Progress | Artix

Artix ESB Java Runtime Command Reference

Version 5.6, August 2011

PROGRESS software
BUSINESS MAKING PROGRESS™

Progress Software

Publication date 12 Aug 2011

Legal Notices

Third Party Acknowledgements -- See Third Party Acknowledgements on page 11.

Table of Contents

List of Examples

Preface

What is Covered in This Book

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

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 ESB can benefit from using this as a reference.

The Artix ESB Documentation Library

For information on the organization of the Artix ESB library, the document conventions used, and where to find additional resources, see Using the Artix ESB Library[1].
See the entire documentation set at the Artix Product Documentation Web Site[2]

[1] http://documentation.progress.com/output/Iona/artix/5.6/library_intro/library_intro.pdf
[2] http://communities.progress.com/pcom/docs/DOC-106903

Third Party Acknowledgements

Progress Artix ESB v5.6 incorporates Apache Commons Codec v1.2 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 2001-2003 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear. 4. The names "Apache", "The Jakarta Project", "Commons", and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", "Apache" nor may "Apache" appear in their name without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org/.

Progress Artix ESB v5.6 incorporates Jcraft JSCH v0.1.44 from Jcraft. Such technology is subject to the following terms and conditions: Copyright (c) 2002-2010 Atsuhiko Yamanaka, JCraft,Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT, INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Prerequisites

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

To set up your environment to use Artix ESB 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.

Name

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`	
`-h`	
`-frontend {jaxws|simple}`	Specifies front end to use for processing the SEI and generating the support classes. `jaxws` is the default.

Option	Interpretation
`-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.
`-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 on page 18 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>
```

Name

idl2wsdl — generates an Artix ESB Java Runtime compliant WSDL document from a CORBA IDL file

Synopsis

`idl2wsdl` [-l *`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.

Option	Interpretation
-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.
-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.

Option	Interpretation
`-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.

Name

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.

Option	Interpretation
`-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 on page 23 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.

Name

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.

Option	Interpretation
`-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 on page 27 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>
```

Name

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.

Option	Interpretation
`-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.
`-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 on page 29 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>
```

Name

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

Tip

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.

Option	Interpretation
`-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.

Name

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.
`-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.

Option	Interpretation
`-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 on page 35 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>
```

Name

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.

Option	Interpretation
`-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.
`-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 on page 37 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.

Name

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

Name

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 WSDLGen Guide[1].

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 on page 43 for details.
`-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.

[1] ../wsdlgen/index.htm

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.

Name

wsdl2java — generates JAX-WS compliant Java code from a WSDL document

Synopsis

`wsdl2java` [[-?] | [-help] | [-h]] [-fe *frontend*...] [-db *databinding*...] [-wv *wsdlVersion*...] [-p [*wsdlNamespace*=]*PackageName*...] [-b *bindingName*...] [-sn *serviceName*] [-d *output-directory*] [-catalog *catalogName*] [-compile] [-classdir *compile-class-dir*] [-client] [-server] [-impl] [-all] [-ant] [-keep] [-defaultValues[=*DefaultValueProvider*]] [-nexclude *schema-namespace* [=*java-packagename*]...] [-exsh { true | false }] [-dns { true | false }] [-dex { true | false }] [-wsdlLocation *wsdlLocation*] [-xjc*args*] [-noAddressBinding] [-validate] [-v] [[-verbose] | [-quiet]] *wsdlfile*

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` *frontend*	Specifies the front end used by the code generator. The default is `jaxws`.[a]
`-db` *databinding*	Specifies the data binding used by the code generator. The default is `jaxb`.[b]

Option	Interpretation
`-wv wsdlVersion`	Specifies the WSDL version expected by the tool. The default is `1.1`.[c]
`-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 bindingName`	Specifies zero, or more, JAXWS or JAXB binding files. Use spaces to separate multiple entries.
`-sn serviceName`	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 catalogUrl`	Specifies the URL of an XML catalog to use for resolving imported schemas and WSDL documents.
`-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.
`-keep`	Instructs the tool to not overwrite any existing files.
`-defaultValues[=DefaultValueProvider]`	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).

Option	Interpretation
`-exsh (true/false)`	Enables or disables processing of extended soap header message binding. Default is `false`.
`-dns (true/false)`	Enables or disables the loading of the default namespace package name mapping. Default is `true`.
`-dex (true/false)`	Enables or disables the loading of the default excludes namespace mapping. Default is `true`.
`-wsdlLocation` *wsdlLocation*	Specifies the value of the `@WebService` annotation's wsdlLocation property.
`-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`.
`-noAddressBinding`	Instructs the tool to use the Artix ESB proprietary WS-Addressing type instead of the JAX-WS 2.1 compliant mapping.
`-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.
wsdlfile	The path and name of the WSDL file to use in generating the code.

[a]Currently, Artix ESB only provides the JAX-WS front end for the code generator.

[b]Currently, Artix ESB only provides the JAXB data binding for the code generator.

[c]Currently, Artix ESB only provides WSDL 1.1 support for the code generator.

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 on page 46 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>
```

Name

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 ESB 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.

Option	Interpretation
classname	Specifies the name of the SEI class.

Name

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.

Option	Interpretation
`-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.

Name

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.

Option	Interpretation
`-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.