Web Services Invocation Framework

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#mw-head)[Jump to search](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#searchInput)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | hide**This article has multiple issues.** Please help [**improve it**](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit) or discuss these issues on the [**talk page**](https://en.wikipedia.org/wiki/Talk:Web_Services_Invocation_Framework). *(*[*Learn how and when to remove these template messages*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)*   |  | | --- | | This article **may be too technical for most readers to understand**. Please [help improve it](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit) to [make it understandable to non-experts](https://en.wikipedia.org/wiki/Wikipedia:Make_technical_articles_understandable), without removing the technical details. *(September 2014)* |  |  | | --- | | This article **uses**[**abbreviations**](https://en.wikipedia.org/wiki/Abbreviation)**that may be**[**confusing or ambiguous**](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style#Acronyms_and_abbreviations)**.** *(March 2012)* |  |  | | --- | | This article includes a [list of references](https://en.wikipedia.org/wiki/Wikipedia:Citing_sources), related reading or [external links](https://en.wikipedia.org/wiki/Wikipedia:External_links), **but its sources remain unclear because it lacks**[**inline citations**](https://en.wikipedia.org/wiki/Wikipedia:Citing_sources#Inline_citations). *(November 2014)* |  |  | | --- | | This article **contains content that is written like**[**an advertisement**](https://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not#Wikipedia_is_not_a_soapbox_or_means_of_promotion). *(December 2014)* | |

|  |  |
| --- | --- |
| **Apache WSIF** | |
| [**Developer(s)**](https://en.wikipedia.org/wiki/Software_developer) | [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation) |
| [**Stable release**](https://en.wikipedia.org/wiki/Software_release_life_cycle) | 2.0 / January 27, 2003 |
| [**Operating system**](https://en.wikipedia.org/wiki/Operating_system) | [Cross-platform](https://en.wikipedia.org/wiki/Cross-platform) |
| [**Type**](https://en.wikipedia.org/wiki/Software_categories#Categorization_approaches) | [Web services](https://en.wikipedia.org/wiki/Web_services) |
| [**License**](https://en.wikipedia.org/wiki/Software_license) | [Apache License](https://en.wikipedia.org/wiki/Apache_License) 2.0 |
| **Website** | [ws.apache.org/wsif](http://ws.apache.org/wsif) |

The **Web Services Invocation Framework** (WSIF) supports a simple and flexible [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) API for invoking any [Web Services Description Language](https://en.wikipedia.org/wiki/Web_Services_Description_Language) (WSDL)-described service.

Using WSIF, WSDL can become the centerpiece of an integration framework for accessing software running on diverse platforms which uses different protocols. The software needs to be described using WSDL and have a binding included in its description[[*clarification needed*](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify)] that the client's WSIF framework has a provider for. WSIF defines and comes packaged with providers for local Java, [Enterprise JavaBeans](https://en.wikipedia.org/wiki/Enterprise_JavaBeans) (EJB), Java Message Service (JMS), and [Java EE Connector Architecture](https://en.wikipedia.org/wiki/Java_EE_Connector_Architecture) (JCA) protocols, which means that a client can define an EJB or a [Java Message Service](https://en.wikipedia.org/wiki/Java_Message_Service)-accessible service directly as a WSDL binding and access it transparently using WSIF, using the same API one would use for a [SOAP service](https://en.wikipedia.org/wiki/SOAP) or a local Java class.



**Contents**

* [1Structure](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#Structure)
* [2Bindings for EJBs, JMS, and JCA](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#Bindings_for_EJBs,_JMS,_and_JCA)
* [3Description](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#Description)
* [4Differences between WSIF and Axis](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#Differences_between_WSIF_and_Axis)
* [5Differences between WSIF and JAX-RPC](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#Differences_between_WSIF_and_JAX-RPC)
* [6See also](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#See_also)
* [7External links](https://en.wikipedia.org/wiki/Web_Services_Invocation_Framework#External_links)

Structure[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=1)]

In WSDL, a binding defines how to map between the abstract *PortType* and a real service format and protocol. For example, the SOAP binding defines the encoding style, the *SOAPAction* header, the namespace of the body (the targetURI), and so forth.

WSDL allows multiple implementations for a Web service and multiple ports that share the same PortType. In other words, WSDL allows the same interface to have bindings to services like SOAP and [IIOP](https://en.wikipedia.org/wiki/General_Inter-ORB_Protocol).

WSIF provides an API to allow the same client code to access any available binding. As the client code can be written to the PortType, it can be a deployment or configuration setting (or a code choice) which port and binding it uses.

WSIF uses *providers* to support these multiple WSDL bindings. A provider is a piece of code that supports a WSDL extension and allows invocation of the service through that particular implementation. WSIF providers use the J2SE JAR service provider specification, making them discoverable at [runtime](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)).

Clients can utilize new implementations and delegate the choice of port to the infrastructure and runtime, which allows the implementation to be chosen on the basis of quality of service characteristics or business policy.

Bindings for EJBs, JMS, and JCA[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=2)]

WSIF defines additional binding extensions so that [Enterprise JavaBean](https://en.wikipedia.org/wiki/Enterprise_JavaBean) (EJBs), local Java classes, software accessible over [message queues](https://en.wikipedia.org/wiki/Message_queue) using the [Java Message Service](https://en.wikipedia.org/wiki/Java_Message_Service) (JMS) API, and software that can be invoked using the [Java Connector architecture](https://en.wikipedia.org/wiki/Java_EE_Connector_Architecture) can also be described in WSDL. WSIF is packaged with providers that allow transparent invocation given the corresponding WSDL description.

Description[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=3)]

WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol (SOAP) APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed.

WSIF allows stubless or completely dynamic invocation of a Web service, based upon examination of the metadata about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime and the calling service to defer choosing a binding until runtime.

It is closely based on WSDL so it can invoke any service that can be described in the language.

If a complicated enterprise software system consists of various pieces of software, developed over a period of decades—EJBs, legacy apps accessed using Java's connector architecture, SOAP services hosted on external servers, old code accessed through messaging middleware—it is necessary to write software applications that use all these pieces to do useful things, where the differences in protocols, mobility of software, etc. conflict with one another.

If the software is moved to a different server, the code breaks. The SOAP libraries used change—for example, when one moves from using Apache SOAP to Apache Axis—as it uses a now-deprecated SOAP API. Something that was previously accessible as an EJB is now available through messaging middleware via JMS—again, the code that uses the software must be fixed, or if one has an EJB which is offered as a SOAP service to external clients. Using SOAP results in a performance penalty compared to accessing the EJB directly.

WSIF fixes these problems by allowing WSDL to be used as a normalized description of disparate software, and allows users to access this software without depending on a protocol or location. The separation of the API from the actual protocol also means there is flexibility—protocols, location, etc. can be switched without having to recompile client code. If an externally available SOAP service becomes available as an EJB, users can use RMI/IIOP by changing the service description (the WSDL), without making any modification in applications that use the service. WSDL's extensibility, its capability to offer multiple bindings for the same service, deciding on a binding at runtime, etc. can be exploited.

Differences between WSIF and Axis[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=4)]

Axis is an implementation of SOAP. It includes on the server-side infrastructure for deploying web service implementations and then routing SOAP messages between clients and those implementations. It also implements the JAX-RPC specification for invoking SOAP services.

WSIF is similar to the client piece of Axis, in that it is used for invoking services. However, WSIF's API is WSDL-driven and protocol independent; it allows protocol-specific code ("providers") to be plugged in. For invoking SOAP services, WSIF is packaged with an Axis provider, that uses Axis APIs (i.e. JAX-RPC) to do the invocation. So WSIF operates at a more abstract level than Axis.

Differences between WSIF and JAX-RPC[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=5)]

JAX-RPC is an API for invoking XML-based RPC services – essentially its current scope is limited to invocation of SOAP services. WSIF is an API for invoking WSDL-described services, whether they happen to be SOAP services or not (for example, WSIF defines WSDL bindings so that EJBs, enterprise software accessible using JMS or the Java Connector architecture as well as local Java classes can all be described as first-class WSDL services and then invoked using the same, protocol-independent [WSIF](https://en.wikipedia.org/wiki/WSIF) API).

See also[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=6)]

* [Apache XML](https://en.wikipedia.org/wiki/Apache_XML)

External links[[edit](https://en.wikipedia.org/w/index.php?title=Web_Services_Invocation_Framework&action=edit&section=7)]

* [Web Services Invocation Framework documentation](http://ws.apache.org/wsif/)
* [WSIF and WSDL](http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cwsf_wsdl.html)
* [Service Binding](http://www.s-cube-network.eu/km/terms/s/service-binding)