14

Web Services and SOA for DBA, Data Architects, and Others

Things should be made as simple as possible, but not any simpler.

—Albert Einstein

The promise of Web services is simplified interfaces for application-to-application interaction in heterogeneous and distributed environments. This chapter describes the key technologies that enable Web services (i.e., XML, WSDL, SOAP, UDDI) and then peeks at the bigger picture, the service-oriented architecture (SOA). After reading this chapter, you will be well armed to understand database Web services, our final destination.

14.1 Web Services 101

In traditional Web interactions, humans interact with applications through browsers, which interpret HTML to produce graphical displays and accept user inputs. As depicted by Figure 14.1, Web services, by contrast, allow *application-to-application interaction* through XML messages *irrespective of the implementations, the location, and the platforms* of the client and the server application modules. The World Wide Web Consortium (W3C) defines Web services as "software applications or components identified by a URL that describes their interfaces (i.e., services or operations they furnish) and their binding in XML, and that can be accessed by client-applications using XML messages and Internet protocols."

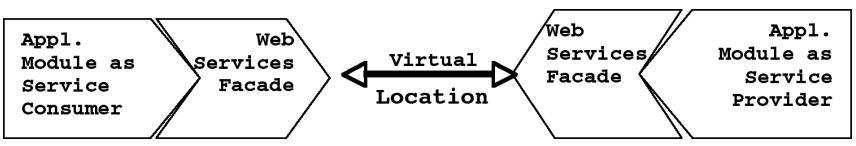


Figure 14.1 *A Simplistic View of Web Services*

The key benefits are interoperability (language/platform neutral); simpler, flexible, and dynamic integration (provision to support new protocols, dynamic service discovery and binding, defined interfaces foster composition); automation (application-to-application interaction, services orchestration/workflow); and time to market (reuse of existing applications). These benefits are made possible by the standards technologies (formats, protocols, and description) that compose the Web services stack. We can distinguish the core Web services technologies and then the infrastructure/deployment requirements, as well as higher-level technologies.

14.1.1 Core Web Services Technologies

The core technologies that make up Web services are XML, SOAP, WSDL, and UDDI. Figure 14.2 depicts how these technologies come into play. Let's look briefly into each of these.

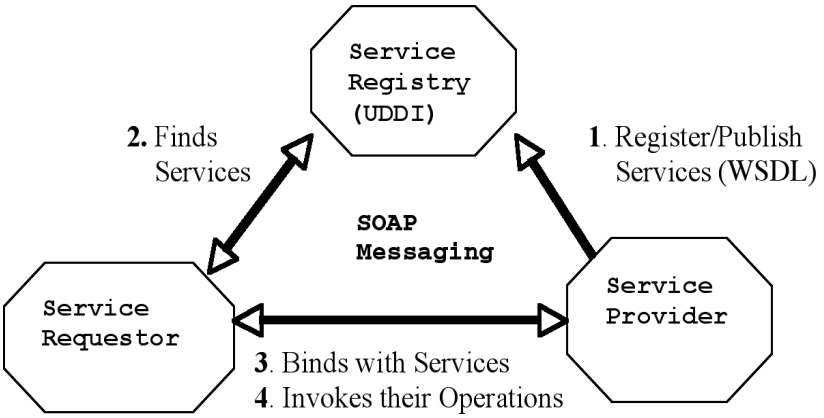


Figure 14.2 *Web Services Interaction*

XML

The eXtended Markup Language (XML) is the *lingua franca* of Web services (and many other technologies); it serves as the base language for type definition, service description, data format, data transfer, messaging, discovery, security, and so on. Web services requesters and providers exchange information using XML documents, which are formatted according to either XML Document Type Definition (DTD) rules or XML schema rules (XSD).

As briefly described in Chapter 8, XML Schema Definition (XSD) is a W3C recommendation, an alternative to DTD, for describing *the structure, content, and semantics of XML documents*, thereby allowing all parties

involved to have a common and exact understanding of the document in question. The XSD defines which elements the document may contain, their attributes, and their relationship. Because XML does not come with predefined tags, the role of XML namespaces and XML schemas is integral for a shared understanding of the document.

An XML document contains:

- A prolog or processing instruction, which starts and ends with "?" into brackets, and are the only "predefined" tags in XM; everything else is self-described, which is the "eXtensibility" in XML.
- XML namespace (`xmlns`), namespace instance (`xmlns:xsi`), and an association of the schema and the instance (`xsi:schemaLocation`). According to the W3C,[1] "XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references."
- A root element (address), which may have attributes and a cascade of subelements:

```
<element>
  <subelement>
    <field1>.....</field1>
  </subelement>
</element>
```

Here is a basic XML document representing an address:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<! This is a comment -->
<Address
      xmlns=http://www.basicxml.org/AddressDoc
      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
      xsi:schemaLocation=
          "http://www.basicxml.org/AddressDoc
           file:./AddressDoc.xsd">
      <Street> 24 Benin Drive</Street>
      <City> San Francisco </City>
```

1. http://www.w3.org/TR/REC-xml-names/.

```
        <State> California</State>
        <Country>USA</Country>
</Address
```

XML defines simple types, such as `string, Base64Binary, hexBinary, integer, date, positiveInteger, genativeInteger, nonNegatveInteger, nonPositiveInteger, decimal, boolean, time, dateTime, duration, date, Name, QNane, anyURI, ID, IDREF`, and so on.

XML allows user-defined complex types such as the following:

```
<complexType name"myComplexType">
<sequence>
<element name="tname" type="string" minOccurs="0"/>
<element name="description" type="string"  minOccurs="0"/>
…
</sequence>
<attribute name="objid" type="ID"/>
<attribute name="objref" type="IDREF"/>
</complexType>
```

See Chapter 15 for more information.

XML documents can be processed and consumed directly by an application as a character stream or serialized and parsed by XML serializers and parsers (e.g., `DOM, SAX, StAX`).

XML documents can also be displayed using XSL and XSL transformers (XSLT) and style sheets. There is a whole range of XML standards, APIs, tools, editors, and utilities. However, their coverage is beyond the scope of this book. There are tons of online resources, tutorials, and FAQs about XML, such as the following:

- http://www.w3.org/XML/Schema
- http://www.w3.org/TR/xmlschema-0
- http://www.w3schools.com/xml/default.asp

WSDL

The Web Services Description Language (WSDL) is an XML language, based on a general-purpose XML schema, for describing how to access a service, including:

- Messages and their style (i.e., document versus RPC)
- Bindings of abstract operations and messages to a concrete network protocol
- Format of messages that a service can receive (see SOAP messages formats discussion)
- Supported operations, their parameters, and return types
- Location of the service

A WSDL structure is made up of two substructures: a service interface definition and a service implementation definition. The service interface definition contains initially an implementation neutral (abstract or reusable) of the service that will be instantiated by the service implementation and includes messages, types, port type, and binding.

- `Message`: Describes supported messages and parameters; may contain parts for RPC arguments.
- `PortType`: Describes the interface of a service (i.e., the set of supported operations with input message, output message, and fault message).
- `Operation`: A message signature, part of `PortType`; can be one-way of request-response.
- `Binding`: How to invoke the operations—that is, style (e.g., `RPC`), transport (e.g., `HTTP`, `SOAP`, `HTTP/MIME`, `SMTP/MIME`), encoding, and security. *Note:* WSDL does not require SOAP.
- `Types`: Describes XSD-related items and user-defined types.

The service implementation definition contains implementation details of the service, including a collection of WSDL ports service and a concrete endpoint or port.

- `Service`: Set of endpoints or port type ports (i.e., groups endpoints into service).
- `Port`: Concrete endpoint corresponding to a WSDL binding (i.e., network address of the Web service).

Do you have to learn how to write WSDL? Remember, "Things should be made as simple as possible." The WSDL is generated for each service by the Web services framework when you deploy or publish the service (see Chapter 15). As depicted in Figure 14.2, the requester retrieves the WSDL from a registry (see UDDI later) but could also find/receive it by other means; it then interacts with the service, either via a dynamic invocation (remember the old-time `CORBA DII`?) or more commonly via a static client or proxy. The other good news is that you don't have to code against the WSDL, because the Web services framework generates the Web service client (also called proxy) corresponding to your platform (e.g., Java client, .Net client), which shields you from SOAP/WSDL programming.

The following listing is a fragment of the WSDL generated for the Google Spell Check Web Service (see a complete demo in Chapter 16):

```
 <definitions name="GoogleSearch"
targetNamespace="urn:GoogleSearch"
 …
  <types>
    <xsd:schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:GoogleSearch">
    <xsd:complexType name="GoogleSearchResult">
    <xsd:complexType name="ResultElement">
…
    </xsd:schema>
  </types>

  <message name="doSpellingSuggestion">
    <part name="key" type="xsd:string" />
    <part name="phrase" type="xsd:string" />
  </message>
  <message name="doSpellingSuggestionResponse">

  portType name="GoogleSearchPort">
    <operation name="doSpellingSuggestion">
```

```
      <input message="typens:doSpellingSuggestion" />
      <output message="typens:doSpellingSuggestionResponse" /
>
    </operation>
  </portType>

  <binding name="GoogleSearchBinding"
type="typens:GoogleSearchPort">
      <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="doSpellingSuggestion"></operation>
  </binding>

  <service name="GoogleSearchService">
    <port name="GoogleSearchPort"
     binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/
beta2" />
    </port>
   </service>

</definitions>
```

A Web service can be assembled from an existing WSDL, using a top-down approach. See Chapter 5 of the Oracle Application Server 10.1.3 Web Services Developer's Guide. Database Web services start from database functionality and correspond therefore to the bottom-up approach. Also, WSDL specifications versions (1.1, 1.2) are not covered but can be seen at the following Web sites:

- http://www.w3.org/TR/wsdl
- http://www.w3.org/2002/ws/desc/

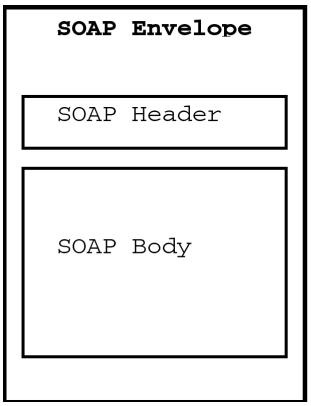
SOAP

The Simple Object Access Protocol (SOAP) is a simple, lightweight, XML-based RPC protocol, which defines a common shape as well as the processing rules for messages exchanged between objects (remember old-time IIOP?). SOAP enables Web services interoperability by providing XML messaging, which is transport neutral (e.g., `HTTP/HTTPS`, `FTP`, `SMTP`, Messaging Protocols, `RPC`, `BEEP`); implementation language neutral (e.g., Java, C/

C++, C#, J#, JScript, Perl, VB, PL/SQL, SQL), and platform neutral (e.g., Java, .NET).

As depicted by Figure 14.3, a SOAP message is represented as a SOAP envelope, which contains an optional header (i.e., SOAP header) and a mandatory body (i.e., SOAP body).

- The `SOAP Envelope` must define a namespace for the envelope (i.e., `xmlns:SOAP-ENV`) and a namespace for the encoding style (covered later; i.e., `SOAP-ENV:encodingStyle`).
- The optional `SOAP Header"` is used for metadata, security/authentication, transaction management, routing/delivery, and other attributes. As an example, in SOAP 1.1, the `actor` attribute, when present, specifies the final destination of the header; if the recipient is not the final, it must forward/route the message to its final destination (this is replaced by the `role` attribute in SOAP 1.2).
- The mandatory `SOAP Body` carries the message content (i.e., payload) or instructions. The SOAP body may contain a SOAP fault message with the `Code`, the `Reason,` and optionally the `Details` of the fault. When large data such as LOB cannot fit within the SOAP body, the "SOAP with Attachment" specification (`SwA`) allows carrying a SOAP envelope within a `MIME` multipart structure and referencing `MIME` parts from the SOA envelope.




```
<SOAP-ENV:Envelope
 xmlns=:SOAP-ENV="SoapEnvelopURI"
 SOAP-ENV:encodingStyle="...">

 <SOAP-ENV:Header>
  ...
 </SOAP_ENV:Header


<SOAP-ENV:Body>

  ...

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Figure 14.3
SOAP Envelope

The following code is a fragment of a SOAP message sent to the Google Spell Check Web Service (see Chapter 16):

```
<?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:doSpellingSuggestion
      xmlns:ns1="urn:GoogleSearch"

      SOAP-ENV:encodingStyle=
           "http://schemas.xmlsoap.org/soap/encoding/">
    <key
xsi:type="xsd:string">00000000000000000000000000000000</key>
    <phrase xsi:type="xsd:string">Nelson Mandelo</phrase>
    </ns1:doSpellingSuggestion>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

The SOAP messages can be transmitted either synchronously or asynchronously using either a Remote Procedure Call (RPC) or document messaging style:

- In the RPC messaging style, the structure of the SOAP body (i.e., the request) must conform to section 7 of the SOAP 1.1 specification and specify the method, the parameters, and the procedure (its URI). Upon execution, the service provider sends back a SOAP message (i.e., the response). The request/response messages are exchanged synchronously.
- In the document messaging style, the structure of the SOAP body is less constrained, because it does not contain direct method invocation, but rather an XML document, which may contain method invocation as well as other information (e.g., status, notification); the XML schema is defined by the `type` element in the WSDL (see WSDL previously). The messages can be exchanged synchronously or asynchronously.

The physical representation (i.e., the wire format) of the SOAP messages exchanged between the service requester and the service provider is control-

led by settings in the WSDL, particularly the value of the `use` attribute of the WSDL `binding`, which specifies the encoding styles of the message:[2]

- *Literal use*: The encoding and interpretation of the SOAP body is dictated by the specified XML schema.
- *Encoded use*: The `encodingStyle` attribute of the SOAP body specifies the encoding and interpretation rules to use. The SOAP 1.1, section 5, defines a set of serialization rules for structures, object graphs, and so on.

The following classification—sometimes referred to as the WSDL style—combines the encoding style or wire format (literal versus encoded) and the message exchange style (RPC versus document) to determine how the SOAP message is interpreted/processed:

- *Document-literal format*: Specifies a document style message with "literal" format. The SOAP body maps only one parameter; other parameters are mapped by the SOAP header. This format conforms to the Web services interoperability.
- *Wrapped-document-literal:* A variant of document-literal used primarily in .NET environments. In this format, the parameters of the method are wrapped by a schema definition.
- *RPC-encoded format*: Specifies RPC message style, with "encoded" format. It is primarily used for object graphs.
- *RPC-literal format*: Specifies an RPC message style, with "literal" format.

REST

SOAP is by no contest *the* standard XML messaging technology for Web services. However, the Representational State Transfer (`REST`) is an alternative XML messaging technology being adopted by Web retailers such as Yahoo, Amazon, and Google for online shopping and search applications. `REST` Web services use XML documents directly as message payload, instead of SOAP envelopes and `HTTP GET/POST` and XML to share `URIs`

2. See "SOAP Encoding." in section 5 of the SOAP 1.1 specification at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383512 and section 3 of the SOAP 1.2 specification at http://www.w3.org/TR/2003/REC-soap12-part2-20030624/#soapenc.

between distributed applications; the applications or end users perform state transitions (navigate following links), and the next state (next page) is transferred to the application or end user (and rendered). `REST` is not a standard, but rather an architectural style; however, it is based on standards such as `HTTP, URL, XML/HTML/GIF/JPEG,` and MIME types (e.g., `text/xml, text/html, image/gif, image/jpeg`). Products such as Oracle Application Server 10.1.3 support `REST` with the literal encoding style `(use=literal).`

UDDI

The Universal Description Discovery and Integration (UDDI) is an industry standard (supervised by OASIS) for publishing and locating Web services and their descriptions (dynamic discovery) in registries. UDDI also designates a collection of peer directories, known as UDDI Business Registries (UBR), which host information about businesses and their services.

The UDDI specification includes:

- An XML schema, which defines the documents that describe the key UDDI data structures relative to: (1) the business/organization/individual that offers services (i.e., businessEntity); (2) the set of services being offered (i.e., businessService); (3) binding information for invoking and using the services (i.e., bindingTemplate); more technical information for connecting to services (i.e., tModel); the relationship between entities (i.e., publisher Assertion)—new in UDDI version 2; and standing orders or requests (i.e., Subscriptions)—new in UDDI version 3.
- A set of UDDI APIs (Inquiry, Publication, Security, Subscription) for querying/browsing (also known as Inquiry API) the directories for details about a given Web service, such as supported security and transport protocols, and publishing/registering information in directories. The Information in UDDI registries is stored/classified in three parts: (1) white pages (general business information), (2) yellow pages (taxonomies industry/category/location), and (3) green pages (technical information). Vendors furnish Web-based interfaces as well.
- The Replication and Custody Transfer APIs for replicating directory entries between peer UDDI registries for failover and custody transfer.

Browse the following links for more UDDI resources:

```
www.oasis-open.org
www.uddi.org
www.uddi4j.org
```

Final Thoughts

Throughout this section, we have seen that Web services are standards-based, XML centric, platform independent, programming language neutral, self-contained, self-describing, and self-advertising software modules allowing application-to-application interaction over the Web.[3] Let's revisit Figure 14.2 to see how the core Web services technologies (i.e., SOAP, WSDL, UDDI) come into play:

Step 1: The provider sends the WSDL to the UDDI registry.

Step 2: The requester retrieves the WSDL from the registry.

Step 3: Based on information in the WSDL, the requester binds to the provider.

Step 4: The requester invokes operations on the provider.

All interactions are SOAP message based. This concludes a high-level "tour d'horizon" of the base technologies that enable Web services, and invokes operations on the services, but while you can publish and deploy Web services using the base technologies, you will rapidly be confronted with solving the usual enterprise deployment requirements of security, reliability, manageability, and so on—and this time with a new dimension: the Web! In order to address these requirements, a set of new Web services–related specifications, recommendations, de facto standards, APIs, and component models are being consolidated into the SOA, which is our next topic.

14.2 Service-Oriented Architecture (SOA): The Bigger Picture

Java brought portability across systems, but Web services go one step beyond and bring interoperability. It is no surprise then that all of the players in the IT industry are actively engaged in making Web services a reality.

3. Take a deep breath first!

As already mentioned, however, it takes more than the core Web services technologies to interoperate across platforms, languages, corporations, and the Web. This section looks at the bigger picture of Web services: the service-oriented architecture (SOA), its specifications, recommendations, de facto standards, APIs, component models, and so on.

SOA 101

To get a feel for the pervasiveness of the concept of SOA, just Google "service-oriented architecture" and you get about 17 million hits. This is probably not the best definition, but SOA can be summarized as *the ability to implement a corporate architecture based on Web services standards, wherein client applications with proper authorization, simply register, discover, and use available services reliably and securely.*

The implications, requirements, and concerns for implementing SOA include advertising, business process, description, discovery, architecture, interoperability, management, messaging and reliable messaging, security, transport, policy, transaction, and user interface.

The following charts list some of the various specifications, standards, initiatives, recommendations, working drafts, and so on that address SOA infrastructure services requirements. Some of these services will be widely adopted and persist, while others will be subsumed by new ones and disappear.

Advertising

	Description	**Status/Sponsor**
UDDI	Web-based registry to publicize and locate services	OASIS standard

Architecture

	Description	**Status/Sponsor**
EbXML	Electronic Business XML	Superseded by various WS specs.
ebSOA TC	Electronic Business Service-Oriented Architecture	TC formed

Business Process

	Description	Status/Sponsor
BPEL4WS	Notation for specifying business process behavior	OASIS Standard
WS-Choreography	Ability to compose and describe the relationships between WS	W3C, WG formed
Web Services Choreography Description Language	Describes peer-to-peer collaborations	W3C, Working draft
Business Process Execution Language (BPEL)	Continue business process language work	OASIS, TC formed

Description

	Description	Status/Sponsor
XML	Extended Markup Language	W3C Recommendation
WSDL	Model and format for describing Web services	W3C
Web Services Architecture	Reference Architecture	W3C

Discovery

	Description	Status/Sponsor
Web Services Inspection Language (WS-Inspection)	Allow WS requester to drill down into WS	Proposal

Interoperability, Specification Profiles

	Description	Status/Sponsor
WS-I Basic Profile	Mandate support for SOAP 1.1, WSDL 1.1, HTTP 1.1, HTTP binding (or HTTPS), and XML Schema (1 and 2)	WS-I
Devices Profile	Interoperability between devices and Web services	

Implementation

	Description	Status/Sponsor
Framework for Web Services Implementation(FWSI TC)	Methods for broad, multiplatform, vendor-neutral implementation	OASIS, TC formed

Management, Auditing, Logging

	Description	Status/Sponsor
WS-Management	Interoperable and cross-platform management using Web services	Microsoft
WS-Management Catalog	Available endpoints or "resources," summary forms, compatible actions, schemas, and WSDL	Microsoft
WS-Manageability	Set of capabilities for discovering the existence, availability, health, performance, usage, and control of a Web service	OASIS, Spec Published

Messaging and Reliable Messaging

	Description	Status/Sponsor
WS-ReliableMessaging	Guaranteed delivery, guaranteed duplicate elimination	OASIS, Spec published
SOAP	Peer-to-peer RPC message exchange	W3C
WS-Addressing	Enables messaging systems to support message transmission in a transport-neutral manner	W3C, Spec published
MTOM (Attachments)	SOAP Message Transmission Optimization Mechanism (Supersedes WS-Attachments)	W3C, Working draft
WS-Enumeration	Enables an application to ask for items from a list of data that is held by a WS	Spec published
WS-Eventing	How to construct an event-oriented message exchange pattern using WS Addressing	Spec published

Messaging and Reliable Messaging (continued)

	Description	Status/Sponsor
WS-Transfer	Defines how to invoke a simple set of familiar verbs (Get, Post, Put, and Delete) using SOAP	Spec published
SOAP-over-UDP	Defines a binding of SOAP to use datagrams, including message patterns, addressing requirements, and security considerations	Spec published
Reliable HTTP (HTTPR)	Guarantees reliable delivery of HTTP packets	IBM

Metadata

	Description	Status/Sponsor
WS-Policy	Describes and communicates the policies of a WS (service requirements, preferences)	Spec published
WS-PolicyAssertions	Details messaging-related assertions for use with WS policy (encoding, language)	Spec published
WS-PolicyAttachment	Specifies three attachment mechanisms for using policy expressions with WS	Spec published
WS-Discovery	Multicast discovery protocol to locate services	Spec published
WS-MetadataExchange	Bootstrap communication with a WS, defines request-response message pairs to retrieve WS-Policy, WSDL, and XMl Schema	Spec published

Security

	Description	Status/Sponsor
WS-Security: SOAP Message Security	Message Integrity and confidentiality	OASIS proposal
WS-Security: Username-Token Profile	How a consumer will specify UsernameToken	OASIS proposal

Security

	Description	Status/Sponsor
WS-Security: X.509 Certificate Token Profile		
`WS-SecureConversation`, WS-Federation, and WS-Authorization	Authenticate message exchanges, security context exchange, and trust	Microsoft, Verisign & IBM proposal
WS-SecurityPolicy	Security policy assertions	Microsoft, Verisign, RSA, and IBM proposal
WS-Trust, WS-Policy, WS-Privacy	Trust, constraints of security policies, and privacy pactices	Microsoft, Verisign and IBM proposal
WS-Federation Active Requester Profile		
WS-Federation Passive Requester Profile		
WS-Security: Kerberos Binding		
Web Single Sign-On Interoperability Profile		
Web Single Sign-On Metadata Exchange Protocol		
XML-Signature	Integrity, message and user authentication	W3c recommendation
SAML	Security Assertion Markup Language	OASIS standard
XML Key Management Specifications (XKMS)	Public-key infrastructure integration	W3C Note
WS-Security Profile for XML-Based Tokens		

Transport

	Description	Status/Sponsor
WS-Coordination	Protocols to coordinate distributed applications	BEA, IBM, Arjuna, Microsoft, Hitachi, and IONA
WS-AtomicTransaction	Transaction completion, volatile two-phase commit, and durable two-phase commit	BEA, IBM, Arjuna, Microsoft, Hitachi, and IONA
WS-BusinessActivity	Protocols for the business activity coordination	BEA, IBM, Arjuna, Microsoft, Hitachi, and IONA

Policy and Binding

	Description	Status/Sponsor
WS-PolicyAttachment	Mechanisms for attaching policy expressions with one or more subjects or resources	BEA, IBM, Microsoft, and SAP
WS-PolicyAssertions	Messaging related assertions for WS-Policy	BEA, IBM, Microsoft, and SAP

Transaction

	Description	Status/Sponsor
WS Composite Application Framework (WS-CAF 3 parts: WS-CTX, WS-CF, WS-TXM)	Ability to compose an application out of multiple Web services	Oracle, Sun, Fujistu, Arjuna, IONA
WS-AtomicTransaction	See above	
WS-Coordination	See above	

User Interface

	Description	Status/ Sponsor
Web Services for Remote Portlets (WSRP)	Set of interfaces and related semantics that standardize interactions with components providing user-facing markup	Oracle, SAP, IBM, Microsoft, Sun, BEA Novell, Tibco, Vignette

User Interface

Web Services for Interactive Applications (WSIA)	Standard based on XML and Web services for presenting interactive Web applications to users	IBM, Sun
Web Services Experience Language (WSXL)	Web services-centric component model for interactive Web applications	IBM

14.3 Conclusion

Web services frameworks vendors such as Oracle, Microsoft, IBM, BEA, Sun MicroSystems, and open source players/products are actively integrating these APIs/technologies while hiding their complexity from the developers/assemblers. Now that we have seen the core Web services technologies and the broader SOA landscape, let's look at how to turn your database into a first-class member of your SOA, including exposing database operations as a Web service and invoking external Web services from within the database.