Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools

Web Services Protocols and Technologies

The web services approach is based on a maturing set of standards that are widely accepted and used. This widespread acceptance makes it possible for clients and services to communicate and understand each other across a wide variety of platforms and across language boundaries. This section briefly describes the protocols and technologies that constitute these web services standards, some of which are mainstays of working on the World Wide Web. In addition to these standards, other web services standards are emerging to meet additional requirements -- especially in the areas of security and management. This section also covers some of these emerging web services standards.

XML

eXtensible Markup Language (XML) has become the de facto standard for describing data to be exchanged on the Web. As it's name indicates, XML is a markup language. It involves the use of tags that "mark up" the contents of a document, and in doing so, describe the contents of a document. An XML tag identifies information in a document, and also identifies the structure of the information. For example, the following XML markup identifies some information as `bookshelf`. The XML markup also describes the structure of `bookshelf`. The `bookshelf` structure includes two subordinate items identified as `book`. Each `book` has three subordinate items identified as`title`, `author`, and `price`.

```
<bookshelf>
   <book>
      <title>My Life and Times</title>
      <author>Felix Harrison</author>
      <price>39.95</price>
   </book>
</bookshelf>
```

Although tags such as `<book>` and `<title>` might appear to inherently give meaning to the information they identify, they really don't. Information tagged with XML tags has meaning only if people associate a particular meaning with a particular tag. If people (1) agree on the meaning of a tag, say that the `<book>` tag is used to identify a book, and that `<title>`, `<author>`, and `<price>`tags identify the title, author, and price of the book, respectively, and (2) use those tags consistently, it gives people a way to exchange data. Their applications can send XML documents to each other, and process the information in those documents, relying on the commonly understood meanings associated with the XML tags. Applications capable of interpreting these tags can then process the information according to the meaning of the information and its organization. XML documents have a well-formed structure, for example, each XML tag must have an ending tag (such as `<bookshelf>...</bookshelf>`) and any tag that begins within another tag must end before the end of the other tag, and must be completely nested within that other tag. So that `</title>`, `</author>`, and `</price>` all come before `</book>`, and in that relative order. An XML document is typically associated with a schema that specifies its "grammar rules." In other words, the schema specifies what tags are allowed in the document, the structure of those tags, and other rules about the tags, such as what type of data is expected in a tag (or no data if it's an empty tag). Because valid XML documents must be well-formed and conform to the associated schema, it makes it relatively easy to process XML documents. As a result, XML has been generally adopted as the data language for web services.

SOAP
Though agreeing on the meaning and structure of XML tags makes the use of XML an effective way to exchange data, it's not sufficient for data interchange over the Web. For instance, you still need some agreed-upon protocol for formatting an XML document so that the receiver understands what the main, "payload," part of the message is, and what part contains additional instructions or supplemental content. That's where Simple Object Access Protocol (SOAP) comes in. SOAP is an XML-based protocol for exchanging information in a distributed environment. SOAP provides a common message format for exchanging data between clients and services. The basic item of transmission in SOAP is a SOAP message, which consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. The envelope specifies two things: an XML namespace and an encoding style. The XML namespace specifies the names that can be used in the SOAP message. Namespaces are designed to avoid name clashes -- the same name can be used for different items, provided that the names are in different namespaces. The encoding style identifies the data types recognized by the SOAP message. If a header is provided, it extends the SOAP message in a modular way. It's important to understand that as a SOAP message travels from a client to a service, it can pass through a set of intermediate nodes along the message path. Each node is an application that can receive and forward SOAP messages. An intermediate node can provide additional services such as transform the data in the message or perform security-related operations. The SOAP header can be used to indicate some additional processing at an intermediate node, that is, processing independent of the processing done at the final destination. Typically, the SOAP header is used to convey security-related information to be processed by runtime components. The body contains the main part of the SOAP message, that is, the part intended for the final recipient of the SOAP message.

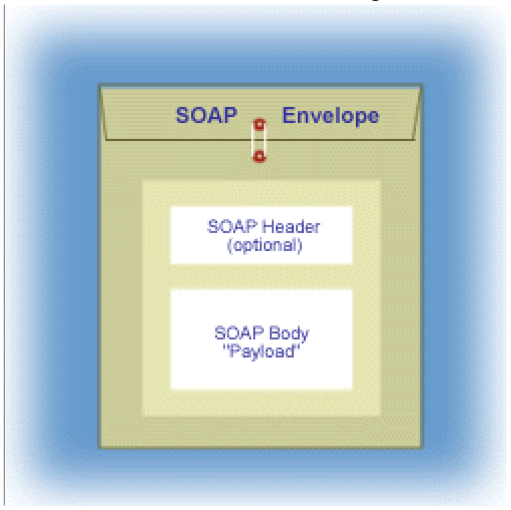

Figure 3: A SOAP Message (Conceptually)

Here's an example of a SOAP message designed to retrieve the price of a book. Notice the `<SOAP-ENV: Envelope>`, `<SOAP-ENV:Header>` and `<SOAP-ENV: Body>` elements that mark the envelope, header, and body parts of the SOAP message, respectively. The attribute value`mustUnderstand=1` in the header means that the receiver of the SOAP message must process it.

```
<SOAP-ENV: Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:
    encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Header>
           <t:Transaction xmlns:t="some-URI">
              SOAP-ENV:mustUnderstand="1">
           </t:Transaction>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
           <m:GetBookPrice xmlns:m="some-URI">
              <title>My Life and Times</title>
              <author>Felix Harrison</author>
           </m: GetBookPrice>
        </SOAP-ENV:Body>
</SOAP-Envelope>
```

A related standard, SOAP Messages With Attachments, specifies the format of a SOAP message that includes attachments (such as, images). SOAP messages (with or without attachments) are independent of any operating system or platform and can be transported using a variety of communication protocols, such as HTTP or SMTP. WS-I Basic Profile 1.0 references SOAP 1.1.

WSDL

How does a client know what format to use in making a request to a service? For that matter, how do the client and the service know what the request means? The answers to these questions are provided by information in an XML document, called a WSDL document, that contains a description of the web service's interface. A WSDL document contains information specified in Web Service Description Language (WSDL), as defined in the WSDL specification. WSDL defines an XML schema for describing a web service. To uncover the description for a Web service, a client needs to find the service's WSDL document. One way, perhaps the most typical way, to do this is for the client to find a pointer to the WSDL document in the web service's registration, which can be in aUDDI registry or an ebXML registry/repository. A typical scenario is that a business registers its service. The registry entry includes a pointer to a WSDL file that contains the WSDL document for the service. Another business searches the registry and finds the service. A programmer uses the interface information in the WSDL document to construct the appropriate calls to the service.

A WSDL document describes a web service as a collection of abstract items called "ports" or "endpoints." A WSDL document also defines the actions performed by a web service and the data transmitted to these actions in an abstract way. Actions are represented by "operations," and data is represented by "messages." A collection of related operations is known as a "port type." A port type constitutes the collection of actions offered by a web service. What turns a WSDL description from abstract to concrete is a "binding." A binding specifies the network protocol and message format specifications for a particular port type. A port is defined by associating a network address with a binding. If a client locates a WSDL document and finds the binding and network address for each port, it can call the service's operations according to the specified protocol and message format.

Here, for example, is a WSDL document for an online book search service. Notice in the example that the WSDL document specifies one operation for the service: `getBooks`:

```
<operation name="getBooks" ...
```

The input message for the operation, `BookSearchInput` contains a string value named `isbn`:

```
<complexType>
    <all>
       <element name="isbn"
       type="string"/>
    </all>
```

The output message for the operation, `BookSearchOutput` returns a string value named `title`:

```
<complexType>
    <all>
       <element name="title"
             type="string"/>
    </all>
```

Notice too that the WSDL document specifies a SOAP protocol binding. The style attribute in the`binding` element specifies that the receiver should interpret the payload in the SOAP message as an RPC method call:

```
<soap:binding
     transport="transport=http://schemas.xmlsoap.org/soap/http"
           style="rpc"/>
```

Alternatively, the style attribute in a `binding`element could specify `"document"`. In that case, a complete document (typically an XML document) would be exchanged in the call. In fact, the document style of binding is more typical of services in an SOA. One of the advantages of passing an XML document to a service instead of an RPC-style message is that it tends to be easier to validate and to apply business rules.

The WS-I Basic Profile 1.0 specifies the constructs in WSDL 1.1 that be can be used to ensure interoperability. It also clarifies some of the construct descriptions in the WSDL 1.1 specification.

UDDI and ebXML

As mentioned earlier, an SOA can also include a service that provides a directory or registry of services. But how can a service be described in a registry so that a program looking for that service can easily find it and understand what it does?

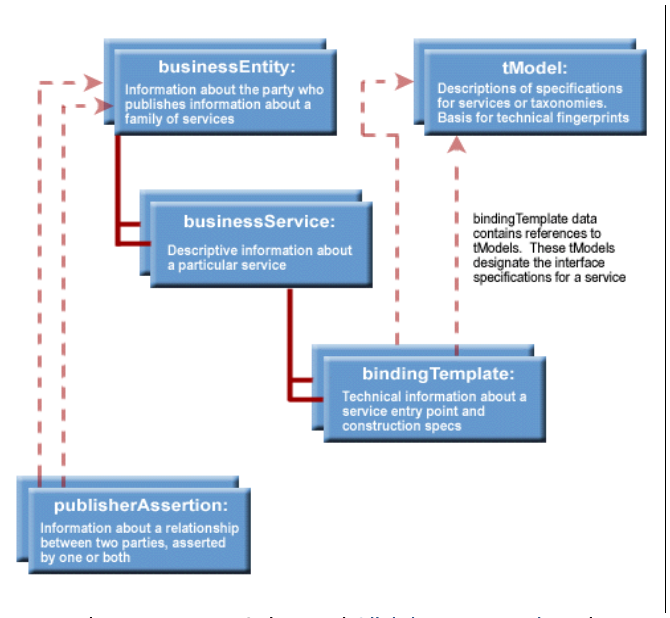
businessEntity:
Information about the party who publishes information about a family of services
tModel:
Descriptions of specifications for services or taxonomies. Basis for technical fingerprints
businessService:
Descriptive information about a particular service
bindingTemplate data contains references to tModels. These tModels designate the interface specifications for a service
bindingTemplate:
Technical information about a service entry point and construction specs
publisherAssertion:
Information about a relationship between two parties, asserted by one or both

Figure 4: UDDI Schema *(Click image to enlarge)*

The Universal Description, Discovery, and Integration (UDDI) specifications define how to publish and discover information about services in a UDDI-conforming registry. More specifically, the specifications define a UDDI schema and a UDDI API. The UDDI schema identifies the types of XML data structures that comprise an entry in the registry for a service. Think of a UDDI registry as a "Yellow Pages" for web services. Like the Yellow Pages directory for phone numbers, a UDDI registry provides information about a service such as the name of the service, a brief description of what it does, an address where the service can be accessed, and a description of the interface for accessing the service. The accompanying figure illustrates the schema. It shows the five types of XML data structures that comprise a registration.

Here is an example that shows part of a complete `BusinessEntity` structure for a hypothetical company named BooksToGo. The company provides various web services, including an online book ordering service.

```
<businessEntity businessKey="35AF7F00-1419-11D6-A0DC-000C0E00ACDD"
authorizedName="0100002CAL"
  operator="www-3.ibm.com/services/uddi">
<name>BooksToGo</name>
<description xml:lang="en">
   The source for all professional books
</description>
<contacts>
<contact>
<personName>Benjamin Boss</personName>
<phone>
  (877)1111111
</phone>
</contact>
</contacts>
```

The API describes the SOAP messages that are used to publish an entry in a registry, or discover an entry in a registry. Here, for example, is a message that searches for all business entities in a registry whose name begins with the characters "Books":

```
<find_business generic="2.0" xmlns=um:uddi-org:api-v2">
<name>Books%</name>
</find_business>
```

UDDI 2.0 is part of the WS-I Basic Profile 1.1
Note that other registry approaches, with their own specifications, exist. For example, ebXML is a comprehensive business-to-business framework that includes a registry as well as other information needed for business collaboration in a global electronic marketplace. The framework was developed as a joint effort between the Organization for the Advancement of Structured Information Standards (OASIS) and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), and includes representation from an extremely wide set of businesses and other entities around the world (including standards bodies). The registry contains a Collaboration-Protocol Profile (CPP) and a Collaboration-Protocol Agreement (CPA). A CPP is an XML document that contains information about a business (in ebXML terms, a business is referred to as a "Party") and the way it exchanges information with another Party. A CPA is an XML document that describes the specific capabilities that two Parties have agreed to use in a business collaboration.

Emerging Standards

Standards such as XML, SOAP, UDDI, and WSDL address the basics of interoperable services. They ensure that a client can find a needed service and make a request that both the client and service understand, irrespective of where the client and service reside or what language the client or service is coded in. But for a web services-based SOA to become a mainstream IT practice, other standards (what many people consider "higher-level" standards) need to be added and adopted. This is especially true in the areas of web service security and web service management. Various standards organizations, such as the World Wide Web Consortium (W3C) and (OASIS), have drafted standards in these areas that promise to gain universal acceptance. Two emerging standards of special interest are WS-Security and WS-BPEL.

WS-Security

WS-Security is a standard released by OASIS in March 2004. It describes security-related enhancements to SOAP messaging that provide for message integrity and confidentiality. Integrity means that a SOAP message is not tampered with as it travels from a client to its final destination. Confidentiality means that a SOAP message is only seen by intended recipients. WS-Security uses security tokens to enable SOAP message security and integrity. A security token is a collection of claims made by the sender of a SOAP message (such as the sender's identity). A sender is authenticated by combining a security token with a digital signature -- the signature is used as proof that the sender is indeed associated with the security token. The standard also provides a general-purpose mechanism for associating security tokens with messages, and describes how to encode binary security tokens.
WS-Security is quite flexible and can be used with a wide variety of security models and encryption technologies, such as Public-key infrastructure (PKI) and Kerberos, as well as the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol that provides basic end-to-end security communication on the Internet.

Here is an example taken from the WS-Security standard that illustrates a SOAP message with a security token. The example also lists and describes the lines that demonstrate WS-Security enhancements.
Other important technologies are emerging in the security area. One of them is Security Markup Assertion Language (SAML). As described by the OASIS Technical Committee responsible for it, SAML "is an XML-based framework for exchanging security information. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain." Through SAML, multiple services can exchange security information. As a result, services could do things like enable a single sign-on approach, where a single security entry (of say a user ID and password) could be used to sign in to multiple, related services.
Building on web services security standards such as WS-Security and SAML, the Liberty Alliance Project, a global consortium for open federated identity standards and identity-based web services, has delivered a number of specifications for identity-based web services. Conformance to these standards will enable web services to use a single identity framework. This will support, among other things, single sign-on, that is, a user will only have to sign in once, even if the user's request invokes a number of services, each requiring authentication.

WS-BPEL

Many business processes, such as the handling of a purchase order, involve multiple steps performed in a specific sequence, potentially requiring the invocation and interaction of multiple services. For this to work properly, the service invocations and interactions need to be coordinated (service coordination is also known as "orchestration") WS-BPEL (Web Services Business Process Execution Language), also identified as BPELWS, BPEL4WS, or simply BPEL, is an XML-based language that is used to coordinate web services across a single business process. WS-BPEL uses WSDL to describe the web services that participate in a process and how the services interact. For example, the following WSDL entry describes one of the services in the purchase order process:

```
<portType name="purchaseOrderPT">
   <operation name="sendPurchaseOrder">
      <input message="pos:POMessage"/>
      <output message="pos:InvMessage"/>
      <fault name="cannotCompleteOrder"
             message="pos:orderFaultType"/>
   </operation>
</portType>
```

The following entry describes part of the flow of the purchase order process:

```
<sequence>
      <receive partnerLink="purchasing"
               portType="lns:purchaseOrderPT"
               operation="sendPurchaseOrder"
               variable="PO">
      </receive>
      <flow>
         <links>
            <link name="ship-to-invoice"/>
            <link name="ship-to-scheduling"/>
         </links>
```

An OASIS Web Services Business Process Execution Language Technical Committee has been established to continue work on the BPEL specification.

Source: http://www.oracle.com/technetwork/articles/javase/wsprotocols-136863.html