OGC®
OPEN GEOSPATIAL CONSORTIUM

Introduction to OGC Web Services

An OGC® White Paper

May 30, 2001

Editors:
Allan Doyle
Carl Reed

Contributors:
Jeff Harrison
Mark Reichardt

What are Web Services?[1]

During the last year, a new technology framework called "Web Services" has emerged as a viable model for Internet based applications. Simply, Web Services reflects the advantages of the Web as a platform for *services*, not just data.

By "services", we do not mean monolithic coarse-grained services like Amazon.com, but, rather, component services that can be plugged together to build larger, more comprehensive services. For example, Microsoft's *Passport* offers an authentication function exported on the Web. So hypothetically, an electronic newspaper like the Washington Post can avoid creating its own user authentication service, delegating it to Passport.

A formal definition of a web service may be borrowed from IBM's tutorial on the topic:

> *Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes...Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.*

[1] This section based on an article by Venu Vasudevan called "A Web Services Primer", April 2001.

It is obvious from this statement that the industry vision for Web Services is closely aligned with the OGC vision for interoperable and ubiquitous geo-services.

IBM's web services tutorial goes on to say that web services would have been too inefficient and costly to implement a few years ago. However, during the last two years IT trends and infrastructure investment have provided for cheaper bandwidth and storage as well as the availability of more dynamic content and the technology to support it. At the same time the pervasiveness and diversity of computing devices with different access platforms make the need for a middleware "glue" more important, while making the costs (bandwidth and storage) less objectionable.

Why bother with the Web Services when there are already a number of middleware platforms available to do this job (RMI, Jini, CORBA, DCOM etc.)? While middleware platforms provide great implementation vehicles for services, none of them is a clear winner. Why?

It's difficult, if not impossible, for large enterprises to standardize on a single middleware platform. Some enterprises require more than one because different departments have different requirements, others because mergers or acquisitions create a middleware mix. Even the lucky enterprise with a single middleware choice still has to use other technologies to interoperate with other enterprises and B2B markets.

The middleware environments that are most visible today are CORBA, Enterprise JavaBeans, message-oriented middleware, XML/SOAP, COM+ and .NET. However, over the past decade or so, the middleware landscape has continually shifted. For years, there was an assumption that a clear middleware winner would emerge and stabilize this state of flux[2]. However, each of these technologies has achieved success in the IT market. And, in spite of the advantages (sometimes real, sometimes imagined) of the latest middleware platform, migration is expensive and disruptive

The strengths of the Web as an information distributor, namely simplicity of access and ubiquity, are important in resolving the fragmented middleware world where interoperability is hard to come by. The Web complements these platforms by providing a uniform and widely accessible interface and access *glue* over services that are more efficiently *implemented* in a traditional middleware platform.

Viewed from a traditional n-tier Internet application architecture perspective, a web service is a veneer – or onion skin - for programmatic *access* to a service that is then *implemented* by other kinds of middleware. Access consists of service-neutral request handler (a listener) and a well-advertised interface that

[2] Synopsis of a discussion on middleware proliferation in: Model Driven Architecture, Richard Soley and the OMG Staff Strategy Group, November 2000

exposes the operations supported by the business logic. The logic itself is implemented in a traditional middleware platform.

What is the Power of a Web Service[3]?

Unlike complex and tightly bundled software packages, Web Service systems encourage significant decoupling and dynamic binding of components: When all components in a system are services, they encapsulate behavior and publish a messaging API to other collaborating components on the network. Services are organized by applications that use a service discovery to dynamically bind these services so they can collaborate. Thus, Web Services reflect a new service-oriented architectural approach, based on the notion of building applications by discovering and orchestrating network-available services – essentially *just-in-time integration of applications.*

With the Web Services approach, application design the act of describing entails describing the capabilities of available network services to perform a function and describing the orchestration of these collaborators. At runtime, application execution is a matter of transmitting the collaborator requirements to a discovery mechanism or service, locating a collaborator capable of providing the right service, and orchestrating the sending of messages to collaborators to invoke their services. Considered in total, this collection of services become an application, but can also be considered as a new aggregated service available for discovery and collaboration.

Web Services reflect a new service-oriented architectural approach, based on the notion of building applications by discovering and orchestrating network-available services

The Web Services architecture is the logical evolution of object-oriented analysis and design. It is also the logical evolution of the architecture, design, implementation, and deployment of e-business solutions. Both approaches have been proven in dealing with the complexity of large systems. As in object-oriented systems, some of the fundamental concepts in Web Services are encapsulation, message passing, dynamic binding, and service description and querying. Fundamental to Web Services, then, is the notion that everything is a service, publishing an API for use by other services on the network and encapsulating

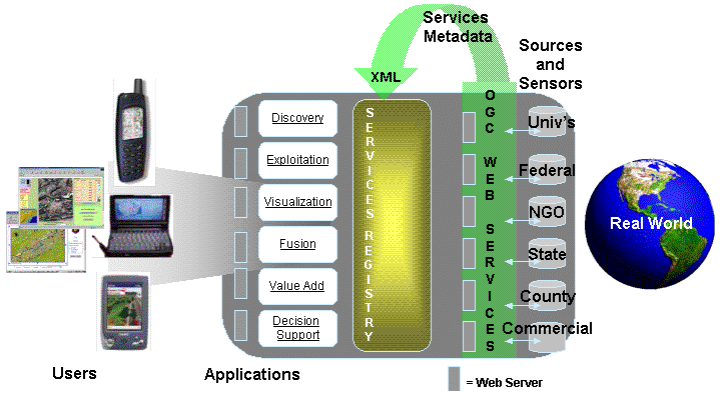
Fundamental to Web Services is the notion that everything is a service…for use by other services on the network…

[3] **Service:** A collection of operations, accessible through an interface, that allows a user to invoke a behavior of value to the user (definition from ISO 19119).

implementation details.

These same arguments define the requirement for an OGC Web Services framework. Further, the philosophy for Web Services is completely in line with current OGC thinking regarding the provision of interoperable Web Services.

What are OGC Web Services?



Within the broader context of Web Services, OGC Web Services (OWS) represent an evolutionary, standards-based framework that enable seamless integration of a variety of online geoprocessing and location services. OWS allows distributed geoprocessing systems to communicate with each other across the Web using familiar technologies such as XML and HTTP. OGC Web Services provide a vendor-neutral, interoperable framework for web-based discovery, access, integration, analysis, exploitation and visualization of multiple online geodata sources, sensor-derived information, and geoprocessing capabilities.

OWS can be a framework for building network-connected geoprocessing applications or for integrating geoprocessing capability into other information applications, such as Customer Relations Management or Enterprise Resource Planning. An analogy for this framework is a free market economy. Just as participants in a free market economy may be both vendors of products and services and consumers of other services and products, so providers of OWS may be both servers of geoprocessing capability and clients of such services. There is a constantly shifting network of agreements and transactions between consumers and vendors. In this manner, OWS is a web of geoprocessing services that can be connected in dynamic, open interoperable chains to create dynamic applications.

OWS will enable future applications to be assembled as desired from web accessible geoprocessing and location services. OWS will be self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. An OGC Web Service can thus be treated as a "black box"

OWS is a web of geoprocessing services that can be connected in dynamic, open interoperable chains to create dynamic applications

(Operations) that performs a task, such as providing driving directions. Since an OGC Web Service can describe the operation or operations they perform in Metadata (Capabilities), it is possible to search for services and understand what operations a given web accessible service can perform. These web services are addressable by a URL and are by definition network accessible. The OWS framework allows multiple services to be connected in sequence while at the same time allowing them to keep internal business logic independent – and if need be, proprietary. Figure 1 provides a general architectural schema for OWS. This schema identifies the generic classes of services that participate in various geoprocessing and location activities. Further, it identifies the properties that the services in those classes must have to connect them into useful applications.

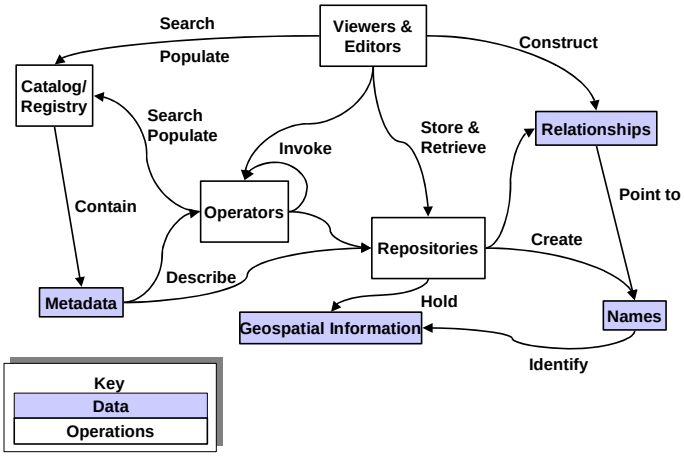


Figure 1. Some classes and properties of the OWS generic architecture

Geoprocessing services over distributed architectures

OGC originally focused on the use of Distributed Computing Platforms, or DCP's (specifically CORBA, OLE/COM, and SQL) as the basis for what were to become three separate implementations of a higher-level service model. The emergence of HTTP and the WWW and the subsequent formation of the Web Mapping Testbed subsequently thrust OGC into a new model of distributed geoprocessing. This new model is still taking shape and is forming the nucleus around which OGC Web Services will continue to grow.

Architectural considerations

The vision for OGC Web Services is influenced by many factors:

1) The need to support many independently developed implementations of a given service, such as geocoding;

2) The need to be able to chain services together to build an application;

3) The need to support many independently provided instantiations of the services;

4) The need to find specific instantiations of the services based on service type, service content, and service characteristics or quality factors;

5) The need to discover what services can be used with specific data holdings;

6) The need to deliver services that are tightly bound to data as well as services that have no direct data content

7) The need to enable access control, security, and e-commerce on the services;

8) A desire to enable ad-hoc chaining of services to satisfy aggregated workflow processes.

These factors define the power of what OGC Web Services for technology developers and users alike. They also describe some of the difficulties in finding the right balance when making architectural decisions. A primary focus for the next phase of the Web Mapping Testbed, WMT-3, will be to address these architectural decisions, which are categorized below for further discussion:

Service Model

The service model is the overall model governing the structure of OGC Web Services. It is an architecture in which individual services have interfaces of known types. These interface types are described in service metadata and the service metadata are available to clients of the service via a "Get Capabilities" request. There are Catalogs or Service Registries that provide queryable access to collections of service metadata. There are services provided by these Catalogs/Registries that assist in maintaining the information contained in the catalog. Finally, the interface types form an inheritance tree of interface properties. Other considerations for the service model are:

- There is a "Basic Service Model" that provides a minimum amount of these infrastructure elements so that we can begin gaining experience with this model;

- There is a "General Service Model" that provides a richer ability to describe and place into operation services that are chainable and services that are loosely coupled to data sources as opposed to being tightly bound to data sources;

- There are wider IT industry efforts underway (cf. UDDI, WSDL, NASSL, SOAP - see below) that provide service models and architectures that we want to take advantage of and be compatible with;

- OGC has adopted ISO 19119 as the basis for an Abstract Specification of this service model.

Service Metadata

Service metadata is that information that describes a given service. There is service type metadata and there is service instance metadata. For example, a portrayal service will have different service type metadata than a feature service. A feature service could be instantiated on Asian data and another, identical feature service could be instantiated on data content for Antarctica; these services would have the same service type metadata but would have different service instance metadata.

Service metadata should take into account existing metadata standards, in particular those of ISO TC 211, OGC, and those of the particular service model we may choose.

Capabilities Request/Response

A given service must provide access to its service type and service instance metadata. Historically within OGC, this is known as the "Capabilities" request and response.

Catalog/Registry Request/Response

The OGC Catalog Services Interface Specification is the initial source of information regarding construction of a service registry. It describes the formulation of requests or queries into the holdings of a catalog and describes a mechanism for receiving a response.

Specific request and response syntax and semantics are under development to satisfy the needs of our service model. The work has primarily concerned itself with the content of the response, following the pattern developed in the FGDC Geo profile of Z39.50.

Catalog/Registry Services

An OGC discussion paper on Web Registry Services following material developed during the IP2000 initiative describes the operations of a registry service. This includes the ability to make stateless queries.

Further services that support maintaining the coherency and currency of the holdings of a catalog are important additions to the OGC Catalog Services model.

Service Types & Interface Inheritance

Using the service metadata and the capabilities response mechanism, we have to develop a taxonomy of service types and develop an inheritance tree of those service types.

OGC Web Services in Relation to broader efforts (UDDI/SOAP/WSDL)

Many large, well-known companies are developing and promulgating web services visions of their own. Some of this is a matter of market positioning and attempt to capitalize on what is possibly a major new buzzword (soon, no doubt, someone will declare 2001 to be the "year of web services"). Under all the marketing messages lie some very strong efforts at developing a web services infrastructure[4]. Most prominent among these are SOAP (Simple Object Adaptor Protocol), UDDI (Universal Description and Discovery Interface), and WSDL (Web Services Description Language). These efforts, while somewhat overlapping, form a layering of interface and service description and query mechanisms that should not be ignored as OGC engages in WMT-3.

SOAP – Simple Object Adaptor Protocol

SOAP is a protocol specification that defines a uniform way of passing XML-encoded data. In also defines a way to perform remote procedure calls (RPCs) using HTTP as the underlying communication protocol.

SOAP arose from the realization that no matter what the current middleware offerings are, they need a Wide Area Network wrapper. Architecturally, sending messages as plain XML has advantages in terms of ensuring interoperability. The middleware players seem willing to put up with the costs of parsing and serializing XML in order to scale their approach to wider networks.

Submitted in 2000 to the W3C as a Note by IBM, Microsoft, UserLand, and DevelopMentor, the further development of SOAP is now in the care of the W3C's XML Protocols Working Group. This effectively means that SOAP is frozen and stable until such time as the W3C Working Group delivers a specification.

UDDI (Universal Description, Discovery and Integration Service)

UDDI Universal description discovery integration (UDDI)

Universal description discovery integration (UDDI) is a collection of technologies that work together to advertise a web service across an intranet, an extranet, or the Internet. It was designed through a collaborative effort of over 200 companies that create and manage web services technology. Several companies including Microsoft, IBM, Hewlett Packard, Oracle, and Sun

[4] IBM, BEA Systems, Oracle, Microsoft and HP are just a few of the major IT companies that have developed or are developing a Web Services infrastructure strategy and capability.

Microsystems made contributions. UDDI is managed by the Organization for the Advancement of Structured Information Standards (OASIS) and is currently in open standard version 2.0, but the version 3.0 specifications were released as of July 2002.

UDDI allows a potential user or customer to find a web-based service that they require and retrieve pertinent business and technical information to create an interface to use it.

Why is this important? In the past, finding a business with suitable web services to solve a specific problem or provide a business-to-business (B2B) service was like finding a needle in a haystack. Moreover, once a service was found, gathering the necessary information to create an interface so the service could be accessed was time consuming and unnecessarily complicated. UDDI was created to provide an accessible system where a company could post all the relevant information (either publicly or privately) about their web services to speed the development time for building B2B service interfaces. Currently, UDDI is simply a repository contained in a public or private database that helps facilitate the technical end of accessing a web service.

UDDI is based on XML and SOAP since each is independent of any computing platform software or hardware and each are able to communicate between disparate systems. As the name universal description discovery integration implies, this system is comprised of three distinct areas:

Description

Discovery

Integration

Description

The heart of UDDI is the business registry, also known as the UDDI business registry (UBR). The UBR is an XML-based database containing detailed and structured entries. It was designed to mimic a telephone directory and is broken down into white, yellow, and green pages. Each database entry has information that describes the following about a company and the service it will provide:

White pages: Company information such as phone numbers, emails, addresses.

Yellow pages: Information to identify the specific industry of the business.

Green pages: Service-specific technical details.

Discovery

Although there are a large number of businesses that provide a variety of services, it is still difficult for potential clients to find the specific services they can use to enhance their business. The discovery portion of UDDI is loosely

analogous to an Internet search engine in that the UBR can be searched to a certain extent for services and businesses in specific industries. Businesses publish their information in the UBR and potential customers and business partners access the database to find the services they need. It should be noted that in version 2 of UDDI the search capabilities are not as robust as in 3.0.

Integration

Once a suitable web service has been identified, the final step is to integrate the customer and the web service so they can communicate. The green pages of the UBR contain technical details about an advertised service that can be used by a potential customer to create an interface to access the service. This streamlines the development process and minimizes the time it takes to implement a web service solution.

UDDI is unique in that it does not gather information from an external source. All entries are intentionally input by individual companies and the information is complied into a single resource, making the information more accurate, up-to-date, and pertinent for a potential client searching for a specific service.

As UDDI develops and becomes more searchable and more secure, it will begin to shape the future of e-commerce. UDDI gives web services an accessible and competitive venue to be bought and sold like any other product in today's electronic marketplace.

WSDL (Web Services Definition Language)

WSDL provides a way for service providers to describe the basic format of web service requests over different protocols or encodings. WSDL is used to describe *what* a web service can do, *where* it resides, and *how* to invoke it. While the claim of SOAP/HTTP independence is made in various specifications, WSDL makes the most sense if it assumes SOAP/HTTP/MIME as the remote object invocation mechanism. UDDI registries describe numerous aspects of web services, including the binding details of the service. WSDL fits into the subset of a UDDI service description.

WSDL defines services as collections of network endpoints or *ports*. In WSDL the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions of messages, which are abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding; a collection of ports define a service