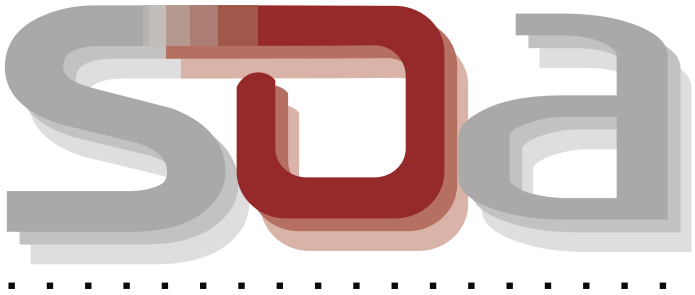
Torry Harris



Web Service Interaction Models

Torry Harris

SOA

Overview

One usually gets to hear the terms 'RPC style' and 'Document style' in the context of web services and SOAP protocol. What exactly do they mean and how do they differ from each other? Under what conditions should we use which one? While most Web services are built around remote procedure calls, the WSDL specification allows for another kind of Web services architecture: *document style*, in which whole documents are exchanged between service clients and servers.

This article provides a brief introduction to the two styles of web-services interaction and details the benefits and challenges associated with each one. It also covers the two encoding types namely the XML encoded and literal, present for each RPC and document style interactions.

1. Introduction

"*A Web Service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols*" - W3C

Web Services define a platform-independent standard, based on XML, to communicate within distributed systems. They allow applications to expose their business functionality as a service, invoked by other applications over the World Wide Web, using the SOAP/XML protocol. These services are self-describing and can be published and inquired in the repository. Web Services Description Language (WSDL) is an XML-formatted language used to describe how the service is bound to a messaging protocol, particularly the SOAP messaging protocol. A WSDL SOAP binding can be either a Remote Procedure Call (RPC) style binding or a Document style binding. In addition to this, a SOAP binding can also have an XML encoded use or a literal use. The style has nothing to do with the programming model. It merely dictates how to translate a WSDL binding to a SOAP message.

2. Binding Style and Use

A typical WSDL document consists of the following elements:

- Types
- Message
- portType for the abstract definitions
- Binding and service for the concrete specification

It is the binding element that we need to take a closer look at, in order to understand the style of the Web Service. A WSDL binding describes how the service is bound to a messaging protocol like SOAP. The <wsdl:binding> element of the WSDL contains a pair of parameters that influence the form of the resulting SOAP messages: binding style (RPC or document) and use (encoded or literal).

```
<wsdl:binding name="Config1Binding" type="prt0:CreditLimitLocalWebServiceVi_Document">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
         <wsdl:operation name="creditLimitCheck">
               <soap:operation soapAction="" />
               <wsdl:input>
                    <soap:body use="literal" parts="parameters" />
               </wsdl:input>
               <wsdl:output>
                   <soap:body use="literal" />
               </wsdl:output>
         </wsdl:operation>
 </wsdl:binding>
```

The Style Attribute:

The 'Style' attribute specifies the style of the binding as either 'RPC' or 'document'. The value of this attribute also affects the way in which the body of the SOAP message is constructed.

- RPC: The structure of an RPC style <soap:Body> element needs to comply with the rules specified in the SOAP 1.1 specification. According to these rules, <soap:Body> should contain a single element, that is named after the operation, and all parameters must be represented as sub-elements of this wrapper element.
- Document: The content of <soap:Body> is specified by the XML Schema defined in the <wsdl:type> section. It does not need to follow specific SOAP conventions. In short, the SOAP message is sent as one "document" in the soap body element.

The Use Attribute:

The 'Use' attribute specifies the encoding rules of the SOAP message and its value can be either 'encoded' or 'literal', indicating whether the message parts are encoded using certain encoding rules, or whether the parts define the concrete schema of the message. The parties in the web services exchange can agree on a predefined encoding scheme or use an XML schema to define the data type.

- Literal: The rules to encode and interpret the SOAP body are specified by an XML schema.
- Encoded: The rules to encode and interpret the SOAP body are in a URL specified by the "encodingStyle" attribute.

3. RPC Style Web Services

Remote procedure call (RPC)-Style Web Services are tightly coupled and interface-driven, meaning that the business methods of the underlying implementation determine how the Web Service works.

In this style, the clients invoke the Web Service by sending parameter values to the Web Service, and receive return values from the Web Service. RPC-style Web Services are tightly coupled owing to the fact that the parameters sent and the return values need to conform to the description in the web service's WSDL file.

RPC-style Web services are typically characterized as synchronous, meaning that the client sends the request and waits for the response, for the duration of time required to completely process the request. It continues with the remainder of the processing, only after getting the response.

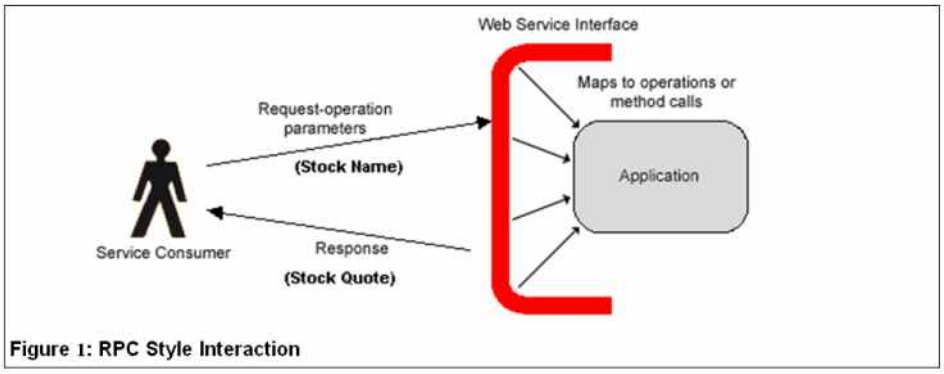


Figure 1: RPC Style Interaction

In the above example, the client sends a SOAP request to a Web service, the SOAP request contains the name of the method (myMethod) to be executed in the Web service, along with the parameters (stock name) that the method will need when it is executed. The Web Service converts this request to appropriate objects and executes the operation. It then sends the response as a SOAP message to the client. On the client side, this response is converted to appropriate objects and the required information is returned (out) to the client.

WSDL for RPC Style

```
.....
<message name="myMethodRequest">
        <part name="x" type="xsd:string"/>
</message>
<message name="myMethodResponse">
        <part name="y" type="xsd:float"/>
</message>
<portType name="PT">
        <operation name="myMethod">
                <input message="myMethodRequest"/>
                <output message="myMethodResponse"/>
        </operation>
</portType>
<binding name="IStringServiceBinding" type="tns:IStringService">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
.....
```

SOAP request for RPC Style

```
<soap:envelope>
        <soap:body>
                <myMethod>
                        <x xsi:type="xsd:string">Enron</x>
                </myMethod>
        </soap:body>
</soap:envelope>
```

The major benefits of RPC style web services are:

- Platform Independence: Clients and servers can use different programming languages or technologies, to implement their respective sides of the interface.
- Simple & Understandable: For RPC style web services, the WSDL is very simple and straightforward. Since RPC is a paradigm that most developers are already familiar with, the learning curve is a short one. With a good IDE support, the developer need not worry about the lower-level details such as the network protocols being used etc.
- Industry Support: RPC Web service has got industry wide support, which not many other RPC technologies have gained till date. Web services and the associated standards — especially XML, SOAP, and WSDL — are almost universally supported. Most commercial software vendors have introduced or plan to introduce support for these technologies into their products and tools.
- Add-ons for non-functional aspects: The coolest feature of SOAP — and thus one that is available for RPC-style Web services — is its support for extensibility by means of SOAP headers. In contrast to IIOP, there's a clearly specified model for passing any sort of contextual information with your method invocations. XML, together with name spaces provides the necessary technology to compose one or more of the multitude of WS-specifications that exist for different purposes.

Some of the disadvantages of RPC-style messaging include:

- Strong coupling: An RPC style web service is meant to be relatively static and any changes, such as the number, order, or data types of the parameters, to the interface, would break the contract between the client and the server.
- Synchronization: RPC style web services are generally meant for synchronous applications. However, one might want to use the asynchronous feature of the web service so as to avoid the waiting time for the user. RPC web services can be made asynchronous but that adds another level of complexity and is thus not advisable.
- Marshaling and serialization overhead: Marshaling and serializing XML is more expensive than marshaling and serializing a binary data stream. With XML, at least on one side of the interface, or possibly both, involves some parsing in order to move data between internal variables and the XML document.

The coupling and synchronization issues are common to RPC-based component technologies. However, the marshaling and serialization overhead is greater for RPC-style messaging and places this messaging style at a relative disadvantage. With today's high-speed processors and networks however, the performance overhead can be combated.

4. Document-Style Web Service

Document-Style web services, also known as 'Message-Style', are loosely coupled and document driven. They are characterized as asynchronous, with the potential for highly complex document structures, which are more often used for data-oriented programming rather than process-oriented programming. In a document-based interaction, the service consumer interacts with the service, using complete documents (that are processed as a whole). These documents typically take the form of an xml, which is defined by a common, agreed upon schema, between the service provider and the service consumer. The document exchanged could also be in a format other than an XML.

For example, consider an Automobile Manufacturing company that accepts purchase order requests from the dealers. The dealer would submit the entire bulk order to the manufacturer at a given time. This is like submitting a message to a queue for asynchronous processing. These interactions may be 'long-surviving' in nature, as the manufacturing company may need to execute a long-running business process in order to respond appropriately to the request.

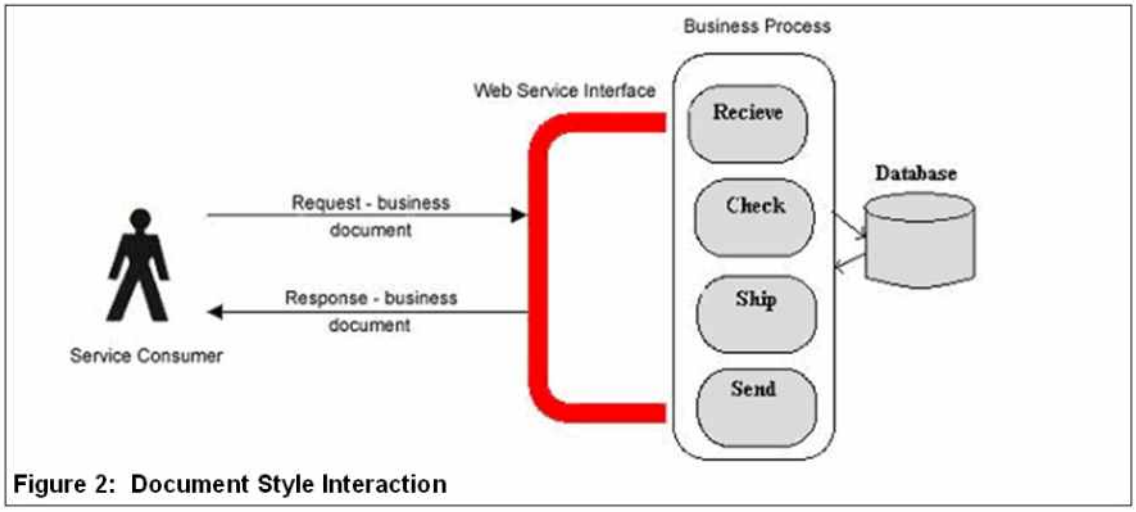


Figure 2: Document Style Interaction

WSDL for Document/literal Formatting

```
....
<types>
   <schema>
      <element name="PurchaseOrder">
         <complexType>
            <sequence>
               <element name="productId" type="xsd:int"/>
               <element name="quantity" type="xsd:int"/>
            </sequence>
         </complexType>
   </schema>
</types>
<message name="PurchaseOrderRequest">
   <part name="parameters" element="PurchaseOrder"/>
</message>
<message name="empty"/>
<portType name="PT">
   <operation name="PurchaseOrder">
      <input message=" PurchaseOrderRequest"/>
      <output message="empty"/>
   </operation>
</portType>
<binding  name="PurchaseServiceBinding" type="tns:PurchaseService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
....
```

SOAP Request for Document/literal Formatting

```
<soap:envelope>
    <soap:body>
        <PurchaseOrder>
            <productId>123</ productId >
            <quantity>10</quantity >
        </ PurchaseOrder >
    </soap:body>
</soap:envelope>
```

The Benefits of Document style web services are:

- Utilization of XML: The XML specification allows complex data to be described in an open format that is easily readable, self-describing, and self-validating. When a Web service uses document messaging, it can use the full capabilities of XML to describe and validate a high-level business document.
- Loosely Coupled: Usually, with web services, changing an interface would cause all of the applications that rely on a specific method signature to break. With document messaging, many enhancements and changes can be made to the XML schema without breaking the calling application.
- Asynchronous Processing: Unlike RPC style, the purpose of Document style web services is not for request/response semantics, though it can be achieved through this style.

A few disadvantages of Document style messaging include:

- No standard service identification mechanism: In this style, the client and server must agree on a service identification mechanism. One option is to include service information in the document's header. Another option is to name the element in the body tag of the message for the services that need to process the payload that the elements contain. The third option is to perform structure or content analysis in order to identify the service's need to process the document.
- Marshaling and serialization overhead: Document style messaging suffers from the same drawbacks as RPC style messaging in this context. It incurs overhead in three areas: in using DOM or other techniques to build the XML documents; in using DOM or SAX to parse those documents and extract data values; and in mapping extracted data values and internal program variables.

5. Considerations to decide on the formatting style

The designers can consider the following points while making a decision on the formatting style for the web services:

1. State maintenance: If the stubs generated by a toolkit cannot maintain state, then document style can be used to pass the contents of an entire transaction as an XML document. The service implementation can then ensure the processing sequence and maintain state in the execution of that sequence.
2. Industry standard schemas: If the service consumer is only requesting for information in a pre-defined format, such as those defined by industry standard bodies, a document style message makes more sense, since it is not constrained by the RPC-oriented encoding.
3. Simplicity: RPC style web service is preferred when the interaction is simple and when interfacing with an existing component. The effort to build a service that uses document messaging is usually greater than the effort required to build an RPC message service. This extra effort usually involves the design of an XML schema or support for a preexisting schema, as well as the extraction of relevant information from a document. In contrast, an RPC message only requires the design of the method interface, from which it will automatically marshal and unmarshal the parameters.
4. Validation: If the service is accepting or returning a complex XML structure, a document style is better suited, since the XML can be validated against the schema, prior to calling the service.
5. Performance: Marshalling and un-marshalling parameters to XML, in memory, can be an intensive process in RPC style as well as document style. The RPC-encoded scheme is the least performing because of the extra processing overhead in encoding the payloads. However, document style services can choose alternate parsing technologies like SAX and StAX to optimize and improve performance.

6. Conclusion

When designing a Web service, we need to consider all of the options that the current WSDL specification provides us. Before starting, consider how the service will be used, who will be using it, and the type and volume of information that needs to be exchanged. If the document style Web service seems to be the answer as per the requirements, use it without worrying about the extra resources that might be required. Though designing and developing a document style Web service may require a little extra effort, in many cases the effort will pay off in the quality of information and the reliability of the exchange.

Regarding the combination of style and use, some of these combinations are rarely used in practice, such as document/encoded. In general, the literal use is gaining importance, and as far as RPC/encoded is concerned, the Web Services Interoperability Organization (WS-I) in its Basic Profile Version (1.0a of August 2003), ruled out the use of SOAP encoding with web services. Document/literal and RPC/literal will be the only allowed style/use combinations in future.

Torry Harris Business Solutions (THBS) is a US based IT service provider with development facilities in India and China. The company, started in 1998, has for several years delivered a large variety of middleware services to enterprise clients around the world. Now, with a large pool of highly skilled technologists and rapidly growing, the company remains focused on the middleware and integration space, implementing large projects across the US, Europe, the Middle East and the Far East. The company is committed to Service-oriented Architecture (SOA), which it sees as the logical movement to follow the phenomenon of distributed computing in the late nineties, where THBS was clearly the market leader in implementing the offshore/onsite delivery model. Further information can be found at www.thbs.com/soa