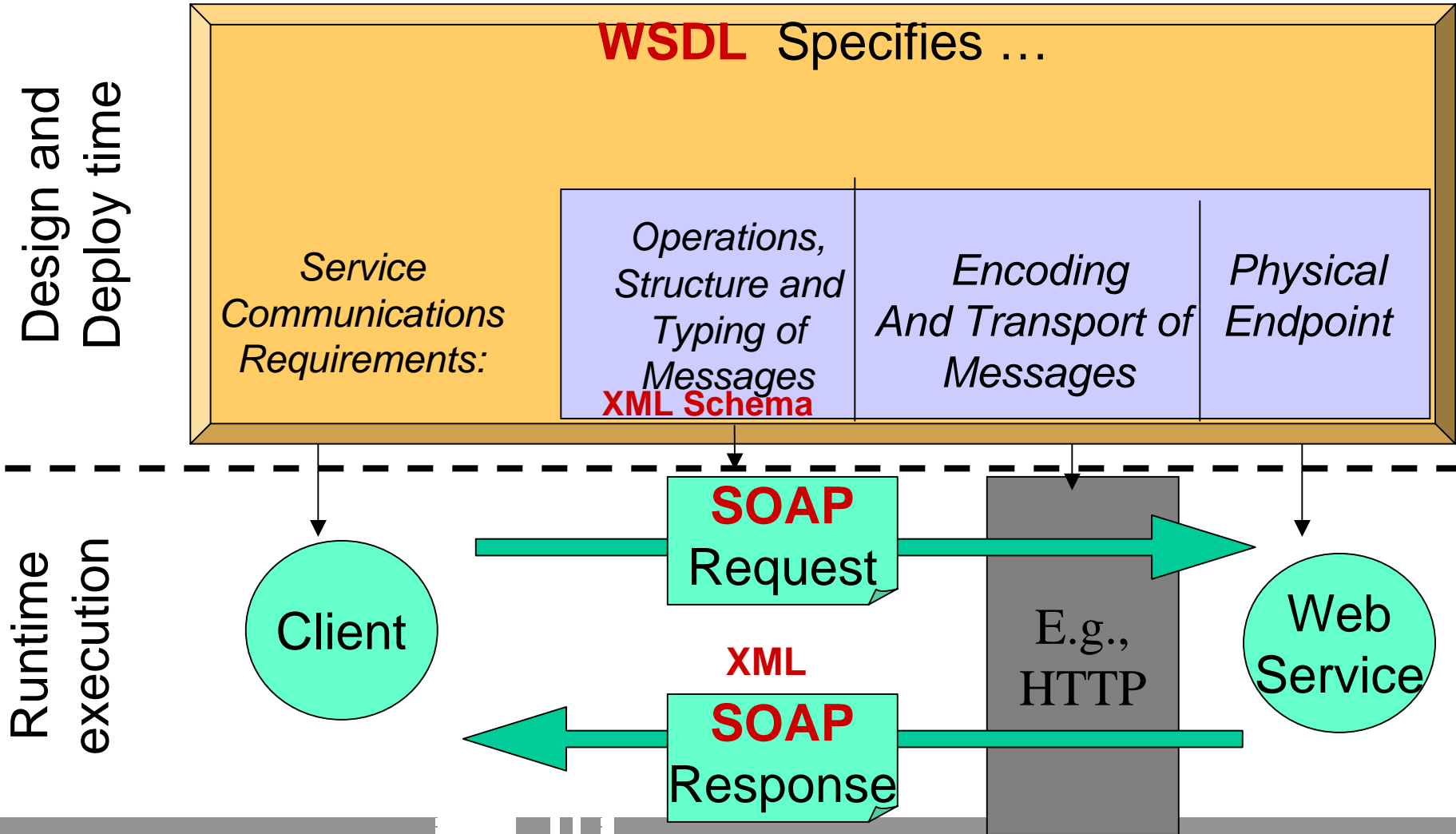
```

ds.services.stockquote.StockQuote/"
xmlns:electric="http://www.theminelectric.com/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encodin
g/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.theminelectric.com/wsdl/net
.xmethods.services.stockquote.StockQuote/"
name="net.xmethods.services.stockquote.StockQuote">
  <message name="getQuoteResponse1">
    <part name="Result" type="xsd:float"/>
  </message>
  <message name="getQuoteRequest1">
    <part name="symbol" type="xsd:string"/>
  </message>
  <portType

```

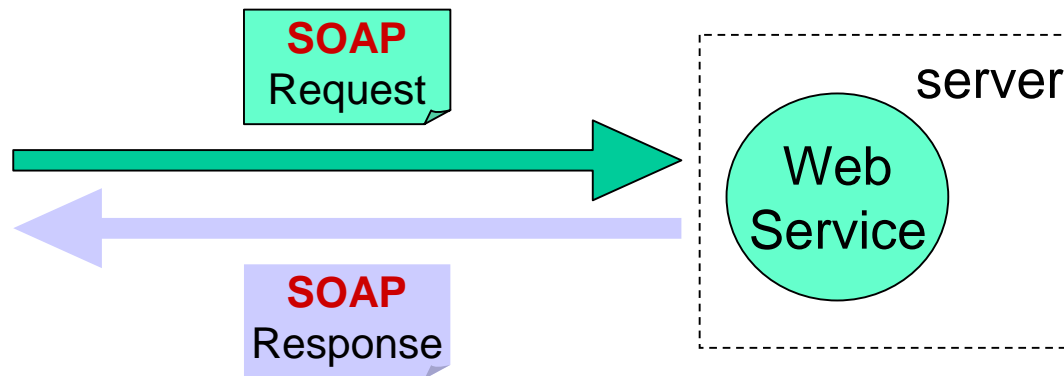
Web Service Technologies Relationships

Run time vs. Design Time



What is the Web Service Server Doing?

- The typical server pattern:
 - (passively) **wait** on arriving XML documents contained within a SOAP envelope
 - Extract and analyze the document
 - Interact with other software components to complete the processing of the document
 - Optionally: return another XML document in response



Points to be made about demo

- This is not the same as Web pages or Web applications
 - No browsers, web servers, HTML, CGI, ASP pages, GUIs, etc
- Rather: exchange of XML documents between “co-operating” programs
 - Web Services is intended for program-to-program communications
 - **Not** human-to-web page communications

Points to be made about demo (cont.)

- Uses newer information exchange standards
 - XML technologies for documents
 - ASCII, human-readable, self-describing, extendable, simple, and portable across OS, hardware, programming languages
 - SOAP as vehicle for transmitting/receiving XML documents
 - WSDL to specify definition/structure/rules of the service
 - Messages/documents
 - Location of service for access
 - Transport options, etc.
- More enterprise-level WS standards emerging for such aspects as transactions, security, choreography of services, etc.

Points to be made about demo (cont.)

- We accessed the service using Web Services
 - **Web Services is an “interfacing technology”**
 - **It is not a technology for implementing servers (or clients)**

Final points...

- Can easily read, edit, re-transmit and interpret the XML documents and SOAP messages - because it's all text
- There are simple tools to help you to make WS calls from inside your programs, and to accept such calls into your server programs
 - Create and send SOAP messages from the client programs
 - Accept and handle them in server programs
- Remember that some XML documents can be very complex (as we've seen)
 - So these tools can save you a lot of effort
- We'll see some of these tools in later chapters

Transition to the rest of the course

- We will explore in greater detail the list of technologies touched upon in this short demo:
 - XML technologies – how to represent documents
 - Including XML Schema (not introduced yet)
 - SOAP – how to wrap documents to be sent out
 - WSDL – how to describe services and document exchanges
 - UDDI (not discussed yet) – how to register, find services
- Also, we will present alternative solutions and approaches beyond what we have seen
 - Other transports, protocols, options, etc.
 - Best practices for designing WS-based systems