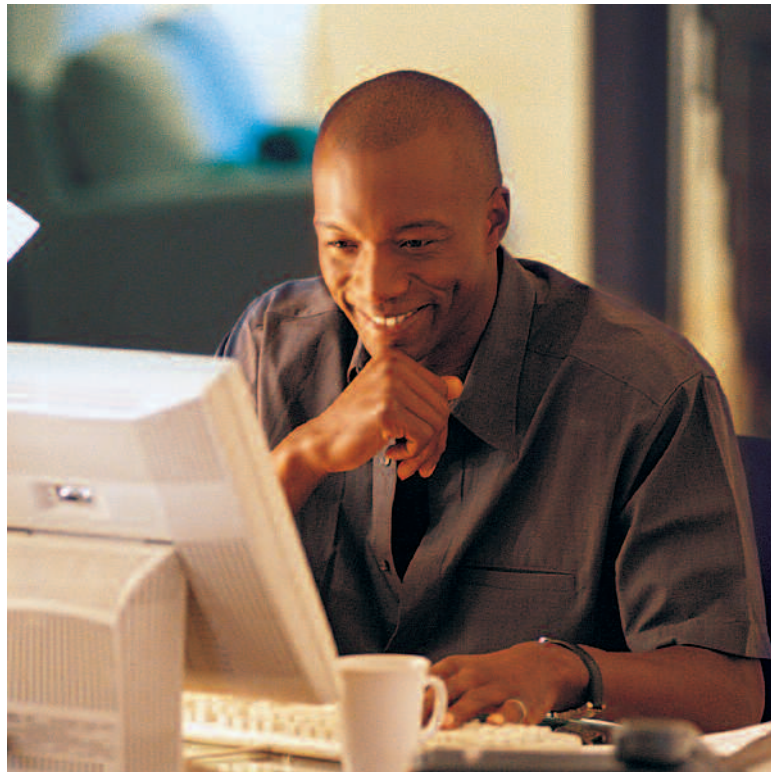# Web Services
in Cincom® VisualWorks®

WHITE PAPER

Cincom In-depth Analysis and Review

Cincom

SIMPLIFICATION THROUGH INNOVATION™

Web Services
in Cincom VisualWorks

WHITE PAPER

Cincom In-depth Analysis and Review

Web Services in VisualWorks

Web services have become a mission-critical component of applications, supporting communications and commerce independent of platforms and programming languages. The technology enables companies to provide public access to proprietary software without distributing the software and data outside their organizations. This is accomplished through a software interface that describes operations that can be accessed over the network through standardized Extensible Markup Language (XML) messaging. Using protocols based on the XML language, web services describe an operation to execute or data to exchange.

Cincom VisualWorks supports web services technology built on such existing and emerging standards as HTTP; XML; XPath, as a language for addressing parts of an XML document; Simple Object Access Protocol (SOAP); Web Services Description Language (WSDL); and Universal Description, Discovery and Integration (UDDI).

Potential web services developed in VisualWorks could be any application: authentication, phrase translation, currency conversion, shipping status lookup, or database queries, to name just a few. VisualWorks imposes no restrictions on the size of the application.

VisualWorks enables developers to create both web services providers and requestors. The service provider is the server side, the platform that hosts the service, while the service requestor is the client side, the application that is interacting with a service. The service requestor can be a browser driven by a user or a program without a user interface, such as another web service.

VisualWorks imposes no restrictions on the size of the application.

Web Services Technologies in VisualWorks

VisualWorks fully supports development of web services by implementing XML, SOAP, WSDL and UDDI standards.

XML

XML is the standard format for data exchange over the web, and it is the essential foundation for deploying web services via SOAP, WSDL and UDDI. The XML web services architecture allows applications written in different languages on different platforms to communicate with each other.

To accommodate this use of XML in VisualWorks, a mechanism is required for mapping XML elements and attributes to Smalltalk objects and back again. VisualWorks provides a robust XML-to-object framework that enables creation of XML documents from Smalltalk objects, as well as creation of Smalltalk objects from XML documents. This capability is the foundation of VisualWorks web services support.

XML-to-object mapping is based on a binding specification, which is an XML document. From this mapping, the engine produces prototype objects. The objects are stored in a registry that maps the prototype objects to tag names in the binding document. This registry is then used by the engine to marshal and unmarshal an XML document.

The XML-to-object mechanism supports marshaling for:

• Simple types to simple types

• Simple types with attributes to complex objects

• Complex elements to complex objects

• Elements, attributes, or text to instance variables of Smalltalk objects

SOAP

Simple Object Access Protocol (SOAP), the communications protocol for XML web services, is a specification that defines the XML format for messages. The SOAP component of VisualWorks wraps the XML document into the SOAP envelope body, adds headers, and sends it to another party. At the core of SOAP support in VisualWorks is the XML-to-object mapping mechanism. Using a SOAP binding, Smalltalk messages are marshaled into a SOAP/XML representation for communication to a service provider, and the XML response is unmarshaled back into Smalltalk.

VisualWorks also provides support for WSDL, explained in more detail in the next section. Support for WSDL simplifies SOAP messaging in VisualWorks by automatically producing the appropriate SOAP message, using transformations based on a WSDL schema. However, for situations where SOAP is needed without a WSDL schema, developers can still write directly to the SOAP protocols in VisualWorks.

The easiest way to use SOAP in VisualWorks is to request services for which there is a WSDL document describing those services. Because there is a SOAP binding for WSDL, VisualWorks can provide bindings to map a Smalltalk object, specifically instances of Message, to the appropriate SOAP messages. By specifying the mappings between a WSDL schema and Smalltalk in a transformation method or other mechanism, the task of constructing the appropriate SOAP messaging is automated in VisualWorks.

WSDL

The ability to process SOAP messages by any party requires a services description. Web Services Description Language (WSDL) is an XML-based language used to describe web services, defining the interface and mechanics of service interaction, so that other applications can access and interact with those web services. WSDL specifies what a request message must contain and what the response message will look like.

A WSDL document represents a web service as a collection of "endpoints," or ports, that receive and handle messages. The port description includes such details as the protocol, host and port number used, the operations that can be performed, the formats of the input and output messages, and the exceptions that can be raised.

The VisualWorks WSDL tool helps create a WSDL schema that covers all the details necessary to interact with the service, including message formats that detail the operations, transport protocols, and location. Currently, VisualWorks supports SOAP over HTTP in the WSDL binding.

VisualWorks WSDL support provides simple mechanisms for:

• Loading and parsing WSDL documents

• Generating mappings between XML (WSDL) elements and Smalltalk objects

• Creating classes from user-defined object types in a WSDL document

• Programmatically invoking a web service based on the port information in the document

UDDI

UDDI (Universal Description, Discovery and Integration) is a specification for information registries of web services. To make a service available, a provider identifies the API for that service, implements the service, and optionally publishes the service in a UDDI directory so that potential users can find it.

A UDDI directory entry is an XML file that describes a web service. The registries provide information such as the provider's name and contact information, industry codes, product classifications, URL and e-mail addresses, and details about interfaces and other properties of the services.

VisualWorks supports the UDDI version 1.0 API. The VisualWorks UDDI component enables an application to search existing directories and publish service descriptions in those directories.

UDDI "inquire" and "publish" API functions are exposed as SOAP messages over HTTP. HTTPS is used for publishing functions that require authentication. The implementation of UDDI support in VisualWorks is built on top of support for SOAP, HTTP and HTTPS, providing a very simple interface to the UDDI services. A registry is represented as a UDDIService instance. Inquiries and publishing requests are sent to it by sending Smalltalk messages that correspond to messages in the UDDI API. VisualWorks marshals the Smalltalk message as an appropriate SOAP message and sends it to the registry. Any response is marshaled back into Smalltalk objects for use by the application.

VisualWorks Web Services Components

The following is a list of all the web services components available in VisualWorks to support the XML, SOAP, WSDL and UDDI standards:

XMLObjectMarshalers

XMLObjectMarshalers is the basic XML-to-object marshaling machinery required to support all web services in VisualWorks. This functionality can also be used to develop protocols other than SOAP.

SOAP

The SOAP component installs basic web services support, specifically the SOAP bindings for XML-to-object marshaling.

WSDL

The WSDL component adds WSDL support to SOAP for identifying interfaces from WSDL documents.

WSDLTools

WSDLTools generates classes from a WSDL schema and a WSDL schema from classes.

UDDIInquire

UDDIInquire installs basic UDDI inquiry services, required for all other UDDI parcels.

UDDISearchTool

UDDISearchTool provides a GUI for the UDDIInquire functionality.

UDDIPublish

UDDIPublish adds publishing capability to UDDIInquire.

WSDL Wizard

The WSDL Wizard guides users through the process of exposing an existing Smalltalk application as a web service. Functions of the WSDL Wizard include:

- Describing the operations from a given service class
- Describing the Smalltalk classes mapping to XML types
- Creating a server to run the web services
- Creating a client to make requests
- Creating and publishing a WSDL schema with the web service description

Alternatively, given a WSDL schema, the WSDL Wizard can generate Smalltalk classes required to access the service the schema describes or to implement it. In this case, the WSDL Wizard can:

- Find a WSDL description using a UDDI registry
- Load a description from the internet
- Create an XML-to-object binding
- Create Smalltalk classes from the WSDL specification
- Invoke services described in a WSDL document

XML-to-Object Binding Wizard

The XML-to-Object Binding Wizard describes the Smalltalk classes mapping to XML types and tests the XML-to-Smalltalk object mapping.

VisualWorks Web Services Advantages

VisualWorks' web services capabilities provide several strong advantages to application developers.

Ease of Use

VisualWorks provides a range of capabilities that simplify, automate and expedite the deployment and use of web services. The VisualWorks code hides the complexity so that the developer does not have to be a web services expert to use the technology. The WSDL Wizard can automatically create the client and the code to make a request, while the only information the programmer will need is the WSDL schema location. With this high level of automation, a user with minimal knowledge about web services can easily integrate the technology into any application. Using VisualWorks' powerful yet easy-to-use suite of web services tools, it takes only a few minutes to convert an existing application into a running web service.

Flexibility

The VisualWorks web services framework is easy to extend and customize to meet the unique needs of any size application.

Cross-Platform Portability

Web services applications in VisualWorks benefit from the environment's portability across a wide range of platforms. Applications developed in VisualWorks can run on Windows, Linux, UNIX, Mac OS/X, and even mobile platforms like Windows CE – without requiring any modification to the code.

Client and Server Development

VisualWorks provides full support to develop web services client and server applications.

Compliance With Standards

VisualWorks offers full compliance with the latest web services standards, including XML 1.0, SOAP 1.1 and WSDL 1.1.

Interoperability

VisualWorks provides seamless interoperability with all the popular web services implementations, including .NET, Apache SOAP/Axis, BEA, Oracle and others. The implementation is compliant with WS-I Basic Profile 1.0a.

For example, VisualWorks is fully interoperable with .NET by executing against the Microsoft .NET servers. With .NET, the preferred method of interapplication communication is via web services. A web service implemented in VisualWorks and available on the network will be immediately visible to .NET developers, and will be indistinguishable from a native .NET service. Conversely, .NET services are easily discovered and accessed within VisualWorks.

The Perl SOAP:Lite package is useful for testing the interoperability of web services based in VisualWorks, and is freely available at http://www.soaplite.com. Interoperability with Java web services and clients can be tested using the tools available at http://ws.apache.org/axis/index.html.

HTTP Transport Support

HTTP 1.0/1.1 and HTTPS support includes GET and POST, cookies, compression, chunking, keep-alive, basic authentication, digest authentication, SSL/TLS encryption and certificate authentication, SSL session caching, proxies and proxy authentication.

SOA Platform

VisualWorks web services support SOA (service-oriented architecture) for the delivery of services via the web. The VisualWorks virtual SOA platform provides all of the necessary components of SOA, including the host environment, consumer environment, integration and assembly environment, development environment, publishing and discovery, service-level management, security infrastructure, monitoring and measurement, diagnostics and failure, web service protocols and certification.

World Headquarters • Cincinnati, OH USA • US 1-800-2CINCOM
Fax 1-513-612-2000 • International 1-513-612-2769
E-mail info@cincom.com • www.cincom.com • http://smalltalk.cincom.com