

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Part V: CREATING SERVICES

Chapter 14, **Web Services Basics**

Chapter 15, **Creating Enterprise Services in ABAP**

Chapter 16, **Creating and Consuming Services in Java**

Chapter FOURTEEN. Web Services Basics

Enterprise services are web services. For this reason, to comprehend what ESA can do for you, you have to understand the basic principles of web services.

14.1. What are web services and why do we care?

To understand the benefits of web services and even what they are, it helps to think back a bit to life before the advent of the Web. It was very difficult for diverse kinds of computers to communicate; they each had their own protocol stacks and translation was tricky. The Web offered a lowest-common-denominator approach, making text its foundation with HTML and sending it over one standard protocol, HTTP. This created a revolution. Suddenly web browsers running on any platform could communicate with web servers on any platform, and they didn't need to know or care where those resources resided.

In essence, web services create the same type of revolution, but this time for programs. Integrating disparate applications is the bane of the IT professional. Integration work requires using specialized application programming interfaces (APIs) or complex standards such as DCOM and CORBA to communicate between applications. Further, when applications are upgraded, the integrations typically break and must be refactored.

Web services strip this process down from programmatic remote procedure calls to simple text messages, which are exchanged between systems. With the help of web services, integrators do not need to know or care whether the application they will be interoperating with is written in Java, C++, C#, Perl, or ABAP. It simply doesn't matter, as we will see.

Here's a formal definition of web services:

Web services are self-contained and self-describing application functionalities that can be processed through open Internet standards.

Web services are, in essence, small, modular applications that communicate using a text format and open Internet standards.

Their modularity is important because we can create and expose services slowly and organically with a gradual adoption plan. We can expose existing application functionality as services, allowing reuse instead of reinventing the wheel. Their small scope allows us to respond quickly to changing business conditions, introducing new services as needed. We will unpack the more formal definition further as we go, but this is enough to get you started.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.2. What are some examples of web services?

Web services, as the term says, provide a service. It might be a price check, an inventory status check, a stock quote, or an airline reservation. Examples of web services include:

- Intelligent product catalog searches
- Product availability checks
- Pricing inquiries
- Customer credit checks
- Order status checks
- Vendor-managed inventories
- Demand forecasts, stock replenishment
- Dynamic auctioning and bidding
- Publishing and analyzing financial reports
- Electronic bill presentment and payment
- Matching vacancies and job applicants' profiles
- Postal service address checks
- Universal Description, Discovery, and Integration (UDDI) registration and discovery
- Automated web searches (Google)

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.3. What are services?

Services are callable entities or application functionalities accessed via the exchange of messages. To facilitate this exchange, a web service has a clearly defined contract or interface. This interface defines:

- What actions are performed
- The structure of request, response, and fault messages
- The URL of the service
- The technical means to access the URL

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.4. What is service-oriented architecture?

According to Eric Newcomer and Greg Lomow in their book *Understanding SOA with Web Services* (Independent Technology Guides):

A service-oriented architecture (SOA) is a style of design that guides all aspects of creating and using business services throughout their lifecycle (from conception to retirement), as well as defining and provisioning the IT infrastructure that allows different applications to exchange data and participate in business processes regardless of the operating systems or programming languages underlying those applications.

Simply put, SOA is a system architecture based on services. Although older styles of SOAs include those that use DCOM or CORBA, today when people speak of SOAs, they mean SOAs based on web services and the standards that go along with them.

User name: VLADIMIR BLAGOJEVIC

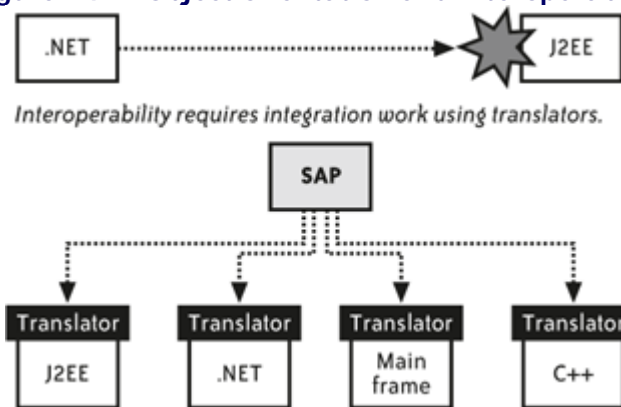
Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.5. Why is service orientation better than object orientation?

Object orientation is a powerful programming paradigm in which applications expose functionality through well-defined interfaces called methods. Details are hidden from the calling application. However, the calling application must be written in the same language as the object it is trying to access. If it's not, we need a translator program, as shown in Figure 14-1.

Figure 14-1. Object orientation and interoperability



Web services provide a standard way to communicate between services, which may be written in different languages, as shown in Figure 14-2.

Figure 14-2. Service orientation and interoperability



User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

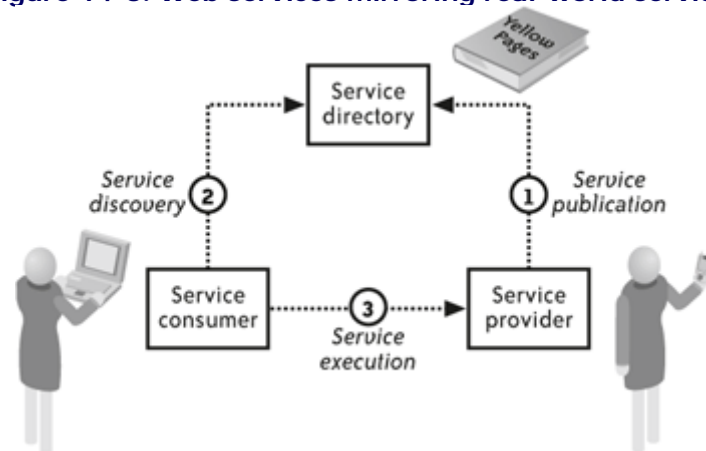
No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.6. What are the main components of web services?

The easiest way to understand the main components of web services, which are providers, directories, and consumers, is to think about a real-world example.

You can think of nearly everything a business does as a service. What do you do if you are looking for a service, such as auto repair? You look it up in a service directory, such as the Yellow Pages. You as the potential service consumer look in a directory for a service provider. Using the information in the directory (in this case, a phone number), you then contact the provider, who performs the service you requested, as shown in Figure 14-3.

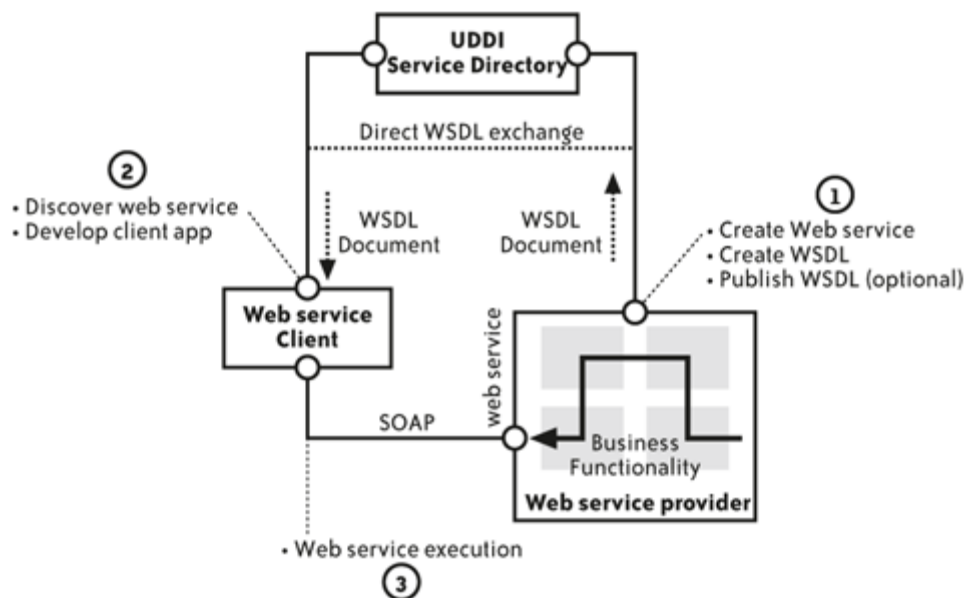
Figure 14-3. Web services mirroring real-world services



If you think of what happens step by step, first the service provider publishes business information in the directory, then you find or discover the service, you contact the provider, and the provider executes the service for you.

Web services mirror this structure. A provider of a web service writes a service or exposes functionality as a service (the service itself may be written in any language) and then writes a description of that service in the Web Services Description Language (WSDL). The WSDL might be passed directly to a business partner, or it could be published in a UDDI directory. The web services consumer uses the WSDL description to either construct or configure a client, depending on the nature of the service. Now it is time to execute the service. The client and the server must translate their programmatic calls into SOAP messages that handle the execution. Figure 14-4 shows this process.

Figure 14-4. Basic web services components



Although conceptually it is clear what each web services standard is doing, let's examine the main web services standards in more detail now, starting with the standard that underpins the web services standards: XML.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.7. What is XML?

XML is the Extensible Markup Language. It is the universal format for structured information on the Web.

XML is a markup language similar to HTML. Both are simple to write and use and both can be read by humans and by intelligent clients. Both are text formats.

To understand XML and how it differs from HTML, let's step back and consider what we know about HTML:

- HTML separates data from presentation.
- HTML describes how data should look when it is viewed in a browser.
- HTML uses a fixed tag set.
- HTML tags relate to formatting: how to make text look bold, how to display tables, how to define styles for displaying information, and so on.

XML, on the other hand, separates data from meaning. XML's characteristics include the following:

- XML has no fixed tag set.
- XML has no fixed semantics.
- Instead of tags to describe what data should look like, XML tags define the data itself.

Figure 14-5 shows an XML snippet.

Figure 14-5. An XML-based purchase order

```
<?xml version="1.0" ?>
<PurchaseOrder>
  <date>1999-05-20</date>
  <shipTo>
    <name>John Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <lineitems>
    <Item>
      <product>Lawnmower</product>
      <price>148.95</price>
    </Item>
    <Item>
      <product>Birdbath Deluxe</product>
      <price>249.95</price>
    </Item>
  </lineitems>
</PurchaseOrder>
```

The root element of a purchase order document may have the tag `<PurchaseOrder>`. The tag describes the data it contains. A `<price>` element holds the price of an item.

Because the tags define the items that they hold and because XML data is always structured, simple parsers can retrieve the data from an XML document.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.8. What is XML schema?

XML data is structured. We can define what a complete purchase order looks like and then compare the documents we receive to that document. This is known as *validation*. One way to specify a document's structure is with the XML Schema Definition (XSD) language. XSD is an XML vocabulary used to articulate rules for business data. [Example 14-1](#) shows the XML schema used in a web service that retrieves an address.

Example 14-1. An XML schema

```
<xsd:schema
  targetNamespace="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Request_Message" type="Request"/>
  <xsd:element name="Response_Message" type="Response"/>
  <xsd:complexType name="Request">
    <xsd:sequence>
      <xsd:element name="Employee_Id" type="xsd:string"/>
      <xsd:element name="Address_Type" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Response">
    <xsd:sequence>
      <xsd:element name="First_Name" type="xsd:string"/>
      <xsd:element name="Last_Name" type="xsd:string"/>
      <xsd:element name="Care_of_Name" type="xsd:string"/>
      <xsd:element name="Street_Name" type="xsd:string"/>
      <xsd:element name="House_Number" type="xsd:string"/>
      <xsd:element name="Name_of_City" type="xsd:string"/>
      <xsd:element name="Zipcode" type="xsd:string"/>
      <xsd:element name="County" type="xsd:string"/>
      <xsd:element name="Country" type="xsd:string"/>
      <xsd:element name="Email_Address" type="xsd:string"/>
      <xsd:element name="Telephone_Number" type="xsd:string"/>
      <xsd:element name="Status" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Based on this document, a program given the correct inputs can construct a valid XML message to call this functionality in an SAP system.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.9. What are XML namespaces?

The fact that XML as a standard does not dictate semantics means that we can define any document structures we wish. I may decide on a certain structure for a purchase order, but if you decide on a different structure, we will need to translate each other's messages or else we will not be able to communicate.

The flexibility that XML provides by allowing us to create any tags we want can also lead to problems. An XML document can be constructed from multiple sources. Since these different sources may use tags with the same names, conflicts are possible. Consider the seemingly innocent tag, `<Order_Number>`. Do we mean the order number used when we purchased the item or the order number assigned when we sold the item? Without further definition, such a common term can be ambiguous. The basic solution to this problem is to use an XML technology known as namespaces.

We declare XML namespaces and then preface a tag with the namespace name. In this way, we remove all ambiguity about which tag we are using.

The schema in [Example 14-1](#) uses namespaces. In the beginning of the file, we see the namespace declaration, using the `xmlns` element:

```
<xsd:schema
  targetNamespace="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

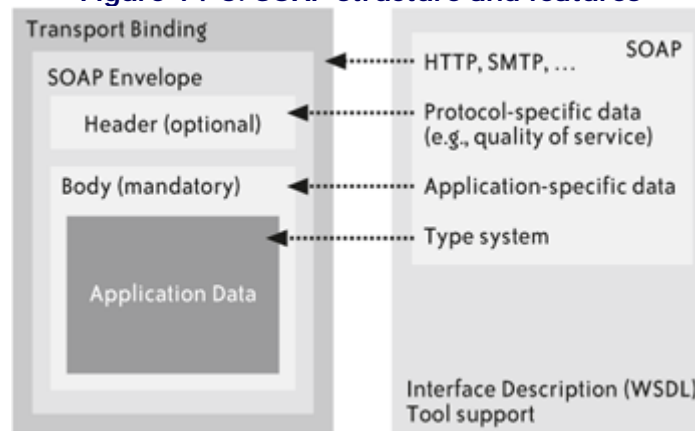
14.10. What is SOAP?

SOAP (which once stood for Simple Object Access Protocol but is now known simply by its rather catchy acronym) is an important web services standard that describes the message structures passed at runtime to call a web service.

A SOAP message is an XML document that describes the operation to be performed and the parameters to pass to the application. Optionally, a SOAP message can include other information that describes how a recipient should process the SOAP message.

A SOAP message, which is always an XML message, wraps the service call in a SOAP envelope, as shown in [Figure 14-6](#). A SOAP envelope includes an optional header area for additional processing information such as quality of service or processing restrictions that have meaning only if the service application is coded to process those headers. A SOAP envelope also includes a mandatory body section that describes the functions to be called and the parameters that are passed to the service.

Figure 14-6. SOAP structure and features



A strength of SOAP as a code wrapper is that a SOAP message is a text file that is generally passed via HTTP or HTTPS and therefore can cross corporate firewalls. Other interoperability standards such as CORBA, DCOM, and RPC cannot cross firewalls and are therefore not suitable for web services.

[Figure 14-7](#) shows a simplified representation of a SOAP request being exchanged via HTTP. The HTTP header starts with a POST request, followed by the path to the applications, `/Accounts/Savings/` and the HTTP version. The second line specifies the host that is processing this request. The next two lines in the HTTP header show the SOAP HTTP binding.

The body of the HTTP request contains the SOAP message. It begins with a mandatory envelope element followed by a SOAP header containing the transaction element. This element has some meaning to the application processing the web service call but no general meaning to the SOAP standard. A body element describes the action to perform—namely, deposit—and an account number, currency type, and amount for the action. This SOAP message allows us to pass an application instruction over the Internet—namely, to deposit \$200 to account 112233 and to do it in a way that can cross corporate firewalls. This service may generate a response SOAP message that indicates success or failure.

Figure 14-7. Sample of a basic SOAP message over HTTP



User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.11. What is WSDL?

While SOAP is used as the format of the exchanged message, WSDL is used to describe the callable web service. WSDL is the XML vocabulary for describing web services, where they are located, and how they can be called. Using the WSDL description of a web service, we can code a client typically with at least some code generation to reduce the amount of hand coding that must be done.

WSDL documents describe the what, how, and where of web services, as illustrated in the simplified WSDL document shown in [Figure 14-8](#). First, consider the what. A port type describes the abstract interface, which is the web service to be called. A port type can have one or more operations. An operation describes the functionality to be called, and the input, output, and fault message types associated with it. The different messages are built from built-in or custom data types. The data types themselves are defined using the XSD language.

To describe how to call a web service, a binding specifies the transport protocol for exchanging messages, such as HTTP, HTTPS, SMTP, FTP, and so on. The service can be bound to multiple protocols if the service provider can accept them. Finally, a WSDL port details the specific network address at which the service can be found.

[Example 14-2](#) shows an actual WSDL document for a `GetAddress` function.

Figure 14-8. A simplified WSDL document

<pre><?xml version="1.0" encoding="utf-8" ?> <definitions></pre>		
<pre> <types> ... <element name="qty" type="string" minOccurs="0"/> ... </types></pre>		<p>What A <i>portType</i> describes the abstract interface (web service type) of the web service and the operations it performs.</p>
<pre> <message name="POMessageIn"> ... <part name="Quantity" type="qty"/> ... </message></pre>		<p>Each contained operation can have an <i>input</i>, an <i>output</i> and a number of <i>fault</i> messages.</p> <p>Different messages are built from built-in or custom data types.</p>
<pre> <portType name="POPortType"> <operation> ... <input message="POMessageIn" /> ... </operation> </portType></pre>		<p>Data types are defined using XML Schema</p>
<pre> <binding name="SOAP" portType="POPortType"> ... SOAP/HTTP binding definition ... </binding></pre>		<p>How A <i>binding</i> specifies exactly one protocol for the operations of a portType.</p>
<pre> <service name="OrderWineService"> <port name="Order" binding="SOAP"> <address location="http://www.dijan.fr/Order/" /> </port> </service></pre>		<p>Where A <i>port</i> defines the web service endpoint by specifying a single network address.</p>
<pre></definitions></pre>		

Example 14-2. A WSDL file

```

<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
  targetNamespace="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://sap.com/demo/WS/GetAddress/prepared"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <!-- XSD schema types shown in Example 14-1 -->
  </wsdl:types>
  <wsdl:message name="Request_Message">
    <wsdl:part name="parameters" element="tns:Request_Message"/>
  </wsdl:message>
  <wsdl:message name="Response_Message">
    <wsdl:part name="parameters" element="tns:Response_Message"/>
  </wsdl:message>
  <wsdl:portType name="Z_WS_Address_Test">
    <wsdl:operation name="Z_Get_Address">
      <wsdl:input message="tns:Request_Message"/>
      <wsdl:output message="tns:Response_Message"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="Z_WS_Address_TestSoapBinding"
    type="tns:Z_WS_Address_Test">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Z_Get_Address">
      <soap:operation soapAction="" />
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Z_WS_Address_TestService">
    <wsdl:port name="Z_WS_Address_TestSoapBinding"
      binding="tns:Z_WS_Address_TestSoapBinding">
      <soap:address
        location=
          "http://iwdfvm1035.wdf.sap.corp:8000/Z_WS_ADDRESS_TEST?sap-client=100" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

The WSDL file describes the web service: how to format the messages that will be exchanged to implement the service and the network location for accessing the service.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.12. What is UDDI and how does it relate to SAP?

How does a potential client get the WSDL file? Essentially, there are two options. We can pass the WSDL file itself (or its location) to our partners, or we can publish the WSDL in a directory so that potential clients can find the service. The Yellow Pages, or directory, for web services is a somewhat complicated protocol named Universal Description, Discovery, and Integration, or UDDI. The UDDI registry includes metadata that can be used to search for services by name, ID, category, type, and so on.

UDDI registries can be public or private, can be used for test or production, and can be used in marketplaces or public exchanges. The registry is a structured database of organizations, the services they offer, and the technical descriptions—namely, the WSDL documents—for those services. In a sense, UDDI is a sort of DNS for web services.

SAP's UDDI registry at <http://uddi.sap.com> allows SAP and other organizations to publish service descriptions of their associated WSDL files. For corporate use, SAP NetWeaver includes a UDDI server. [Chapter 8](#) describes the Enterprise Services Repository. Although the Enterprise Services Repository includes WSDL files, that is its only UDDI-like function. The Enterprise Services Repository includes everything needed for creating enterprise services, and WSDL files are a vital but small subset of what the Enterprise Services Repository contains.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.13. How can we ensure that web services will interoperate?

Given the proliferation of standards and the importance of web services to the future of e-commerce, it is important to have a method to guarantee interoperability between organizations at different levels. The Web Services Interoperability Organization, or WS-I, is tasked with accelerating the adoption of web services by reducing the cost associated with standards adoption. WS-I promotes collections of technical standards called profiles that are used for web services between organizations. The initial profile, called WS-I Basic Profile 1.0, includes HTTP, XML, XSD, SOAP, WSDL, and UDDI. Companies know that if another company complies with the basic profile, interoperability is assured via open standards. WS-I also provides documentation and test tools for working with services and the basic profile.

User name: VLADIMIR BLAGOJEVIC

Book: Enterprise SOA: Designing IT for Business Innovation

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

14.14. What about web services security?

Security is a paramount concern for web services, especially since XML is essentially text. Many layers of security can be applied to XML exchanges. For example, HTTPS can be used to encrypt the data stream on the wire. Message-level security provides another level of security and encryption that can be very useful. To achieve this more granular level of security, WS-Security describes how to include encryption and digital signature headers in SOAP messages. WS-Security is one of many WS-* protocols or standards. These standards address many advanced web services requirements, such as reliable message exchange, stateful interaction, message exchange profiles, and so forth. Most of these are in the process of being developed or are not yet widely adopted.

Web services are no longer new technology; they are the basic building blocks of SOAs. Nonetheless, having an understanding of web services basics will serve you well as you delve deeper into the details of enterprise services development. With this understanding of web services in mind, [Chapters 15](#) and [16](#) examine methodologies for creating web services—and enterprise services—using SAP NetWeaver.