CORBA and Web Services

PJ Murray and Elizabeth Golluscio

Cape Clear Software

CORBA and Web Services (July 2002)

Trademarks

Contents

The Emergence of Web Services

Web Services have attracted a lot of attention over the past year as a means of building and deploying software to simplify development and systems integration. Web Services are ideal for application integration of internal systems or for linking software over the Internet. Web Services technologies are based on open standards recommended by the World Wide Web Consortium (W3C). The standards enjoy unprecedented industry support from major IT suppliers such as IBM, Microsoft, and Sun Microsystems along with the main CORBA vendors, such as Borland and IONA Technologies. An example of a Web Service is a bank exposing its credit card validation service to partners for use in their electronic business applications. There are examples of live Web Services at www.xmethods.com and www.capescience.com.

Web Services Standards

The key Web Services standards are WSDL, SOAP, and UDDI, which are all based on XML. These standards have been driven by the desire of businesses to find standardized ways of working over the ubiquitous infrastructure (for example, TCP/IP, HTTP, and HTTPS) already in place within organizations. The underlying technology itself is not particularly new, rather it is the result of a number of trends in Internet use and distributed computing technology. Perhaps the most significant aspect of these standards is their very widespread industry acceptance, which displays a level of cooperation that is unprecedented in the IT industry. Even with the support of hundreds of software vendors, the Common Object Request Broker Architecture (CORBA) suffered because Microsoft developed a proprietary alternative, with its COM/DCOM solution. Just as SQL became the standard "grammar" for data, Web Services will become the standard grammar for integration. As such, Web Services will turn the Internet into a reliable and easy-to-use application integration bus, bringing integration capabilities to the mainstream developer. Historically, systems integration was an option only for highly skilled and highly budgeted organizations.

XML

Extensible Markup Language (XML) promises to simplify and lower the cost of data interchange and Web publishing. XML is predicted to play a dominant role as a data interchange format in business-to-business (B2B) Web applications, such as e-commerce and enterprise application integration (EAI). XML is the key to all the other Web Services standards, because it represents a truly interoperable data representation, allowing disparate applications to communication across the enterprise or the Internet.

WSDL

Web Services Description Language (WSDL) is the metadata language that defines how service providers and service requesters communicate with Web Services applications. WSDL is an XML schema that describes network services as collections of communication endpoints that are capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and automate the details involved in communications between applications. Like XML, WSDL is extensible to allow for the description of endpoints and their messages, regardless of what message formats or network protocols are used for communication. WSDL provides the flexibility to define software components that are written in the CORBA, Java/J2EE, or .NET programming models. This layer of abstraction allows SOAP access to applications written on different operating systems, programming languages, and implementation models.

When exposing CORBA back-ends as Web Services, WSDL defines and exposes the CORBA components, detailing their data types, operations, and parameters. WSDL provides all the information that a client application needs to build a valid SOAP invocation. At runtime, this information is mapped by the Web Services platform onto the CORBA component.

SOAP

Simple Object Access Protocol (SOAP) is a protocol for exchanging information in a decentralized, distributed environment. It defines a mechanism to pass commands and parameters between clients and servers. Like Web Services as a whole, SOAP is independent of the platform, object model, and programming language being used. The SOAP protocol is XML-based and consists of three parts:

- An envelope that defines a framework for describing the message content and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.
- A convention for representing remote procedure calls and responses.

Figure 1: The problem of mapping SOAP and IIOP

For CORBA users, a key issue is the translation of SOAP requests to IIOP invocations at runtime. This problem is solved by the new generation of Web Services platforms.

UDDI

A Universal Description, Discovery, and Integration (UDDI) server is the "meeting place" for Web Services. An information database of Web Services, it stores descriptions about companies and the services they offer in a common XML format. Just as businesses list their products and services in a telephone directory, UDDI is used to register services that requesters can then discover and invoke. Web-based applications interact with a UDDI registry using SOAP messages. Both Microsoft and IBM host public UDDI registries, which are kept in sync. In addition, several vendors offer commercial implementations for hosting a "private" UDDI within an intranet environment.

Conceptually, the data in a UDDI registry can be divided into three different types of telephone directories:

- A white-pages section, which provides business contact information.
- A yellow-pages section, which categorizes businesses and their services.
- A green-pages section, which provides technical information about the businesses' services.

A UDDI registry enables an organization to publish its software services, whether written in a CORBA framework or in any other implementation model. A UDDI registry is similar to the CosNaming or CosTrader services.

CORBA and Web Services

CORBA, as defined by the Object Management Group (OMG) since 1992, is an open, vendor-independent architecture and infrastructure for distributed object technology. It is widely used today as the basis for many mission-critical software applications. CORBA vendors have progressively added richer quality-of-service features through the implementation of various CORBA services, like transactions and security.

While CORBA may be the best solution for certain applications, developers often face the significant challenge of Web-enabling these systems. Few client-side, Internet-access products support IIOP. Even if IIOP is available (either pre-installed or downloaded), it is not firewall-friendly. Because IIOP proxies are not widely installed on firewalls, these packets may need to be filtered out before reaching their intended server. Additionally, system administrators are reluctant

to open an IIOP route through the firewall, because it exposes another potential access point for malicious attacks and complicates filtering.

Another issue faced by organizations is the mismatch between the skills of Web developers and authors, and the skills required to build CORBA systems. The typical Web developer may be content using HTML, JavaScript and Visual Basic. Such developers are often uncomfortable building CORBA clients, which require advanced Java or C++ skills. The problem is that much of the business logic they wish to expose, access, or reuse is implemented in CORBA with IDL interfaces!

Because Web Services and CORBA standards are very similar, CORBA users can feel confident that their technical skills can be easily transferred to this new development model, which includes WSDL, SOAP and UDDI. More importantly, CORBA users have the very rare, yet critical, skills necessary to understand the complexity involved in building distributed systems. Making interface design and architectural decisions can impact performance, security, and systems management. These skills, which are essential when conducting Web Services projects, are typical among CORBA developers and architects.

Comparing Technologies

Similar to CORBA IDL, WSDL contains the abstract definition for a service, which defines both types and messages. WSDL also contains a concrete section that defines how the service is contacted, for example, protocol, encoding, and URI details.

The IIOP specification defines a very efficient binary protocol. SOAP is text-based and optionally includes type information as part of the message, which simplifies debugging and traffic monitoring because the message content is human-readable text. The CORBA IDL type system cannot accommodate certain requirements, such as DOC or PDF files as part of the message. The SOAP with Attachments specification allows MIME attachments to be included as part of the message content.

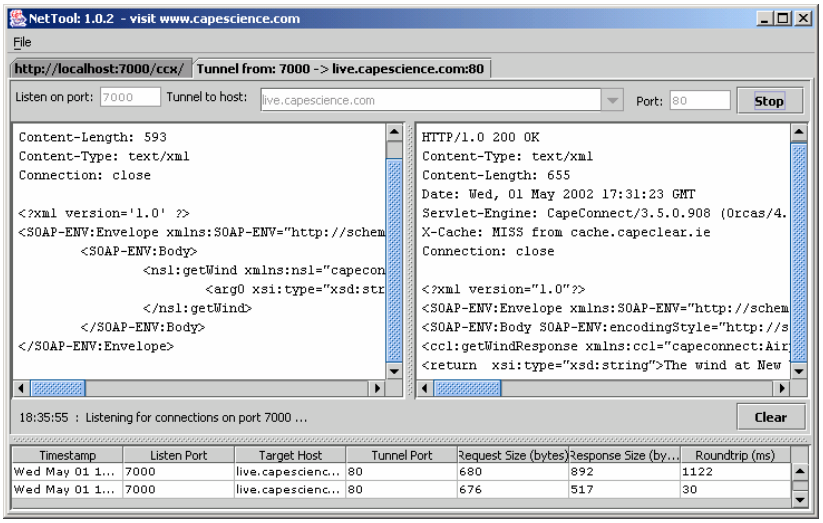

Figure 2: Tracing a SOAP message, using the free NetTool utility

CORBA IDL is bound to IIOP as a transport mechanism, whereas WSDL uses SOAP, which is not tied to any specific transport protocol. Typically, SOAP uses the HTTP protocol, but commercial SOAP implementations already support other protocols such as HTTPS, SMTP, and JMS. A UDDI registry closely corresponds to the CORBA Trader Service, yet there are plans for another version of UDDI that resembles the CORBA Naming Service and will offer a simplified view of its data.

These technical differences are mainly due to the origins of CORBA and its focus as a solution for industrial-strength applications within private or corporate networks. Web Services is focused on lightweight, Internet-based services, which can be reused and combined as required, decoupling clients from the service implementation. As such, Web Services offer a great opportunity to reuse and extend CORBA systems.

Building Web Services from CORBA

In *Joint CORBA to WSDL/SOAP Interworking RFP Initial Submission* (OMG Document mars/02-06-03), the OMG outlines "a natural mapping from IDL to WSDL that is also suitable for a reverse mapping, from the mapped subset of WSDL back to IDL." This submission, led by Cape Clear's Chief Technology Officer Hugh Grant, has the support of industry leaders such as Fujitsu, Hewlett-Packard, and IONA Technologies.

Cape Clear's customers typically want to expose existing CORBA-based logic as one or more Web Services interfaces. This requires the IDL-to-WSDL tool to provide full support for complex data-types and user-defined data constructs. It is not feasible to modify the IDL, because this requires changes to the existing business logic. The following sections describe how the OMG's proposal to

integrate Web Services with CORBA logic works in practice, using the CapeStudio development tool and CapeConnect runtime platform.

When installing CapeStudio and CapeConnect, there is an option to integrate with a specific CORBA ORB. With this configuration, CapeConnect can expose new and existing components deployed in the CORBA ORB as Web Services, as depicted in Figure 3.

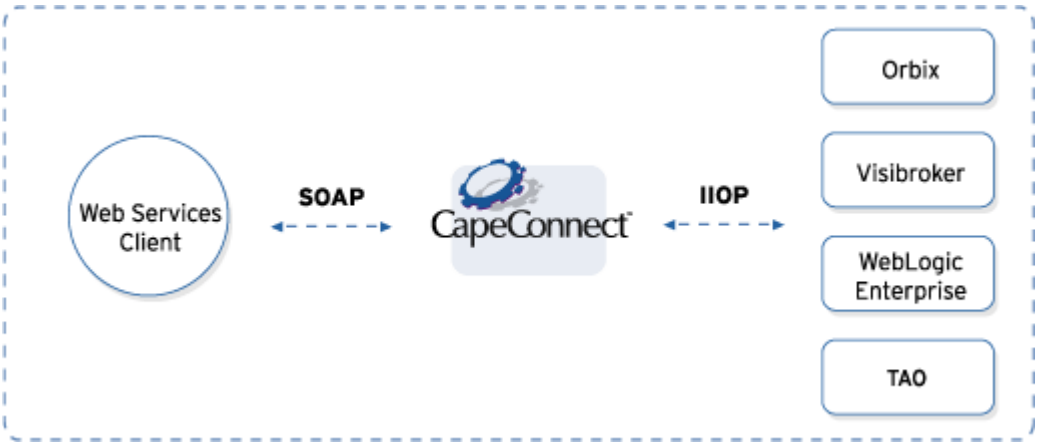

Figure 3: Exposing deployed CORBA ORB components as Web Services

Development

Using CapeStudio, exposing existing application logic developed in CORBA is a simple process that does not require system redesign. CORBA business logic is exposed as Web Services using the following steps:

1. Generate WSDL from the new or existing IDL. CapeStudio's takes an .idl file as input and creates WSDL for the interface.

   The screenshots in Figure 4 illustrate how CapeStudio assists the developer in deploying a back-end CORBA server as a Web Service. In this case, the IDL being mapped to WSDL is one of the examples provided with Borland's VisiBroker ORB.

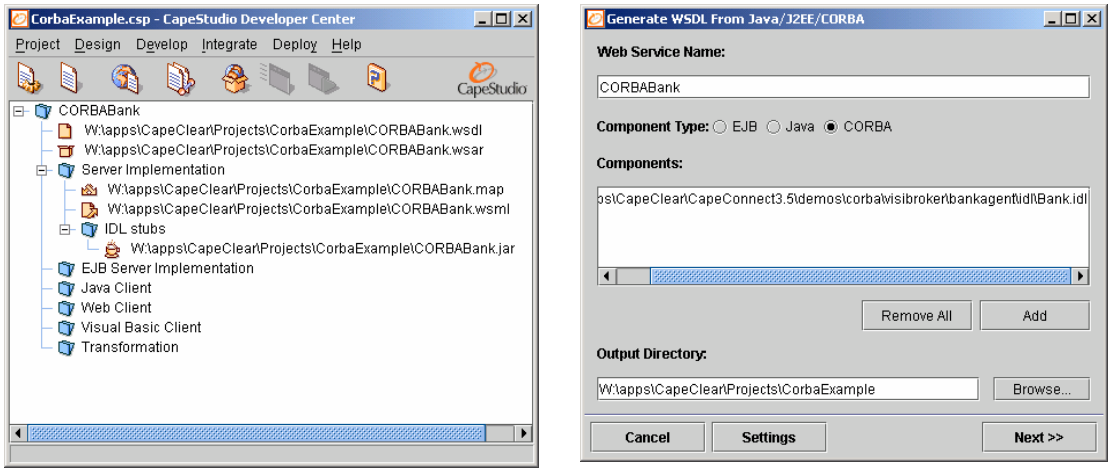

Figure 4: The CapeStudio project view, and generating WSDL from IDL

2. Generate Web Service client proxies and client code. Client applications can be implemented in the developer's language of choice, provided that a SOAP toolkit is available for that language (for example, Microsoft .NET for C#, or SOAP::Lite for a PERL implementation). CapeStudio can generate Visual Basic, Java, or JSP clients from the WSDL generated in step 1.

3. Deploy the Web Service. The WSDL generated in step 1 is packaged and deployed into CapeConnect, using a simple CapeStudio wizard.

   Deploying the Web Service is a simple process. From the CapeStudio Developer Center, you can package and deploy your Web Service, as shown in Figure 5.

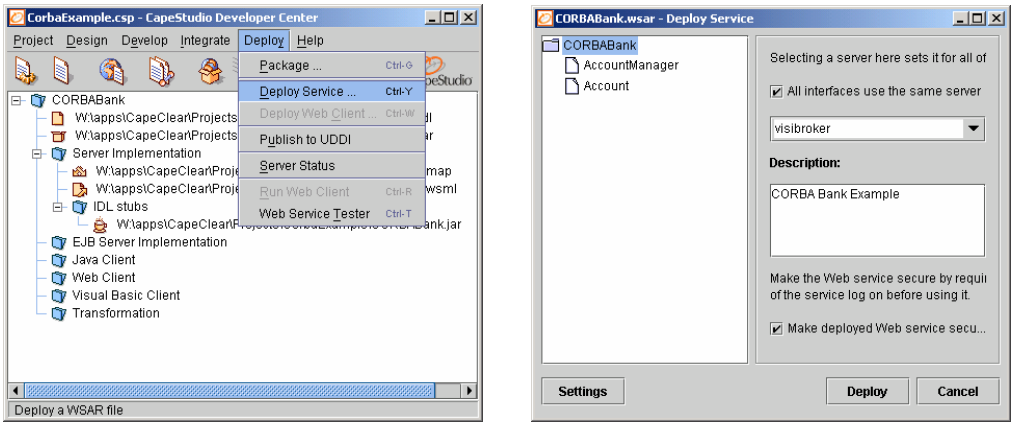

Figure 5: Deploying the Web Service into CapeConnect

Runtime

4. Link SOAP with IIOP requests and responses. At runtime, CapeConnect translates SOAP client requests into IIOP invocations, using the CORBA Naming Service (or a stringified IOR) to locate the required object.

5. Optionally, use a UDDI registry. The CapeConnect platform includes a private UDDI registry that can hold information about Web Services. You can interrogate the registry to list the available Web Services.

6. Test your Web Service. Use the CapeStudio Web Service Tester (or NetTool, as shown depicted in Figure 2) to analyze your Web Service.

CapeStudio includes a UDDI Browser to retrieve information from any public or private UDDI registry. Publishing details of a Web Service to CapeConnect's private UDDI is simplified with the UDDI Publisher tool, which is shown in Figure 6.

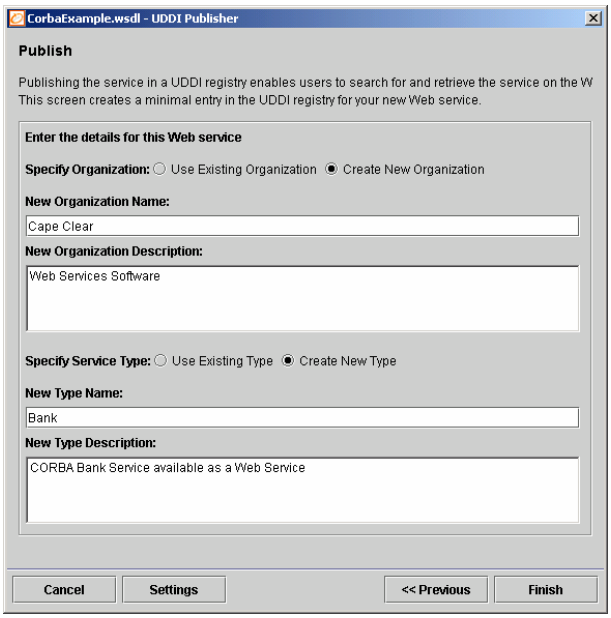

Figure 6: Publishing a Web Service to CapeConnect's UDDI registry

Once deployed, the Web Service is immediately available. CapeStudio's Web Service Tester generates sample SOAP requests from the WSDL, allowing quick verification that the Web Service is functioning correctly.

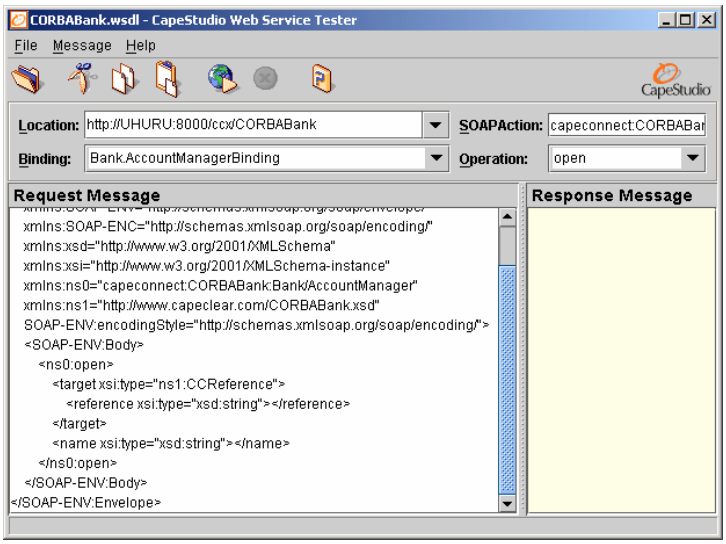

Figure 7: Testing the Web Service with CapeStudio's Web Service tester

In the following example, CapeConnect exposes a VisiBroker CORBA server as a Web Service by generating WSDL from the CORBA IDL. The following CORBA IDL is input:

```
// Bank.idl
module Bank {
 interface Account {
  float balance();
 };
 interface AccountManager {
  Account open(in string name);
 };
};
```

The corresponding WSDL that is generated is:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="VisiBrokerBankAgentExample"
    targetNamespace="http://www.capeclear.com/VisiBrokerBankAgentExample.
    wsdl" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.capeclear.com/VisiBrokerBankAgentExample.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://www.capeclear.com/VisiBrokerBankAgentExample.xsd">
- <types>
```

```
- <xsd:schema
      targetNamespace="http://www.capeclear.com/VisiBrokerBankAgentExample.
      xsd" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <xsd:complexType name="CCReference">
- <xsd:sequence>
 <xsd:element name="reference" type="xsd:string" />
    </xsd:sequence>
    </xsd:complexType>
    </xsd:schema>
    </types>
- <message name="open">
 <part name="target" type="xsd1:CCReference" />
 <part name="name" type="xsd:string" />
    </message>
- <message name="openResponse">
 <part name="return" type="xsd1:CCReference" />
    </message>
- <message name="balance">
 <part name="target" type="xsd1:CCReference" />
    </message>
- <message name="balanceResponse">
 <part name="return" type="xsd:float" />
    </message>
- <portType name="Bank.AccountManager">
- <operation name="open" parameterOrder="target name">
 <input message="tns:open" />
 <output message="tns:openResponse" />
    </operation>
    </portType>
- <portType name="Bank.Account">
- <operation name="balance" parameterOrder="target">
 <input message="tns:balance" />
 <output message="tns:balanceResponse" />
    </operation>
    </portType>
- <binding name="Bank.AccountManagerBinding"
      type="tns:Bank.AccountManager">
 <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="open">
 <soap:operation
      soapAction="capeconnect:VisiBrokerBankAgentExample:Bank/AccountManage
      r#open" />
- <input>
```

```
 <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="capeconnect:VisiBrokerBankAgentExample:Bank/AccountManager
     " use="encoded" />
    </input>
- <output>
 <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="capeconnect:VisiBrokerBankAgentExample:Bank/AccountManager
     " use="encoded" />
    </output>
    </operation>
    </binding>
- <binding name="Bank.AccountBinding" type="tns:Bank.Account">
 <soap:binding style="rpc"
     transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="balance">
 <soap:operation
     soapAction="capeconnect:VisiBrokerBankAgentExample:Bank/Account#balan
     ce" />
- <input>
 <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="capeconnect:VisiBrokerBankAgentExample:Bank/Account"
     use="encoded" />
    </input>
- <output>
 <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="capeconnect:VisiBrokerBankAgentExample:Bank/Account"
     use="encoded" />
    </output>
    </operation>
    </binding>
- <service name="VisiBrokerBankAgentExample">
 <documentation>Comments typed in by user.</documentation>
- <port binding="tns:Bank.AccountManagerBinding"
     name="Bank.AccountManager">
 <soap:address location="http://localhost:8080/ccgw/GWXmlServlet" />
    </port>
- <port binding="tns:Bank.AccountBinding" name="Bank.Account">
 <soap:address location="http://localhost:8080/ccgw/GWXmlServlet" />
    </port>
    </service>
  - <!--

     Created by CapeStudio on Wed May 01 20:02:57 GMT 2002 See
     http://www.capeclear.com for more details

       -->

       </definitions>
```

As shown in this example, WSDL has a very specific syntax and format. One of the key benefits of CapeStudio is that the WSDL is generated automatically, without the need to edit the WSDL or source IDL in any way. CapeStudio also includes an innovative WSDL Editor, which is shown in Figure 8, for viewing and editing WSDL in a graphical way. The use of CapeStudio's tools reduces errors and saves considerable time and effort, both during the learning phase and during development.

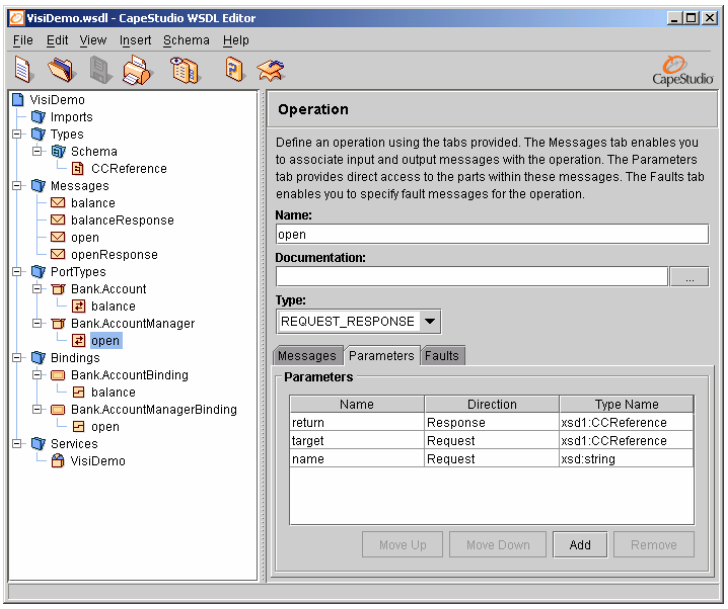

Figure 8: CapeStudio's WSDL Editor

A more detailed tutorial, based on the TAO ORB, is located at www.capescience.com/corba.

Business Value from Web Services

Optimistic visionaries predict a day when "millions of Web Services" are commercially available to businesses (and consumers) for use as the building blocks for modern software systems, and are used internally or externally. However, a more practical approach for an initial project involves the reuse of existing CORBA logic, along with Web Services technology to expose these systems in a new way. This strategy will shift the focus to Web Services benefits (and new integration options) rather than on new development efforts.

There are many immediate uses for CORBA-based Web Services:

- Enterprise Application Integration (EAI): SOAP can be used to integrate CORBA logic with other enterprise systems such as J2EE and .NET.

- Deploying applications across firewalls: SOAP (over HTTP or HTTPS) can be used to integrate applications across firewalls. This is particularly useful for business partner integration or browser-based, customer-facing applications.

- Exposing CORBA applications to new users and development teams: ROI is increased, because mainstream developers (such as JSP or Visual Basic programmers) develop the client-side application, while specialist CORBA programmers write the back-end logic.

- Packaged application integration: Web Services support is being added to many packaged applications. For instance, Cape Clear offers pre-built Web Services for accessing salesforce.com data.

- Component reuse and customization: The UDDI registry and XSLT support in CapeConnect enable the publishing and customization of Web Services. Valuable IT assets can now be reused and customized within the organization, reducing data duplication and software development.

Conclusions

CORBA and Web Services were developed for different reasons using different technologies, yet they are complementary in nature. CORBA provides a mature middleware infrastructure, with robust and scalable features and services, for building mission-critical systems. Recently, there has been much industry analysis about the future of Web Services and its potential for IT innovation and new business models. This should not distract from the fact that Web Services platforms already provide the necessary technology to leverage existing CORBA deployments. More importantly, mainstream developers familiar with scripting language can now access business functionality implemented in CORBA.

Further Reading

Papers and articles on XML and CORBA:

http://xml.coverpages.org/xmlAndCORBA.html

www.omg.org/technology/xml/index.htm

Article on SOAP:

http://www.capeclear.com/clear_thinking/soap.shtml

OMG's Web Services and CORBA information:

http://cgi.omg.org/cgi-bin/doc?mars/02-06-03

http://cgi.omg.org/cgi-bin/doc?mars/2002-06-04

Information on the World Wide Web Consortium (W3C) XML standards:

http://www.w3.org/2000/xp/

Cape Clear's Developer Network:

http://www.capescience.com

Terminology

application programming interface (API)

The specification of how a programmer writing an application accesses the behavior and state of classes and objects.

CapeConnect XML engine

The component that converts SOAP messages into RMI or IIOP calls on Java/EJB or CORBA back-end components. It also converts the results of these calls to SOAP messages and returns them to the client process.

Common Object Request Broker Architecture (CORBA)

An architecture and specification, defined by the OMG, for creating, distributing, and managing distributed objects in a network.

deployment

The process whereby software is installed into an operational environment.

Document Object Model (DOM)

A tree of objects with interfaces for traversing the tree and writing an XML version of it, as defined by the World Wide Web Consortium (W3C) specification.

Document Type Definition (DTD)

A description of the structure and properties of a class of XML files.

Extensible Markup Language (XML)

A flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere.

firewall

A functional unit that protects and controls the connection of one network to other networks. It prevents unwanted or unauthorized communication traffic

from entering the protected network and allows only selected communication traffic to leave the protected network.

Hypertext Transfer Protocol (HTTP)

The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. The HTTP application protocol is based on TCP/IP, which underlies information exchange on the Internet.

Microsoft SOAP Toolkit

A Microsoft programming tool that enables developers to use Microsoft Visual Studio 6.0 to build, expose, and consume Web Services (for example, from Visual Basic 6 applications).

Remote Procedure Call (RPC)

A protocol for requesting a service from another program, which may be located on another computer in the network, without having to understand the network details.

schema

In XML, a conceptual framework that describes the underlying structure of a collection of elements.

Secure Hypertext Transfer Protocol (HTTPS)

A Web protocol developed by Netscape that encrypts and decrypts user page requests as well as the pages that are returned by the Web server. See also Hypertext Transfer Protocol (HTTP).

Simple API for XML (SAX)

An event-driven, serial-access mechanism for accessing XML documents.

Simple Object Access Protocol (SOAP)

An RPC mechanism, based on XML, to access services, objects, and servers in a platform-independent manner.

Uniform Resource Identifier (URI)

A superset of URL, a URI is a necessary part of the HTTP header in a SOAP request.

Uniform Resource Locator (URL)

The address of a resource available on the Internet. It contains the name of the protocol required to access the resource, a domain name identifying a specific computer on the Internet, and a hierarchical description of the resource.

Universal Description, Discovery, and Integration (UDDI)

A meeting place for Web Services, UDDI is a specification for information registries of Web Services. A UDDI repository stores descriptions about companies and the services they offer in a common XML format.

Web Services

Self-describing, self-contained, modular applications that have a WSDL interface and are accessible via SOAP. Web Services enable developers to integrate applications using different platforms, object models, or programming languages.

Web Services Description Language (WSDL)

The common language for describing Web Services. It enables service providers and requestors to communicate with each other, detailing the data types and operations available.

Web Services Meta Language (WSML)

A language, specific to the Microsoft SOAP Toolkit, that enables developers to map Web Services to COM objects. When using the Microsoft SOAP Toolkit with CapeConnect, WSML definitions enable SOAP clients to send and receive complex data types.

XML Schema Definition (XSD)

The World Wide Web Consortium standard for specifying XML schemas. See also Extensible Markup Language (XML).