Artix 5.6.4

Command Reference: Java

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

http://www.microfocus.com

2017-02-23

Contents

Preface

What is Covered in This Book

This book is a reference to the command line tools included with Artix for Java.

Who Should Read This Book

This book is intended for developers who use command line tools as part of their build and development environments. However, all users of Artix for Java can benefit from using this as a reference.

The Artix Documentation Library

For information on the organization of the Artix Java library, the document conventions used, and where to find additional resources, see ***Using the Artix Library: Java***.

Further Information and Product Support

Additional technical information or advice is available from several sources.

The product support pages contain a considerable amount of additional information, such as:

- The WebSync service, where you can download fixes and documentation updates.
- The Knowledge Base, a large collection of product tips and workarounds.
- Examples and Utilities, including demos and additional product documentation.

Note:
Some information may be available only to customers who have maintenance agreements.

If you obtained this product directly from Micro Focus, contact us as described on the Micro Focus Web site, http://www.microfocus.com. If you obtained the product from another source, such as an authorized distributor, contact them for help first. If they are unable to help, contact us.

Information We Need

However you contact us, please try to include the information below, if you have it. The more information you can give, the better Micro Focus SupportLine can help you. But if you don't know all the answers, or you think some are irrelevant to your problem, please give whatever information you have.

- The name and version number of all products that you think might be causing a problem.
- Your computer make and model.
- Your operating system version number and details of any networking software you are using.
- The amount of memory in your computer.
- The relevant page reference or section in the documentation.
- Your serial number. To find out these numbers, look in the subject line and body of your Electronic Product Delivery Notice email that you received from Micro Focus.

Contact information

Our Web site gives up-to-date details of contact numbers and addresses.

Additional technical information or advice is available from several sources.

The product support pages contain considerable additional information, including the WebSync service, where you can download fixes and documentation updates. To connect, enter http://www.microfocus.com in your browser to go to the Micro Focus home page.

If you are a Micro Focus SupportLine customer, please see your SupportLine Handbook for contact information. You can download it from our Web site or order it in printed form from your sales representative. Support from Micro Focus may be available only to customers who have maintenance agreements.

You may want to check these URLs in particular:

- http://www.microfocus.com/products/corba/artix.aspx (trial software download and Micro Focus Community files)
- https://supportline.microfocus.com/productdoc.aspx (documentation updates and PDFs)

To subscribe to Micro Focus electronic newsletters, use the online form at:

http://www.microfocus.com/Resources/Newsletters/infocus/newsletter-subscription.asp

Prerequisites

Artix Java Runtime provides a tool for setting up your environment.

To set up your environment to use Artix Java Runtime do the following:

1. Run the **artix_java_env** script located in `*InstallDir*/bin`.
2. Ensure that `JAVA_HOME` points to a Java 6 (or higher) JDK.

Generating WSDL

Artix provides a number of command line tools for generating WSDL.

This chapter describes the following tools:

- java2ws
- idl2wsdl
- xsd2wsdl

java2ws

Generates WSDL and other artifacts from JAX-WS compliant Java code.

Synopsis

`java2ws` [[-?] | [-help] | [-h]] [-frontend { jaxws | simple }] [-databinding { jaxb | aegis }] [-wsdl] [-wrapperbean] [-client] [-server] [-ant] [-o *`outFile`*] [-s *`sourceDir`*] [-d *`resourceDir`*] [-classdir *`classDir`*] [-cp *`classpath`*] [-soap12] [-t *`targetNamespace`*] [-beans *`beanPath`*...] [-servicename *`serviceName`*] [-portname *`portName`*] [-createxsdimports] [-v] [[-verbose] | [-quiet]] *`classname`*

Description

java2ws takes a service endpoint implementation (SEI) and generates the support files used to implement a Web service. **java2ws** can generate the following:

- a WSDL document
- the server code needed to deploy the service as a POJO
- client code for accessing the service
- wrapper and fault beans

Arguments

The arguments used to manage the code generation process are reviewed in the following table.

Option	**Interpretation**
`-?`	Displays the online help for this utility.
`-help`	

Option	Interpretation
`-h`	
`-frontend {jaxws|simple}`	Specifies front end to use for processing the SEI and generating the support classes. `jaxws` is the default.
`-databinding {jaxb|aegis}`	Specifies the data binding used for processing the SEI and generating the support classes. The default when using the JAX-WS front end is `jaxb`. The default when using the simple frontend is `aegis`.
`-wsdl`	Instructs the tool to generate a WSDL document.
`-wrapperbean`	Instructs the tool to generate the wrapper bean and the fault beans.
`-client`	Instructs the tool to generate client code.
`-server`	Instructs the tool to generate server code.
`-ant`	Instructs the tool to generate an Ant build script to compile the generated code.
`-o` *`outFile`*	Specifies the name of the generated WSDL file.
`-s` *`sourceDir`*	Specifies the directory into which the generated source files are placed.
`-d` *`resourceDir`*	Specifies the directory into which the resource files are placed.
`-classdir` *`classDir`*	Specifies the directory into which the generated source files are compiled. If this option is not used, the generated source is not compiled.
`-cp` *`classpath`*	Specifies the classpath searched when processing the SEI.
`-soap12`	Specifies that the generated WSDL document is to include a SOAP 1.2 binding.
`-t` *`targetNamespace`*	Specifies the target namespace to use in the generated WSDL file.
`-beans` *`beanPath`*	Specifies the path used to locate the bean definition files.
`-servicename` *`serviceName`*	Specifies the value of the generated `service` element's `name` attribute.
`-portname` *`portName`*	Specify the value of the generated `port` element's `name` attribute.

Option	Interpretation
`-createxsdimports`	Instructs the tool to generate a separate schema for the types instead of including the types directly in the generated WSDL document.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`classname`	Specifies the name of the SEI class.

Using Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.java2ws.JavaToWS` class. Example 1 shows the **java** task to generate WSDL from an SEI.

Example 1. Generating WSDL From Ant

```
<java classname="org.apache.cxf.tools.java2ws.JavaToWS" fork="true">
  <arg value="-wsdl"/>
  <arg value="Service.greeter"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

idl2wsdl

Generates an Artix Java Runtime compliant WSDL document from a CORBA IDL file.

Synopsis

`idl2wsdl` [-I *`idl-include-dir`*...] [-o *`output-dir`*] [-a *`corba-address`*] [-b] [-f *`corba-address-file`*] [-n *`schema-import-file`*] [-s *`idl-sequence-type`*] [-w *`target-namespace`*] [-x *`schema-namespace`*] [-t *`corba-typemap-namespace`*] [-L *`logical-wsdl-filename`*] [-P *`physical-wsdl-filename`*] [-T *`schema-filename`*] [-qualified] [-e *`xml-encoding-type`*] [-mns*`namespaceMapping`*] [-ow *`wsdloutput-file`*] [ex*`excludedModules`*] [-pf] [-v] [[-verbose] | [-quiet]] *`idl`*

Description

idl2wsdl supports several options that control the generation of a WSDL file from an IDL file. The default behavior of the tool is to create WSDL file that uses wrapped doc/literal style messages. Wrapped doc/literal style messages have a single part, defined using an element, that wraps all of the elements in the message.

Required Arguments

The command has the following required arguments:

Option	Interpretation
idl	Specifies the name of the IDL file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
`-I` *idl-include-dir*	Specify a directory to be included in the search path for the IDL preprocessor. You can use this flag multiple times.
`-o` *output-dir*	Specifies the directory into which the WSDL file is written.
`-a` *corba-address*	Specifies an absolute address through which the object reference may be accessed. The address may be a relative or absolute path to a file, or a corbaname URL.
`-b`	Specifies that bounded strings are to be treated as unbounded. This eliminates the generation of the special types for the bounded string.
`-f` *corba-address-file*	Specifies a file containing a string representation of an object reference. The object reference is placed in the `corba:address` element in the port definition of the generated service. The file must exist when you run the IDL compiler.
`-n` *schema-import-file*	Specifies that a schema file is to be included in the generated contract by an import statement. This option cannot be used with the `-T` option.
`-s` *idl-sequence-type*	Specifies the XML Schema type used to map the IDL sequence<octet> type. Valid values are `base64Binary` and `hexBinary`. The default is `base64Binary`.
`-w` *target-namespace*	Specifies the namespace to use for the WSDL document's target namespace.
`-x` *schema-namespace*	Specifies the namespace to use for the generated XML Schema's target namespace.
`-t` *corba-typemap-namespace*	Specifies the namespace to use for the CORBA type map's target namespace.

Option	Interpretation
-L *logical-wsdl-filename*	Specifies that the logical portion of the generated WSDL specification into is written to *logical-wsdl-filename*. The logical WSDL is then imported into the default generated file.
-P *physical-wsdl-filename*	Specifies that the physical portion of the generated WSDL specification into is written to *physical-wsdl-filename*. The physical WSDL is then imported into the default generated file.
-T *schema-filename*	Specifies that the schema types are to be generated into a separate file. The schema file is included in the generated contract using an import statement. This option cannot be used with the -n option.
-qualified	Generates fully qualified WSDL.
-e *xml-encoding-type*	Specifies the value for the generated WSDL document's xml encoding attribute. The default is UTF-8.
-mns*namespaceMapping*	Specifies a mapping between IDL modules and XML namespaces.
-ow *wsdloutput-file*	Specifies the name of the generated WSDL file.
-ex*excludeModules*	Specifies one or more IDL modules to exclude when generating the WSDL file.
-pf	Specifies that polymorphic factory support is enabled.
-h	Displays the tool's usage statement.
-v	Displays the version number for the tool.
-verbose	Displays comments during the code generation process.
-quiet	Suppresses comments during the code generation process.

xsd2wsdl

Generates a WSDL document containing the types defined in an XML Schema document.

Synopsis

xsd2wsdl [[-?] | [-help] | [-h]] [-t *target-namespace*] [-n *wsdl-name*] [-d *output-directory*] [-o *output-file*] [-v] [[-verbose] | [-quiet]] {*xsdurl*}

Description

xsd2wsdl imports an XML Schema document and generates a WSDL file containing a `types` element populated by the types defined in the XML Schema document.

Required Arguments

The command has the following required arguments:

Option	Interpretation
`-t` *`target-namespace`*	Specifies the target namespace for the generated WSDL.
`xsdurl`	The path and name of the existing XSDSchema file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-n` *`wsdl-name`*	Specifies the value of the generated `definition` element's `name` attribute.
`-d` *`output-directory`*	Specifies the directory in which the generated WSDL is placed.
`-o` *`output-file`*	Specifies the name of the generated WSDL file.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.

Using Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.misc.XSDToWSDL` class.

Example 2 shows the **java** task to execute this command.

Example 2. Generating a WSDL from a Schema Using Ant

```
<java classname="org.apache.cxf.tools.misc.XSDToWSDL" fork="true">
  <arg value="-t"/>
  <arg value="http://cxf.apache.org/demos"/>
  ...
  <arg value="MyXSD.xsd"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

Adding Bindings

Artix provides command line tools for adding SOAP, XML, and CORBA bindings to WSDL documents.

This chapter describes the following tools:

- wsdl2soap
- wsdl2xml
- wsdl2idl -corba

wsdl2soap

Generates a WSDL document containing a valid SOAP/HTTP endpoint definition based on a `portType` element.

Synopsis

`wsdl2soap` [[-?] | [-help] | [-h]] {-i *port-type-name*} [-b *binding-name*] [-soap12] [-d *output-directory*] [-o *output-file*] [-n *soap-body-namespace*] [-style { document | rpc }] [-use (literal/encoded)] [-v] [[-verbose] | [-quiet]] *wsdlurl*

Description

wsdl2soap will generate a new WSDL file with a SOAP binding from an existing WSDL file containing a `portType` element.

Required Arguments

The command has the following required arguments:

Option	Interpretation
`-i` *port-type-name*	Specifies the `portType` element for which a binding should be generated.
wsdlurl	The path and name of the WSDL file containing the `portType` element definition.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-b` *`binding-name`*	Specifies the name of the generated SOAP binding.
`-soap12`	Specifies that the generated binding will use SOAP 1.2.
`-d` *`output-directory`*	Specifies the directory to place generated WSDL file.
`-o` *`output-file`*	Specifies the name of the generated WSDL file.
`-n` *`soap-body-namespace`*	Specifies the SOAP body namespace when the style is RPC.
`-style` (`document`/`rpc`)	Specifies the encoding style (document or RPC) to use in the SOAP binding. The default is `document`.
`-use` (`literal`/`encoded`)	Specifies the binding use (encoded or literal) to use in the SOAP binding. The default is `literal`.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.

If the `-style rpc` argument is specified, the `-n` *`soap-body-namspace`* argument is also required. All other arguments are optional and may be listed in any order.

Using Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.misc.WSDLToSoap` class. Example 3 shows the **java** task to generate a SOAP 1.2 binding.

Example 3. Generating a SOAP 1.2 Binding From Ant

```
<java classname="org.apache.cxf.tools.misc.WSDLToSoap" fork="true">
  <arg value="-i"/>
  <arg value="greeter"/>
  <arg value="-soap12"/>
  ...
  <arg value="MyWSDL.wsdl"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

wsdl2xml

Generates a WSDL document containing an XML binding based on a `portType` element.

Synopsis

`wsdl2xml` [[-?] | [-help] | [-h]] [-i *port-type-name*] [-b *binding-name*] [-e *service-name*] [-p *port-name*] [-a *address*] [-d *output-directory*] [-o *output-file*] [-v] [[-verbose] | [-quiet]] {*wsdlurl*}

Description

wsdl2xml generates an XML binding from an existing WSDL document containing a `portType` element.

Arguments

The arguments used to manage WSDL file generation are reviewed in the following table.

Option	Interpretation
`-i` *port-type-name*	Specifies the `portType` element to use.
wsdlurl	The path and name of the existing WSDL file.

Optional Arguments

The command takes the following optional arguments:

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-b` *binding-name*	Specifies the name of the generated XML binding.
`-e` *service-name*	Specifies the value of the generated `service` element's `name` attribute.
`-p` *port-name*	Specifies the value of the generated `port` element's `name` attribute. To specify multiple `port` elements, separate the names by a space.
`-a` *address*	Specifies the value used in the `address` element of the generated `port` element.

Option	Interpretation
`-d` *output-directory*	Specifies the directory to place generated WSDL file.
`-o` *output-file*	Specifies the name of the generated WSDL file.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.

Using Ant

To execute this tool using Ant set the **java** task's classname property to `org.apache.cxf.tools.misc.WSDLToXML`.

Example 4 shows the **java** task to execute this command.

Example 4. Generating a SOAP Binding From Ant

```
<java classname="org.apache.cxf.tools.misc.WSDLToXML" fork="true">
  <arg value="-i"/>
  <arg value="greeter"/>
  ...
  <arg value="MyWSDL.wsdl"/>
  <classpath>
    <path refid="artix_java.classpath"/>
  </classpath>
</java>
```

wsdl2idl –corba

Adds an Artix Java Runtime CORBA binding to a WSDL document.

Synopsis

`wsld2idl` {-corba} {-i *portType*} [-idl] [-b *binding*] [-d *dir*] [-w *wsdlOut*] [-o *idlOut*...] [-props *namespace*] [-wrapped] [-a *address*] [-f *address-file*] [[-quiet] | [-verbose]] [-v] [-h] *wsdl*

Description

wsdl2idl -corba adds a Artix Java Runtime CORBA binding to an existing WSDL document. The generated WSDL file will also contain an Artix Java Runtime CORBA port with no address specified.

NOTE: You can also generate an IDL file that corresponds to the generated CORBA binding by using the `-idl` option.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
`-corba`	Specifies that the tool will generate a new WSDL file with a CORBA binding.
`-i` *`portType`*	Specifies the name of the interface for which the CORBA binding is generated.
`wsdl`	Specifies the WSDL document to which the binding is added.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
`-idl`	Specifies that an IDL file will be generated for the generated CORBA binding. You must also use the `-b` flag in conjunction with this flag.
`-b` *`binding`*	Specifies the name of the generated CORBA binding.
`-d` *`dir`*	Specifies the directory into which the new WSDL document is written.
`-w` *`wsdlOut`*	Specifies the name of the WSDL document containing the generated CORBA binding.
`-o` *`idlOut`*	Specifies the name of the generated IDL file.
`-props` *`namespace`*	Specifies the namespace to use for the generated CORBA typemap.
`-wrapped`	Specifies that the generated binding uses wrapped types.
`-a` *`address`*	Specifies the value of the generated binding's `corba:address` element's `location` attribute.
`-f` *`address-file`*	Specifies the name of a file whose contents are to be used as the value of the generated binding's `corba:address` element's `location` attribute.
`-v`	Displays the tool's version.
`-h`	Specifies that the tool will display a detailed usage statement.
`-quiet`	Specifies that the tool is to run in quiet mode.
`-versbose`	Specifies that the tool is to run in verbose mode.

Adding Endpoints

Artix provides command line tools for adding endpoints to WSDL documents.

This chapter describes the following tools:

- wsdl2service -transport http
- wsdl2service -transport jms

wsdl2service -transport http

Generates a WSDL document containing a valid HTTP endpoint definition from a `binding` element.

Synopsis

`wsdl2service - transport http` [[-?] | [-help] | [-h]] [-e *service-name*] [-p *port-name*] { -n *binding-name*} [-a *address*] [-soap12] [-o *output-file*] [-d *output-directory*] [-v] [[-verbose] | [-quiet]] { *wsdlurl*}

Description

wsdl2service -transport http creates a new WSDL file containing an HTTP service definition from an existing WSDL document containing a binding element.

Arguments

The arguments used to manage the WSDL file generation are reviewed in the following table.

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-e` *service-name*	Specifies the value of the generated `service` element's `name` attribute.
`-p` *port-name*	Specifies the value of the generated `port` element's `name` attribute. To specify multiple port elements, separate the names by a space.
`-a` *address*	Specifies the value used in the `address` element of the port.

Option	Interpretation
`-soap12`	Specifies that the SOAP version to use is 1.2.
`-n` *`binding-name`*	Specifies the binding used to generate the service.
`-o` *`output-file`*	Specifies the name of the generated WSDL file.
`-d` *`output-directory`*	Specifies the directory in which the generated WSDL is placed.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`wsdlurl`	The path and name of the existing WSDL file.

Using Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.misc.WSDLToService` class. Example 5 shows the **java** task to generate a HTTP binding.

Example 5. Generating a JMS Binding From Ant

```
<java classname="org.apache.cxf.tools.misc.WSDLToService" fork="true">
  <arg value="-transport"/>
  <arg value="http"/>
  <arg value="-n"/>
  <arg value="JMSSoapBinding"/>
  ...
  <arg value="MyWSDL.wsdl"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

wsdl2service -transport jms

Generates a WSDL document containing a valid JMS endpoint definition from a `binding` element.

Synopsis

`wsdl2service -transport jms` [[-?] | [-help] | [-h]] [-e *`service-name`*] [-p *`port-name`*] { -n *`binding-name`*} [[-jds (queue/topic)] | [-jpu *`jndi-provider-URL`*] | [-jcf *`initial-context-factory`*] | [-jfn *`jndi-connection-factory-name`*] | [-jdn *`jndi-destination-name`*] | [-jmt { text | binary }] | [-jmc { true | false }] | [-jsn *`durable-subscriber-name`*]] [-o *`output-file`*] [-d *`output-directory`*] [-v] [[-verbose] | [-quiet]] { *`wsdlurl`* }

Description

wsdl2service creates a new WSDL file containing an HTTP or JMS service definition from an existing WSDL document containing a binding element.

Arguments

The arguments used to manage the WSDL file generation are reviewed in the following table.

Option	**Interpretation**
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-e` *`service-name`*	Specifies the value of the generated `service` element's `name` attribute.
`-p` *`port-name`*	Specifies the value of the generated `port` element's `name` attribute. To specify multiple port elements, separate the names by a space.
`-n` *`binding-name`*	Specifies the binding used to generate the service.
`-jds {queue/topic}`	Specifies the JMS destination style.
`-jpu` *`jndi-provider-URL`*	Specifies the URL of the JMS JNDI provider.
`-jcf` *`initial-context-factory`*	Specifies the JMS initial context factory.
`-jfn` *`jndi-connection-factory-name`*	Specifies the JMS JNDI connection factory name.
`-jdn` *`jndi-destination-name`*	Specifies the JMS JNDI destination name.
`-jmt (text/binary)`	Specifies the JMS message type.
`-jmc (true/false)`	Specifies if the `MessageID` is used as the `CorrelationID`.
`-jsn` *`durable-subscriber-name`*	Specifies an optional durable subscriber name.
`-o` *`output-file`*	Specifies the name of the generated WSDL file.
`-d` *`output-directory`*	Specifies the directory in which the generated WSDL is placed.
`-v`	Displays the version number for the tool.

Option	Interpretation
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`wsdlurl`	The path and name of the existing WSDL file.

Using Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.misc.WSDLToService` class. Example 6 shows the **java** task to generate a JMS binding.

Example 6. Generating a JMS Binding From Ant

```
<java classname="org.apache.cxf.tools.misc.WSDLToService" fork="true">
  <arg value="-transport"/>
  <arg value="jms"/>
  <arg value="-n"/>
  <arg value="JMSSoapBinding"/>
  ...
  <arg value="MyWSDL.wsdl"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

Validating WSDL

Artix can validate your contracts to see if they are well-formed WSDL documents. In addition, Artix can validate your contract against the WS-I Basic Profile.

This chapter describes the following tool:

- wsdlvalidator

wsdlvalidator

Validates a WSDL document.

Synopsis

`wsdlvalidator` [[-?] | [-help] | [-h]] [-s *`schema-url`*...] [-v] [[-verbose] | [-quiet]] {*`wsdlurl`*}

Description

wsdlvalidator validates whether a WSDL document is well-formed and conforms to the WSDL schema.

Arguments

The arguments used to validate WSDL file are reviewed in the following table:

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-s` *`schema-url`*	Specifies the URL of a user specific schema to be included in the validation of the contract. This switch can appear multiple times.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the validation.
`-quiet`	Suppresses comments during the validation.
`wsdlurl`	The path and name of the existing WSDL file

Using Ant

To execute this tool using Ant set the **java** task's classname property to `org.apache.cxf.tools.validator.WSDLValidator`.

Generating Code from WSDL

Artix for Java provides a number of command line tools for generating application code from WSDL documents.

This chapter describes the following tools:

- wsdlgen
- wsdl2java
- java2js
- wsdl2js

wsdlgen

Generates application code based on JavaScript templates.

Synopsis

`wsdlgen` [-G *`ApplicationType`*] [-T *`TemplateID`*...] [-C *`configFile`*] [-D *`name`*=*`value`*...] *`WSDLFile`*

Description

wsdlgen is a customizable code generator. Using JavaScript templates, you can customize the implementation classes generated from a WSDL document. The tool includes a number of standard templates that generate basic Java code if you do not require any customization.

For more information see the ***Artix WSDLGen Guide, Java***.

Arguments

The arguments used to manage the code generation are reviewed in the following table.

Option	Interpretation
`-G` *`ApplicationType`*	Specifies the type of application to generate. The following application type is defined by default: `jaxws`—for generating JAX-WS code
`-T` *`TemplateID`*	Specifies the template ID that governs code generation. See Template IDs for details.

Option	Interpretation
`-C` *`ConfigFile`*	Specifies the location of a configuration file to be used by the code generator.
`-D` *`name`*`=`*`value`*	Specifies the value, *`value`*, of a JavaScript property, *`name`*. Typically you will use this option to specify a value for the portType property. This instructs the code generator the WSDL `portType` element for which code is to be generated.
`WSDLFile`	Specifies the URL of the WSDL document.

Template IDs

When called with `-G` *`ApplicationType`* the `-T` *`TemplateID`* switch supports the following template IDs:

Option	Interpretation
`impl`	Generate the stub and skeleton code require to implement the interface defined by the specified WSDL `portType` element.
`server`	Generate a simple `main()` for a standalone service that will host an implementation of the interface defined by the specified WSDL `portType` element. Stub code is also generated.
`client`	Generate a Java class that invokes all of the operations defined by the specified WSDL `portType` element. Stub code is also generated.
`all`	For JAX-WS, generate a client and a server.
`ant`	Generate an Apache Ant build file for a Java application.

wsdl2java

Generates JAX-WS compliant Java code from a WSDL document.

Synopsis

```
wsdl2java -fe|-frontend <front-end-name> -db|-databinding <data-
binding-name> -wv <wsdl-version> -p <[wsdl-namespace =]package-
name>* -sn <service-name> -b <binding-file-name>* -reserveClass
<class-name>* -catalog <catalog-file-name> -d <output-directory>
-compile -classdir <compile-classes-directory> -impl –server
-client -clientjar <jar-file-name> -all -autoNameResolution
-allowElementReferences|-aer<=true> -defaultValues<=class-name-
for-DefaultValueProvider> -ant -nexclude <schema-namespace
[= java-package-name]>* -exsh <(true, false)> -noTypes -dns
<(true, false> -dex <(true, false)> -validate -keep -wsdlLocation
<wsdlLocation> -xjc<xjc-arguments>*
-asyncMethods<[=method1,method2,...]>*
-bareMethods<[= method1,method2,...]>*
-mimeMethods<[= method1,method2,...]>* -noAddressBinding
-faultSerialVersionUID <fault-serialVersionUID> -exceptionSuper
<exceptionSuper> -mark-generated -h|-?|-help -version|-v
-verbose|-V -quiet|-q|-Q -wsdlList <wsdlurl>
```

Description

wsdl2java takes a WSDL document and generates fully annotated Java code from which to implement a service. The WSDL document must have a valid portType element, but it does not need to contain a binding element or a service element. Using the optional arguments you can customize the generated code. In addition, **wsdl2java** can generate an Ant-based makefile to build your application.

Arguments

The arguments used to manage the code generation process are reviewed in the following table.

Option	**Interpretation**
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-fe` *`front-end-name`*	Specifies the front end used by the code generator. The default is `jaxws`. Currently, Artix only provides the JAX-WS front end for the code generator.

Option	**Interpretation**
`-db` *`databinding-name`*	Specifies the data binding used by the code generator. The default is `jaxb`. Currently, Artix only provides the JAXB data binding for the code generator.
`-wv` *`wsdl-version`*	Specifies the WSDL version expected by the tool. The default is `1.1`. Currently, Artix only provides WSDL 1.1 support for the code generator.
`-p` [*`wsdlNamespace`*=]*`PackageName`*	Specifies zero, or more, package names to use for the generated code. Optionally specifies the WSDL namespace to package name mapping.
`-b` *`binding-name`*	Specifies zero, or more, JAXWS or JAXB binding files. Use multiple -b flags to specify multiple entries.
`-sn` *`service-name`*	Specifies the name of the WSDL service for which code is to be generated. The default is to generate code for every service in the WSDL document.
`-d` *`output-directory`*	Specifies the directory into which the generated code files are written.
`-catalog` *`catalog-file-name`*	Specifies a catalogfile to use for mapping the imported WSDL or schema.
`-compile`	Compiles generated Java files.
`-classdir` *`complile-class-dir`*	Specifies the directory into which the compiled class files are written.
`-client`	Generates starting point code for a client mainline.
`-server`	Generates starting point code for a server mainline.
`-impl`	Generates starting point code for an implementation object.
`-all`	Generates all starting point code: types, service proxy, service interface, server mainline, client mainline, implementation object, and an Ant `build.xml` file.
`-ant`	Generates the Ant `build.xml` file.

Option	**Interpretation**
`-autoNameResolution`	Automatically resolve naming conflicts without requiring the use of binding customizations
`-defaultValues` [=*`DefaultValueProvider`* `impl`]	Instructs the tool to generate default values for the generated client and the generated implementation. Optionally, you can also supply the name of the class used to generate the default values. By default, the `RandomValueProvider` class is used.
`-nexclude` *`schema-namespace`* [=*`java-packagename`*]	Ignore the specified WSDL schema namespace when generating code. This option may be specified multiple times. Also, optionally specifies the Java package name used by types described in the excluded namespace(s).
`-exsh` (`true`/`false`)	Enables or disables processing of implicit SOAP headers (that is, SOAP headers defined in the `wsdl:binding` but not `wsdl:portType` section.) Processing the SOAP headers requires the SOAP binding jars available on the classpath. Default is `false`.
`-dns` (`true`/`false`)	Enables or disables the loading of the default namespace package name mapping. Default is `true`, and http://www.w3.org/2005/08/addressing=org.apache.cxf.ws.addressing namespace package mapping will be enabled.
`-dex` (`true`/`false`)	Enables or disables the loading of the default excludes namespace mapping. Default is `true`.
`-validate`	Instructs the tool to validate the WSDL document before attempting to generate any code.
`-keep`	Instructs the code generataor to not overwrite any existing files.
`-wsdlLocation` *`wsdlLocation`*	Specifies the value of the `@WebService` annotation's wsdlLocation property.
`-xjc`<*`xjc_args`*>	Specifies a comma separated list of arguments to be passed to directly to the XJC when the JAXB data binding is being used. To get a list of all possible XJC arguments use the `-xjc-X`.

Option	**Interpretation**
`-noAddressBinding`	Instructs the tool to use the Artix ESB proprietary WS-Addressing type instead of the JAX-WS 2.1 compliant mapping.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`-exceptionSuper`	A superclass for fault beans generated from `wsdl:fault` elements. Defaults to `java.lang.Exception`.
`-reserveClass classname`	Used with `-autoNameResolution`, defines a class name that **wsdl-to-java** must not use when generating classes. Use this option multiple times for multiple classes.
`-allowElementReferences<=true>` `-aer<=true>`	If `true`, disregards the rule given in section 2.3.1.2(v) of the JAX-WS 2.2 specification disallowing element references when using wrapper-style mapping.
`-asyncMethods=`*`foo,bar,...`*	List of subsequently-generated Java class methods to allow for client-side asynchronous calls, similar to `enableAsyncMapping` in a JAX-WS binding file.
`-bareMethods=`*`foo,bar,...`*	List of subsequently-generated Java class methods to allow wrapper style, similar to `enableWrapperStyle` in a JAX-WS binding file.
`-mimeMethods=`*`foo,bar,...`*	List of subsequently-generated Java class methods to enable mime:content mapping, similar to `enableMIMEContent` in a JAX-WS binding file.
`-faultSerialVersionUID <`*`fault-serialVersionUID`*`>`	How to generate suid of fault exceptions. Use NONE, TIMESTAMP, FQCN, or a specific number. Default is NONE.
`-mark-generated`	Adds the @Generated annotation to classes generated.
`wsdlurl`	The path and name of the WSDL file to use in generating the code.

Using Ant

To call the WSDL to Java code generator from Ant set the **java** task's classname property to `org.apache.cxf.tools.wsdlto.WSDLToJava`. Example 7 shows the **java** task to execute this command.

Example 7. Generating a Java Code From Ant

```
<java classname="org.apache.cxf.tools.wsdlto.WSDLToJava"
              fork="true">
  <arg value="-client"/>
  ...
  <arg value="MyWSDL.wsdl"/>
  <classpath>
    <path refid="fsf.classpath"/>
  </classpath>
</java>
```

java2js

Generates JavaScript code from a Java SEI.

Synopsis

`java2js` [[-?] | [-help] | [-h]] [-jsutils] [-o *outFile*] [-d *outDir*] [-beans *beanPath*...] [-cp *classpath*] [-soap12] [-v] [[-verbose] | [-quiet]] *classname*

Description

java2js takes a compiled Java SEI and generates JavaScript code from which to implement a client that is capable of interacting with a service implementing the service interface.

Arguments

The arguments used to manage the code generation process are reviewed in the following table.

Option	Interpretation
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-jsutils`	Instructs the tool to put the Artix JavaScript utility code at the top of the generated file.
`-o` *`outFile`*	Specifies the name of the generated file.
`-d` *`outDir`*	Specifies the name of the directory into which the generated file is placed.
`-beans` *`beanPath`*	Specify the pathname of a file defining additional Spring beans to customize data binding configuration.
`-cp` *`classpath`*	Specifies the classpath used to discover the SEI and required support files.
`-soap12`	Instructs the tool to generate a SOAP 1.2 binding.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`classname`	Specifies the name of the SEI class.

wsdl2js

Generates JavaScript consumer code from a WSDL document

Synopsis

`wsdl2js` [[-?] | [-help] | [-h]] [-wv *`wsdlVersion`*] [-p {*`wsdlNamespace`* [=*`jsPrefix`*]}...] [-catalog *`catalogUrl`*] [-d *`outDir`*] [-validate] [-v] [[-verbose] | [-quiet]] *`wsdlUrl`*

Description

wsld2js takes a WSDL document and generates JavaScript code from which to implement a consumer capable of interacting with a service provider implementing the described service. The WSDL document must have a valid `portType` element, but it does not need to contain a `binding` element or a `service` element.

Arguments

The arguments used to manage the code generation process are reviewed in the following table.

Option	**Interpretation**
`-?`	Displays the online help for this utility.
`-help`	
`-h`	
`-wv` *`wsdlVersion`*	Specifies the WSDL version the tool expects. The default is WSDL 1.1. The tool can also use WSDL 1.2.
`-p` *`wsdlNamespace`*`[=`*`jsPrefix`*`]`	Specifies a mapping between the namespaces used in the WSDL document and the prefixes used in the generated JavaScript. This argument can be used more than once.
`-catalog` *`catalogUrl`*	Specifies the URL of an XML catalog to use for resolving imported schemas and WSDL documents.
`-d` *`outDir`*	Specifies the directory into which the generated code is written.
`-validate`	Instructs the tool to validate the WSDL document before attempting to generate any code.
`-v`	Displays the version number for the tool.
`-verbose`	Displays comments during the code generation process.
`-quiet`	Suppresses comments during the code generation process.
`wsdlUrl`	Specifies the location of the WSDL document from which the code is generated.

Generating Support Files

Artix provides tools to generate a number of support files.

This chapter describes the following tools:

- wsdl2corba -idl

wsdl2corba -idl

Generates an IDL file from a WSDL document containing a CORBA binding.

Synopsis

`wsdl2corba` {-idl} {-b *`binding`*} [-corba] [-i *`portType`*] [-d *`dir`*] [-w *`wsdlOut`*] [-o *`idlOut`*...] [-props *`namespace`*] [-wrapped] [-a *`address`*] [-f *`address-file`*] [[-quiet] | [-verbose]] [-v] [-h] *`wsdl`*

Description

wsdl2corba -idl generates an IDL file from a WSDL document containing a CORBA binding. In addition, the tool can be used to add a CORBA binding to a WSDL file and generate an IDL file in one step.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
`-idl`	Specifies that the tool is to generate IDL from the binding.
`-b` *`binding`*	Specifies the name of the CORBA binding for which the IDL file is generated.
`wsdl`	Specifies the WSDL document to which the binding is added.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
`-corba`	Specifies that an CORBA binding will be added to the WSDL document. You must also use the `-i` flag in conjunction with this flag.
`-i` *`portType`*	Specifies the name of the port type for which the CORBA binding is generated.
`-d` *`dir`*	Specifies the directory into which the new IDL file is written.
`-w` *`wsdlOut`*	Specifies the name of the WSDL document containing the generated CORBA binding.
`-o` *`idlOut`*	Specifies the name of the generated IDL file.
`-props` *`namespace`*	Specifies the namespace to use for the generated CORBA typemap.
`-wrapped`	Specifies that the generated binding uses wrapped types.
`-a` *`address`*	Specifies the value of the generated binding's `corba:address` element's `location` attribute.
`-f` *`address-file`*	Specifies the name of a file whose contents are to be used as the value of the generated binding's `corba:address` element's `location` attribute.
`-v`	Displays the tool's version.
`-h`	Specifies that the tool will display a detailed usage statement.
`-quiet`	Specifies that the tool is to run in quiet mode.
`-versbose`	Specifies that the tool is to run in verbose mode.