# An overview of the Web Services Inspection Language

Distributed Web service discovery using WS-Inspection documents

Peter Brittenham (peterbr@us.ibm.com)
Web Services Architect, IBM Corporation
November 2001

Service discovery defines a process for locating service providers and retrieving service description documents, and is a key component of the overall Web services model. Service discovery is a very broad concept, which means that it is unlikely to have one solution that addresses all of its requirements. The Universal Description, Discovery and Integration (UDDI) specification addresses a subset of the overall requirements by using a centralized service discovery model. This article will provide an overview of the Web Services Inspection Language (WS-Inspection), which is another related service discovery mechanism, but addresses a different subset of requirements using a distributed usage model. The WS-Inspection specification is designed around an XML-based model for building an aggregation of references to existing Web service descriptions, which are exposed using standard Web server technology.

The Web services architecture is based upon the interactions between three primary roles: service provider, service registry, and service requestor. These roles interact using publish, find, and bind operations. The service provider is the business that provides access to the Web service and publishes the service description in a service registry. The service requestor finds the service description in a service registry and uses the information in the description to bind to a service. A logical view of the Web services architecture is shown in Figure 1. In this view of the Web services architecture, the service registry provides a centralized location for storing service descriptions. A UDDI registry is an example of this type of service registry.

**Figure 1: Web services architecture**

**Contents:**

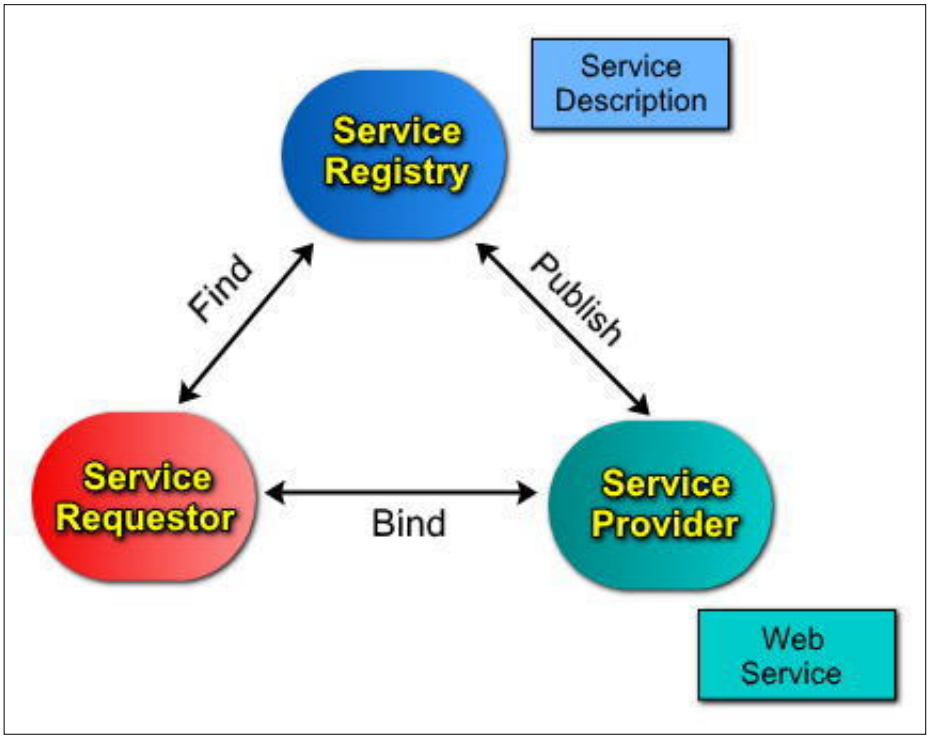


**Related content:**

Web Services Inspection Language (WS-Inspection) specification
The WS-Inspection and UDDI Relationship

**Also in the Web services zone:**

Tutorials
Tools and products
Articles




Although it is important, the centralized service registry is not the only model for Web service discovery. The simplest form of service discovery is to request a copy of the service description from the service provider. After receiving the request, the service provider can simply e-mail the service description as an attachment or provide it to the service requestor on a transferable media, such as a diskette. Although this type of service discovery is simple, it is not very efficient since it requires prior knowledge of the Web service, as well as the contact information for the service provider.

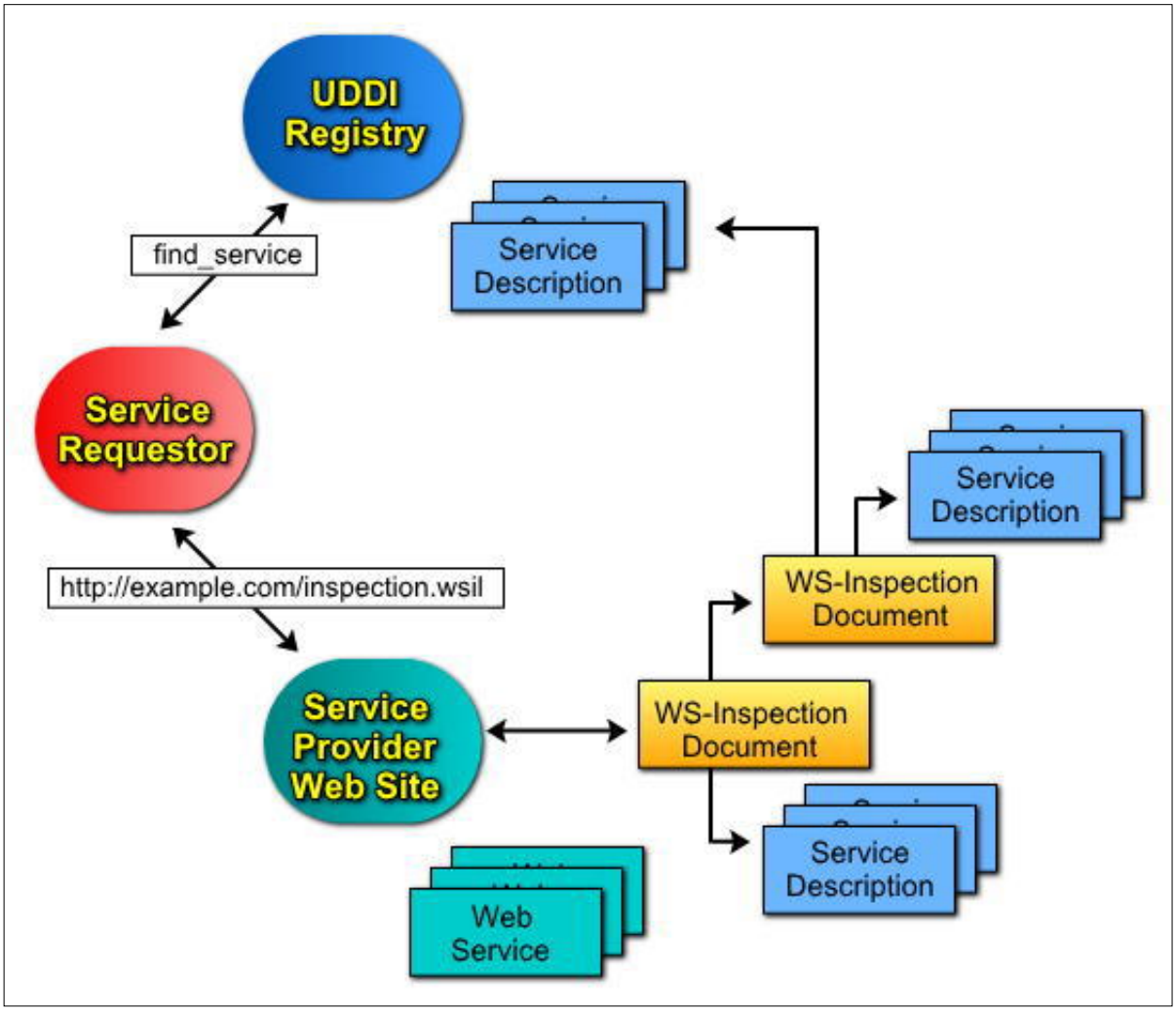
Between these two extremes, there is a need for a distributed service discovery method that provides references to service descriptions at the service provider's point-of-offering. The Web Services Inspection Language provides this type of distributed discovery method, by specifying how to inspect a web site for available Web services. The WS-Inspection specification defines the locations on a Web site where you could look for Web service descriptions.

Since the Web Services Inspection Language focuses on distributed service discovery, the WS-Inspection specification complements UDDI by facilitating the discovery of services available on Web sites, but which may not be listed yet in a UDDI registry. Additional information on the relationship between the Web Services Inspection Language and UDDI can be found in *The WS-Inspection and UDDI Relationship* (see Resources).

WS-Inspection overview

The WS-Inspection specification (see Resources) does not define a service description language. WS-Inspection documents provide a method for aggregating different types of service descriptions. Within a WS-Inspection document, a single service can have more than one reference to a service description. For example, a single Web service might be described using both a WSDL file and within a UDDI registry. References to these two service descriptions should be put into a WS-Inspection document. If multiple references are available, it is beneficial to put all of them in the WS-Inspection document so that the document consumer can select the type of service description that they are capable of understanding and want to use. Figure 2 provides an overview of how WS-Inspection documents are used.

**Figure 2: WS-Inspection overview**




The WS-Inspection specification contains two primary functions, which are discussed in more detail in the next two sections.

- It defines an XML format for listing references to existing service descriptions.
- It defines a set of conventions so that it is easy to locate WS-Inspection documents.

WS-Inspection document format

A WS-Inspection document provides an aggregation of references to service descriptions. These service descriptions can be defined in any service description format, such as WSDL, UDDI, or plain HTML. As mentioned previously, a WS-Inspection document is generally made available at the point-of-offering for the services that are referenced within the document.

A WS-Inspection document can contain a list of references to service descriptions, as well as references to other WS-Inspection documents. A WS-Inspection document will contain one or more `<service>` and `<link>` elements. A `<service>` element will contain one or more references to different types of service descriptions for the same Web service. The `<link>` element may contain references to only one type of service description, but these service descriptions do not have to reference the same Web service.

Listing 1 contains a simple example of a WS-Inspection document. This example contains two references to different service descriptions, and a single reference to another WS-Inspection document. The first `<service>` element contains only one service description, and it is a reference to a WSDL document. The second `<service>` element also contains only one service description reference. This reference is to a business service entry in a UDDI registry. The UDDI service key identifies one unique business service. The UDDI service reference also contains extensibility elements which are discussed in the next section. The `<link>` element is used to reference a collection of service descriptions. In this case, it is referencing another WS-Inspection document.

**Listing 1: Example WS-Inspection document**

```
<?xml version="1.0"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
  <service>
    <description referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
                 location="http://example.com/exampleservice.wsdl" />
  </service>
  <service>
    <description referencedNamespace="urn:uddi-org:api">
       <wsiluddi:serviceDescription location=
                "http://example.com/uddi/inquiryapi">
         <wsiluddi:serviceKey>
                52946BB0-BC28-11D5-A432-0004AC49CC1E</wsiluddi:serviceKey>
       </wsiluddi:serviceDescription>
    </description>
  </service>
  <link referencedNamespace=
                "http://schemas.xmlsoap.org/ws/2001/10/inspection/"
        location="http://example.com/tools/toolservices.wsil"/>
</inspection>
```

WS-Inspection document extensibility

The WS-Inspection specification does not limit the type of service descriptions that can be referenced. Both the `<description>` and `<link>` element may contain extensibility elements that represent information for a specific service description technology. The WS-Inspection specification defines a set of standard extensibility elements for both WSDL and UDDI. Since the `<description>` element is used to reference a single service description and the `<link>` element is used to reference one or more sets of service descriptions, any extensibility elements that are defined for these elements should follow this same pattern.

WSDL service descriptions can only be referenced from within a `<description>` element. The WSDL extensibility elements can be used to indicate whether or not the WSDL document contains an endpoint specification. If there is more than one service element in the WSDL document, then the `<wsilwsdl:referencedService>` element should be used to indicate which one is associated with the entry in the WS-Inspection document. One or more `<wsilwsdl:implementedBinding>` elements may appear in WSDL service description reference. Each of these elements references a binding that is implemented by the WSDL document. Listing 2 contains an example of a WS-Inspection document that contains all of the WSDL extensibility elements.

**Listing 2: WS-Inspection Document with WSDL Extensibility Elements**

```
<?xml version="1.0"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
  ...
  <service>
    <name xml:lang="en-US">StockQuoteService</name>
    <description referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
      <wsilwsdl:reference endpointPresent="true">

location="http://localhost:8080/webservices/wsdl/stockquote/sqs.wsdl">
        <wsilwsdl:referencedService
            xmlns:tns="http://www.getquote.com/StockQuoteService">
          tns:StockQuoteService
        </wsilwsdl:referencedService>
        <wsilwsdl:implementedBinding
            xmlns:interface="http://www.getquote.com/StockQuoteService-interface">
          interface:StockQuoteServiceBinding
        </wsilwsdl:implementedBinding>
      </wsilwsdl:reference>
    </description>
  </service>
```

```
  ...
</inspection>
```

The UDDI extensibility elements may appear within either the `<link>` or `<description>` elements. The elements used within the `<link>` element can only reference a UDDI business entity. Since this element references a UDDI business entity, resolving this reference will result in one or more service descriptions. The elements used within the `<description>` element may only reference a single UDDI business service. Listing 3 contains an example of the UDDI bindings for a WS-Inspection document.

The `<wsiluddi:businessDescription>` element is used within a `<link>` element to specify a reference to a UDDI business entity. The businessService element may contain either a discoveryURL, or a businessKey, or both. If a businessKey is specified, then the location attribute on the businessDescription element must contain an inquiry URL for a UDDI registry. This URL is used to send a `get_businessDetail` message to the UDDI registry using the businessKey that was specified.

The `<wsiluddi:serviceDescription>` element can only be used within the `<description>` element, and can reference only one service description. Within the serviceDescription element, a discoveryURL, a serviceKey, or both can be specified. The location attribute on the serviceDescription must contain the inquiry URL for a UDDI registry, when the serviceKey is specified.

For both the businessDescription and serviceDescription elements, if both the discoveryURL and the businessKey or serviceKey are specified, then the person who is processing the WS-Inspection document can select which one they want to use. The discoveryURL will always return a UDDI business entity. So when it is used with the serviceDescription element, the serviceKey must be used to locate the individual service description within the business entity.

**Listing 3: WS-Inspection document with UDDI extensibility elements**

```
<?xml version="1.0"?>
<inspection targetNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/"
    xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
  <link referencedNamespace="urn:uddi-org:api">
    <wsiluddi:businessDescription location=
                "http://www.getquote.com/uddi/inquiryapi">
      <wsiluddi:businessKey>3BF0ACC0-BC28-11D5-A432-0004AC49CC1E<
                /wsiluddi:businessKey>
      <wsiluddi:discoveryURL useType="businessEntity">
        http://www.getquote.com/uddi?businessKey=
                3BF0ACC0-BC28-11D5-A432-0004AC49CC1E
      </wsiluddi:discoveryURL>
    </wsiluddi:businessDescription>
  </link>
  <service>
    <name>UDDI Service Description</name>
    <description referencedNamespace="urn:uddi-org:api">
      <wsiluddi:serviceDescription location=
                "http://www.getquote.com/uddi/inquiryapi">
        <wsiluddi:serviceKey>52946BB0-BC28-11D5-A432-0004AC49CC1E<
                /wsiluddi:serviceKey>
        <wsiluddi:discoveryURL useType="businessEntity">
          http://www.getquotecom/uddi?businessKey=
                3BF0ACC0-BC28-11D5-A432-0004AC49CC1E
        </wsiluddi:discoveryURL>
      </wsiluddi:serviceDescription>
    </description>
  </service>
</inspection>
```

Linking to WS-Inspection documents
One important feature of the WS-Inspection specification, is the ability to link a WS-Inspection document to one or more

different WS-Inspection documents. This feature can be used to manage service description references by grouping them into different documents. Using the `<link>` element, a hierarchy of WS-Inspection documents can be built using these individual documents. For example, separate WS-Inspection documents can be created for different categories of services, and one primary WS-Inspection document can link all of them together.

Finding WS-Inspection documents

The second primary function provided by the WS-Inspection specification is how to define the locations where you can access WS-Inspection documents. There are two conventions which were created to make the location and retrieval of WS-Inspection documents easy:

- Fixed name WS-Inspection documents
- Linked WS-Inspection documents

The fixed name for WS-Inspection documents is *inspection.wsil*. A document with this name can be placed at common entry points for a Web site. For example, if the common entry point is http://example.com or http://example.com/services, then the location of the WS-Inspection document would be http://example.com/inspection.wsil or http://example.com/services/inspection.wsil, respectively.

References to WS-Inspection documents may also appear within different content documents, such as HTML pages. When putting entries in an HTMLpage, a META tag may be used to convey the location of a WS-Inspection document. Listing 4 contains an example of an HTML page that contains the same WS-Inspection document references listed above. The HTML page that contains these references should be widely used. This could be the root document for a Web server, or it could be a Web page that describes, in a human readable format, one or more Web services that appear in the WS-Inspection document.

**Listing 4: WS-Inspection document references in an HTML page**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <META name="serviceInspection" content=
                "http://example.com/inspection.wsil"
    <META name="serviceInspection"
content="http://example.com/services/inspection.wsil"
  <head>
...
<html>
```

Web Services Toolkit support for the WS-Inspection specification

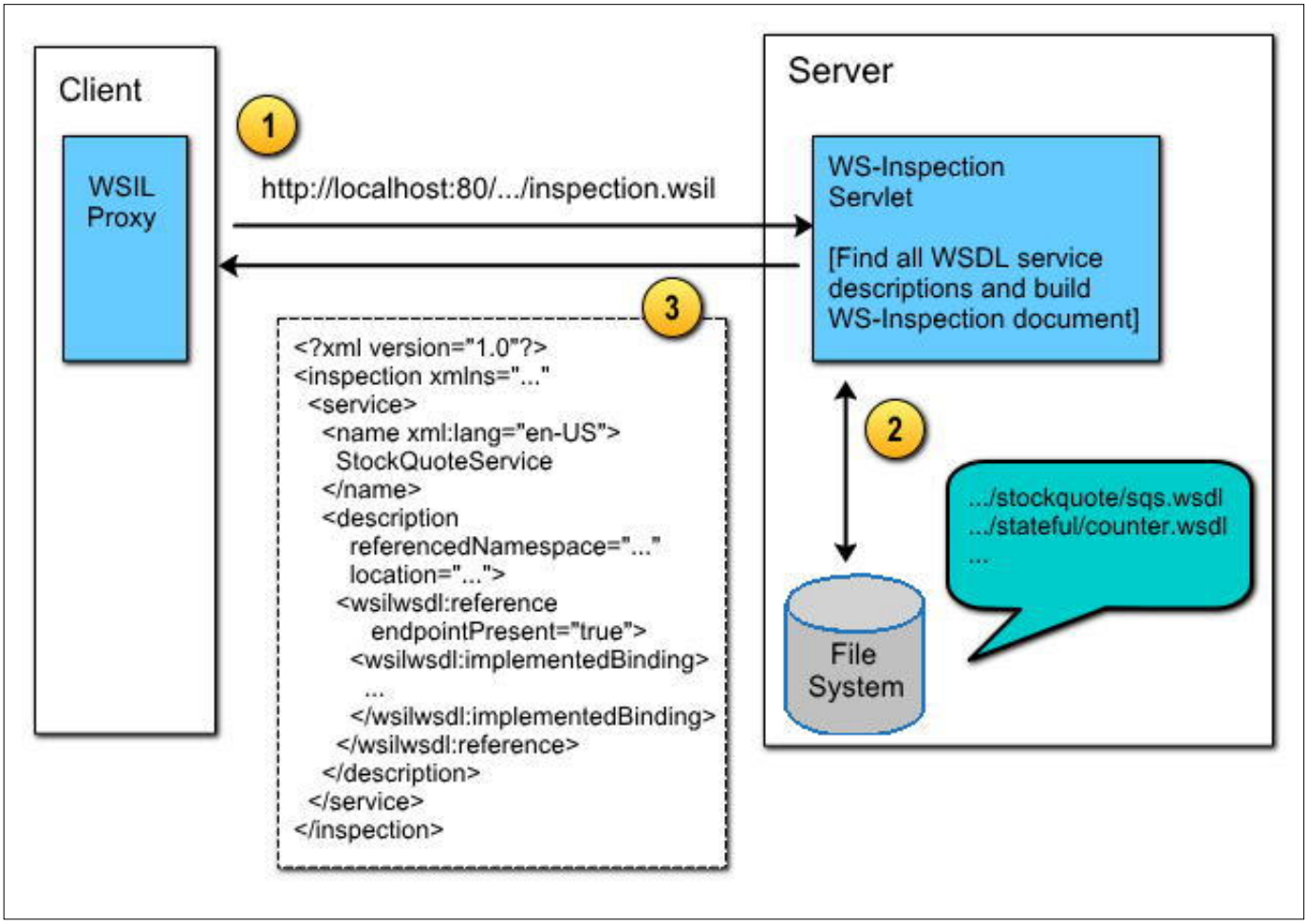
The Web Services Toolkit V2.4.1 (see Resources) includes integrated support for the Web Services Inspection Language. This support includes a demonstration of how to use WS-Inspection documents, and a Java API that allows you to parse existing WS-Inspection documents and programmatically create new documents.

Most of the toolkit demos provide an option to use WS-Inspection technology or UDDI as the service discovery mechanism. If the WS-Inspection option is used, the demos request a WS-Inspection document from the Web server configured using the toolkit configuration utility. This request is submitted using the fixed name for the WS-Inspection document. This document name is set up to invoke a Java servlet. This servlet will dynamically create the WS-Inspection document, by searching for WSDL service description documents within the toolkit directory structure.

Figure 3 contains an overview of this process:

1. The WS-Inspection document proxy is used to request the contents of the WS-Inspection document using a fixed name.
2. The URL that is used to retrieve the WS-Inspection document maps to a servlet. This servlet will search through the local filesystem for all WSDL service descriptions. A reference to each service description will be put into the WS-Inspection document.
3. The dynamically generated WS-Inspection document is returned to the client.

**Figure 3: WS-Inspection document support in the Web Services Toolkit**




Listing 5 contains a portion of the WS-Inspection document that is returned by the WS-Inspection servlet. The service name is set from the name attribute on the definition element within the WSDL document. The entry that appears in this listing is for the stock quote demo.

**Listing 5: WS-Inspection document generated by the Web Services Toolkit**

```
<?xml version="1.0"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
  ...
  <service>
    <name xml:lang="en-US">StockQuoteService</name>
    <description referencedNamespace="http://schemas.xmlsoap.org/wsdl/"

location="http://localhost:8080/webservices/wsdl/stockquote/sqs.wsdl">
      <wsilwsdl:reference endpointPresent="true">
        <wsilwsdl:implementedBinding
            xmlns:interface="http://www.getquote.com/StockQuoteService-interface">
          interface:StockQuoteServiceBinding
        </wsilwsdl:implementedBinding>
      </wsilwsdl:reference>
    </description>
  </service>
  ...
</inspection>
```

Using the Web Services Inspection Language for Java API

The Web Services Inspection Language for Java API (WSIL4J) provides a Java interface, which can be used to parse existing WS-Inspection documents or programmatically create new WS-Inspection documents. Most of the WSIL4J classes represent the elements that can appear in a WS-Inspection document. For example, the `<inspection>` element is represented by the

*Inspection* class, and the `<service>` element is represented by the *Service* class. There are also utility classes that make it easy to read and parse a WS-Inspection document, as well as write out the contents of the WSIL4J objects as an XML document.

Listing 6 contains an example of how to use this API. In this sample code, a WS-Inspection document is read and the service elements are searched for references to WSDL service descriptions. When a WSDL service description is found, its location is saved in a list which is displayed on the console. You can view and download the complete WSInspectionExample application (see Resources). If you have installed the toolkit, you can use the *wstkenv* command to set up the classpath that is needed to compile and run these examples. This command is located in the WSTK bin directory. The purpose of this command is to define a set of environment variables. One of the environment variables is named WSTK_CP. This environment variable contains the classpath that is required to compile and run the examples.

**Listing 6: Example WS-Inspection application**

```
...
// Create a new instance of a WS-Inspection document
WSILDocument document = WSILDocument.newInstance();
// Read and parse the WS-Inspection document
document.read(wsinspectionURL);
// Get the inspection element from the document
Inspection inspection = document.getInspection();
// Obtain a list of all service elements
Service[] services = inspection.getServices();
// Display purpose of list
System.out.println("Display list of WSDL service description references...");
// Process each service element to find all WSDL document references
for (int serviceCount = 0; serviceCount < services.length; serviceCount++)
{
  // Get the next set of description elements
  descriptions = services[serviceCount].getDescriptions();
  // Process each description to find the WSDL references
  for (int descCount = 0; descCount < descriptions.length; descCount++)
  {
    // If the referenced namespace is for WSDL, then save the location reference
    if
(descriptions[descCount].getReferencedNamespace().equals(WSDLConstants.NS_URI_WSDL))
    {
      // Add WSDL location to the list
      wsdlList.add(descriptions[descCount].getLocation());
    }
  }
  // If this service has WSDL service descriptions, then display the list
  if (wsdlList.size() > 0)
  {
    // Get service name
    serviceName = (services[serviceCount].getServiceNames().length == 0) ?
    "[no service name]" : services[serviceCount].getServiceNames()[0].getText();
    // Display service name
    System.out.println("  Service: " + serviceName);
    // Display list
    Iterator iterator = wsdlList.iterator();
    for (int count = 1; iterator.hasNext(); count++)
    {
      System.out.println("    [" + count + "] " + ((String) iterator.next()));
    }
  }
  // Clear the list
  wsdlList.clear();
```

```
}
...
```

Using the WS-Inspection proxy

The WSIL4J API also provides a *WSILProxy* class which can be used to easily access certain types of information within a WS-Inspection document. The proxy interface will read the WS-Inspection document, and then allow you to directly access the WSDL documents for UDDI business services that you need. Listing 7 contains a portion of an application that shows how to use the WS-Inspection proxy to get a list of WSDL documents for a given service name. You can also view and download the complete WSInspectionProxyExample application (see Resources).

**Listing 7: Example WS-Inspection proxy usage**

```
...
// Create a new instance of a WS-Inspection document proxy
WSILProxy proxy = new WSILProxy(wsinspectionURL);
// Get all of the WSDL documents using the input service name
WSDLDocument[] wsdlDocuments =
                proxy.getWSDLDocumentByServiceName(serviceName);
// Display purpose of list
System.out.println("Display contents of WSDL service
                description documents for service name [" + serviceName + "]...");
// Process each WSDL document reference
for (int wsdlCount = 0; wsdlCount < wsdlDocuments.length; wsdlCount++)
{
  // Display contents of the document
  System.out.println("[" + wsdlCount + "]
" + wsdlDocuments[wsdlCount].serializeToXML());
}
...
```

Summary

In this article, I described the Web Services Inspection Language and how it provides a simple, distributed service discovery method for any type of Web service description document. I also described how the WS-Inspection technology is complementary to existing service discovery methods, such as UDDI, because it defines a process for inspecting a Web site for service descriptions.

In the future, developers will start to see additional uses for the WS-Inspection technology. For example, you may see the implementation of WS-Inspection interfaces for Web service description repositories. You may also see this technology used to develop Web service crawlers. These service crawlers would search through Web sites for WS-Inspection documents and then aggregate the service description references from multiple sites. With this and other applications of this technology, the Web Services Inspection Language will become an important part of the overall Web services usage model.

Resources

- Participate in the discussion forum on this article by clicking **Discuss** at the top or bottom of the article.
- The Web Services Inspection Language (WS-Inspection) specification is located on the developerWorks site.
- The WS-Inspection and UDDI Relationship paper contains information on how WS-Inspection relates to UDDI.
- Download the IBM Web Services Toolkit V2.4.1 which contains an implementation of the WS-Inspection specification.
- You can view and download the complete WSInspectionExample and WSInspectionProxyExample from developerWorks.
- The UDDI specifications can be accessed on the uddi.org web site.
- The WSDL V1.1 specification contains information on the structure of WSDL documents.

About the author

Peter Brittenham is currently the lead architect for the IBM Web Services Toolkit. The Web Services Toolkit contains the tools and runtime support that are required to build Web services using SOAP and WSDL, as well as the runtime support to publish and find service definitions using both WS-Inspection documents or a UDDI registry. He can be contacted at peterbr@us.ibm.com.