



Artix ESB

Artix ESB Command Reference

Version 5.1
December 2007

Making Software Work Together™

Artix ESB Command Reference

IONA Technologies

Version 5.1

Published 16 Sep 2008

Copyright © 2001-2008 IONA Technologies PLC

Trademark and Disclaimer Notice

IONA Technologies PLC and/or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this publication. Except as expressly provided in any written license agreement from IONA Technologies PLC, the furnishing of this publication does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Any rights not expressly granted herein are reserved.

IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate, IONA Technologies PLC makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IONA shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Copyright Notice

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this publication. This publication and features described herein are subject to change without notice. Portions of this document may include Apache Foundation documentation, all rights reserved.

Table of Contents

Preface	7
What is Covered in This Book	8
Who Should Read This Book	9
The Artix ESB Documentation Library	10
Using the Artix Tools	11
Generating WSDL	17
artix java2wsdl	18
javatowsdl	20
artix idl2wsdl	22
idltowsdl	25
artix cobol2wsdl	28
artix xsd2wsdl	30
xsdtowsdl	32
artix dbconfig2wsdl	34
dbconfigtowsdl	36
Adding Bindings	39
artix wsdl2soap	40
artix wsdl2xml	42
artix wsdl2idl -corba	44
wsdltosoap	46
wsdltocorba -corba	48
Adding Endpoints	51
artix wsdl2service -transport http	52
artix wsdl2service -transport jms	54
wsdltoervice -transport http/soap	56
wsdltoervice -transport corba	61
wsdltoervice -transport iiop	63
wsdltoervice -transport mq	65
wsdltoervice -transport tibrv	70
wsdltoervice -transport tuxedo	74
Adding Routes	77
wsdltorouting	78
Validating WSDL	81
artix validator	82
schemavalidator	83
Transforming XML	85
xslttransform	86
Generating Code from WSDL	87
artix wsdlgen	88
artix wsdl2cpp	90
artix wsdl2java	94

wSDLtojava	99
artix wsdI2dbservice	102
wsdltodbService	105
Generating Support Files	107
artix wsdI2idl -idl	108
wsdltocorba -idl	110
artix sql2dbconfig	112
wsdd	117
artix wsdI2acl	119

List of Examples

1. Using Artix ESB Java Runtime Tools from Ant	13
2. Generating a WSDL from a Schema Using Ant	31
3. Generating a SOAP Binding From Ant	41
4. Generating a SOAP Binding From Ant	43
5. Generating an HTTP Binding From Ant	53
6. Generating a JMS Binding From Ant	55
7. Generating a Java Code From Ant	98

Preface

What is Covered in This Book	8
Who Should Read This Book	9
The Artix ESB Documentation Library	10

What is Covered in This Book

This book is a reference to the command line tools included with Artix ESB. The command line tools replicates the functionality of the GUI.

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.

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](#) [http://www.ionas.com/support/docs/artix/5.1/library_intro/index.htm].

Using the Artix Tools

For the most commonly used tools, Artix ESB provides a universal access point through the **artix** tool. However, legacy tools, such as the JAX-RPC code generators, are only accessible by explicitly calling them.

Overview

Artix ESB Java Runtime and Artix ESB C++ Runtime share a number of command line tools. To make it easier to determine which version of the tools you are using, Artix has integrated a number of the tools into a single tool. The new tool, **artix** provides access to the the C++ and JAX-WS code generators, the Artix ESB Java Runtime tool for adding a service to a WSDL document, the Artix ESB Java Runtime CORBA tools, and the DB service tools.

The Artix ESB C++ Runtime version of the tools and the JAX-RPC code generators can be called by explicitly calling them.

The artix tool

The **artix** tool provides access to the following tools:

- **wsdl2java**



Warning

This is the JAX-WS version of the Java code generator and generates code the only runs using Artix ESB Java Runtime.

- **wsdl2cpp**
- **java2wsdl**



Warning

This WSDL generator only requires JAX-WS classes that can be run using Artix ESB Java Runtime.

- **wsdlgen**
- **validator**
- **idl2wsdl**



Warning

This version of the tool generates WSDL that is only usable by Artix ESB Java Runtime.

- **wSDL2idl**



Warning

This version of the tool only works with WSDL containing a Artix ESB Java Runtime CORBA binding.

- **wSDL2service**



Important

This version of the tool only generates SOAP and JMS endpoints.

- **wSDL2soap**
- **wSDL2xml**
- **xsd2wSDL**
- **wSDL2dbservice**
- **dbconfig2wSDL**
- **wSDLgen**
- **cobol2brgeinfo**
- **idl2cobol**
- **idl2pli**
- **wSDL2pli**
- **wSDL2cobol**
- **transformerstore**
- **wSDL2acl**

- **sql2dbconfig**

You can retrieve this list by using the following command:

```
artix
```

You can view the options for each of the tools using the following command:

```
artix tool
```

Other tools

The JAX-RPC version of the Java code generator, the JAX-RPC version of the WSDL generator, and the Artix ESB C++ Runtime versions of the IDL tools are still available as part of Artix ESB. You can use them, as well as the mainframe tools, by explicitly calling them from the command line and specifying the full path to the tool.

The Artix ESB C++ Runtime version of the tools are located in *InstallDir/cxx_java/bin*.

The mainframe tools are located in *InstallDir/mainframe/bin*.

Using Ant

The Artix ESB Java Runtime command line tools can all be called directly from Apache Ant using Ant's **java** task. To use the commands from Ant you need to supply the tool's main class as the value of the **java** task's **classname** property. In addition, you need to set the **java** task's **fork** property to true.

You will also need to set the task's classpath to include the Artix ESB Java Runtime's tools jar. The easiest way to do this is add *InstallDir/java/lib/*.jar* to the classpath.

[Example 1 on page 13](#) shows an example of calling the Artix ESB Java Runtime JAX-WS code generator from Ant.

Example 1. Using Artix ESB Java Runtime Tools from Ant

```
<project name="artix wsdl2java" basedir=".">
  <property name="artix_java.home" location
    ="/usr/artix/java"/>
  <path id="artix_java.classpath">
    <fileset dir="${artix_java.home}/lib">
      <include name="*.jar"/>
    </fileset>
  </path>
```

```

<target name="WSDLToJava">
  <java classname="org.apache.cxf.tools.wsdlto.WSDLToJava"
    fork="true">
    <arg value="-client"/>
    <arg value="-d"/>
    <arg value="src"/>
    <arg value="MyWSDL.wsdl"/>
    <classpath>
      <path refid="artix_java.classpath"/>
    </classpath>
  </java>
</target>
</project>

```

You pass options to the command line tool using the `arg` element. Each option requires a separate `arg` element. Options that consist of two parts must be split across two `arg` elements. For example to specify the name of an output file would require you to specify the `-o` with one `arg` element and the name of the output file in a second `arg` element.

Prerequisites

Before you can use these tools you must properly set up your environment. How you set up your environment depends on which runtime you are intending to use.

To set up your environment to use Artix ESB C++ Runtime do the following:

1. Run the **artix_env** script located in `InstallDir/cxx_java/bin`.
2. If you are going to be developing JAX-RPC applications, ensure that `JAVA_HOME` is pointing to a valid Java 5 (or higher) JDK.



Warning

The Artix installer only installs a JRE.

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

1. Run the **artix_java_env** script located in `InstallDir/java/bin`.
2. Ensure that `JAVA_HOME` points to a Java 5 (or higher) JDK.



Warning

The Artix installer only installs a JRE.

Generating WSDL

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

artix java2wsdl	18
javatowsdl	20
artix idl2wsdl	22
idltowsdl	25
artix cobol2wsdl	28
artix xsd2wsdl	30
xsdtowsdl	32
artix dbconfig2wsdl	34
dbconfigtowsdl	36

Name

`artix java2wsdl` — generates a WSDL document from a JAX-WS compliant Java class

Synopsis

```
artix java2wsdl [-o output-file] [-cp class-path] [-soap12] [-t
target-namespace] [-servicename service-name] [-v] [[-verbose] | [-quiet]]
classname
```

Description

artix java2wsdl uses a Web service endpoint's implementation (SEI) class and associated type classes to generate a WSDL file.



Important

This tool will search and load the service endpoint class and types classes. Make certain these classes are on the `CLASSPATH` or in a location identified through the `-cp` flag.

Required Arguments

The command has the following required arguments:

Option	Interpretation
<code><i>classname</i></code>	Specifies the name of the SEI class.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-o <i>output-file</i></code>	Specifies the name of the generated WSDL file.
<code>-cp <i>classpath</i></code>	Specify the SEI and types class search path of directories and zip/jar files.
<code>-soap12</code>	Specifies that the generated WSDL is to include a SOAP 1.2 binding.

Option	Interpretation
<code>-t target-namespace</code>	Specifies the target namespace to use in the generated WSDL file.
<code>-servicename service-name</code>	Specifies the value of the generated <code>service</code> element's <code>name</code> attribute.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.
<code>-quiet</code>	Suppresses comments during the code generation process.

Calling from Ant

To call this tool from Ant you execute the `org.apache.cxf.tools.java2wsdl.JavaToWSDL` class.

Name

`javatowsdl` — generates a WSDL document from a JAX-RPC compliant Java class

Synopsis

```
javatowsdl [-h] [-o output-file] [-t target-namespace] [-x
schema-namespace] [-i portType] [-useTypes] [-qualified] [-L license] [-v]
[[-verbose] | [-quiet]] classname
```

Description

javatowsdl uses the mapping rule described in Sun's JAX-RPC 1.1 specification to generate a WSDL file from a Java class.

The generated WSDL will not contain any physical details concerning the payload formats or network transports that will be used when exposing the service. You will need to add this information manually.



Warning

When generating contracts, **javatowsdl** will add newly generated WSDL to an existing contract if a contract of the same name exists. It will not generate a new file or warn you that a previous contract exists.

Arguments

The tool has the following required arguments:

Option	Interpretation
<code>classname</code>	Specifies the name of the Java class.

Arguments

The tool has the following optional arguments:

Option	Interpretation
-h	Displays the online help for this utility.
-o <i>output-file</i>	Specifies the name of the generated WSDL file.
-t <i>target-namespace</i>	Specifies the target namespace of the generated WSDL document. By default, the java package name will be used as the target namespace. If no package name is specified, the generated target namespace will be <code>http://www.iona.com\ClassName</code> .
-x <i>schema-namespace</i>	Specifies the target namespace of the XML Schema information generated to represent the data types inside the WSDL document. By default, the generated target namespace of the XML Schema will be <code>http://www.iona.com\ClassName\xsd</code> .
-i <i>portType</i>	Specifies the name of the generated <code>portType</code> in the WSDL document. By default the name of the class from which the WSDL is generated is used.
-useTypes	Specifies that the generated WSDL will use types in the WSDL message parts. By default, messages are generated using wrapped doc/literal style. A wrapper element with a sequence will be created to hold method parameters.
-qualified	Specifies that the generated WSDL is fully qualified..
-L <i>license</i>	Specifies the location of your Artix ESB license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</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.

Name

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

Synopsis

```
artix 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] [-mnsnamespaceMapping] [-ow wsdloutput-file]
[excludedModules] [-pf] [-v] [[-verbose] | [-quiet]] idl
```

Description

artix 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
<i>idl</i>	Specifies the name of the IDL file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-I <i>idl-include-dir</i></code>	Specify a directory to be included in the search path for the IDL preprocessor. You can use this flag multiple times.

Option	Interpretation
<code>-o output-dir</code>	Specifies the directory into which the WSDL file is written.
<code>-a corba-address</code>	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.
<code>-b</code>	Specifies that bounded strings are to be treated as unbounded. This eliminates the generation of the special types for the bounded string.
<code>-f corba-address-file</code>	Specifies a file containing a string representation of an object reference. The object reference is placed in the <code>corba:address</code> element in the port definition of the generated service. The file must exist when you run the IDL compiler.
<code>-n schema-import-file</code>	Specifies that a schema file is to be included in the generated contract by an import statement. This option cannot be used with the <code>-T</code> option.
<code>-s idl-sequence-type</code>	Specifies the XML Schema type used to map the IDL sequence<octet> type. Valid values are <code>base64Binary</code> and <code>hexBinary</code> . The default is <code>base64Binary</code> .
<code>-w target-namespace</code>	Specifies the namespace to use for the WSDL document's target namespace.
<code>-x schema-namespace</code>	Specifies the namespace to use for the generated XML Schema's target namespace.
<code>-t corba-typemap-namespace</code>	Specifies the namespace to use for the CORBA type map's target namespace.
<code>-L logical-wsdl-filename</code>	Specifies that the logical portion of the generated WSDL specification into is written to <code>logical-wsdl-filename</code> . The logical WSDL is then imported into the default generated file.
<code>-P physical-wsdl-filename</code>	Specifies that the physical portion of the generated WSDL specification into is written to <code>physical-wsdl-filename</code> . The physical WSDL is then imported into the default generated file.
<code>-T schema-filename</code>	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 <code>-n</code> option.
<code>-qualified</code>	Generates fully qualified WSDL.
<code>-e xml-encoding-type</code>	Specifies the value for the generated WSDL document's xml encoding attribute. The default is UTF-8.
<code>-mnsnamespaceMapping</code>	Specifies a mapping between IDL modules and XML namespaces.
<code>-ow wsdloutput-file</code>	Specifies the name of the generated WSDL file.
<code>-exexcludeModules</code>	Specifies one or more IDL modules to exclude when generating the WSDL file.

Generating WSDL

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

`idltowsdl` — generates an Artix ESB C++ Runtime compliant WSDL document from a CORBA IDL file

Synopsis

```
idltowsdl [-usetypes] [-unwrap] [-a address] [-f file] [-o dir] [-s type]
[-r file] [-L file] [-P file] [-w namespace] [-x namespace] [-t namespace]
[-T file] [-n file] [-b] [-l idlDir...] [-qualified] [-inline] [-3] [-fasttrack]
[-interface name] [-soapaddr port] [-e encoding] [-L file] [-h] [-v] [[-quiet]
| [-verbose]] idlfile
```

Description

`idltowsdl` supports several command line flags that specify how to create 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
<code>idlfile</code>	Specifies the name of the IDL file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-usetypes</code>	Generate rpc style messages. rpc style messages have parts defined using XML Schema types instead of XML elements.
<code>-unwrap</code>	Generate unwrapped doc/literal messages. Unwrapped messages have parts that represent individual elements. Unlike wrapped messages, unwrapped messages can have multiple parts and are not allowed by the WS-I.

Option	Interpretation
<code>-a address</code>	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.
<code>-f file</code>	Specifies a file containing a string representation of an object reference. The object reference is placed in the <code>corba:address</code> element in the port definition of the generated service.  Important The file must exist when you run the IDL compiler.
<code>-o dir</code>	Specifies the directory into which the WSDL file is written.
<code>-s type</code>	Specifies the XML Schema type used to map the IDL sequence<octet> type. Valid values are <code>base64Binary</code> and <code>hexBinary</code> . The default is <code>base64Binary</code> .
<code>-r file</code>	Specify the pathname of the schema file imported to define the Reference type. If the <code>-r</code> option is not given, the idl compiler gets the schema file pathname from <code>etc/idl.cfg</code> .
<code>-L file</code>	Specifies that the logical portion of the generated WSDL is written to <code>file</code> . <code>file</code> is then imported into the default generated file.
<code>-P file</code>	Specifies that the physical portion of the generated WSDL is written to <code>file</code> . <code>file</code> is then imported into the default generated file.
<code>-w namespace</code>	Specifies the namespace to use for the WSDL document's target namespace. The default is <code>http://schemas.ionas.com/idl/idl_name</code> .
<code>-x namespace</code>	Specifies the namespace to use for the generated XML Schema's target namespace. The default is <code>http://schemas.ionas.com/idltypes/idl_name</code> .
<code>-t namespace</code>	Specifies the namespace to use for the CORBA type map's target namespace. The default is <code>http://schemas.ionas.com/typemap/corba/idl_name</code> .
<code>-T file</code>	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 cannot be used with <code>-n</code> .
<code>-n file</code>	Specifies that a schema file, <code>file</code> , is to be included in the generated contract by an import statement. This cannot be used with <code>-T</code> .
<code>-b</code>	Specifies that bounded strings are to be treated as unbounded. This eliminates the generation of the special types for the bounded strings.
<code>-I idlDir</code>	Specify a directory to be included in the search path for the IDL preprocessor. You can use this flag multiple times.

Option	Interpretation
-qualified	Generates fully qualified WSDL.
-inline	Generates a contract that includes all imported documents in-line. This overrides all options that specify that a section of the contract is to be imported.
-3	Use relaxed IDL grammar checking semantics to allow IDL used by Orbix 3 to be parsed.
-fastrack	Use the fastrack wizard. You must also use the <code>-interface</code> and <code>-soapaddr</code> flags with this option. This option also adds a SOAP port and a route between the generated CORBA port and the generated SOAP port.
-interface <i>name</i>	Specifies the IDL interface for which WSDL will be generated by the fastrack wizard.
-soapaddr <i>port</i>	Specifies the SOAP address to use in the generated <code>port</code> element when using the fastrack wizard.
-e <i>encoding</i>	Specifies the value for the generated WSDL document's xml encoding attribute. The default is UTF-8.
-L <i>file</i>	Specifies the location of your license file. The default is <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-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

`artix cobol2wsdl` — generates a WSDL document with a fixed binding from a COBOL copybook

Synopsis

```
artix cobol2wsdl {-b binding} {-op operation} {-im
[inmessage:]incopybook} [-om [outmessage:]outcopybook] [-fm
[faultmessage:]faultbook] [-i portType] [-t target] [-x schema_name]
[-useTypes] [-oneway] [-qualified] [-o file] [-L file] [-quiet] [-h] [-v]
[-verbose]
```

Required Arguments

The command has the following required arguments:

Option	Interpretation
<code>-b <i>binding</i></code>	Specifies the name for the generated binding.
<code>-op <i>operation</i></code>	Specifies the name for the generated operation.
<code>-im [<i>inmessage</i>:]<i>incopybook</i></code>	Specifies the name of the input message and the copybook file from which the data defining the message is taken. The input message name, <i>inmessage</i> , is optional. However, if the copybook has more than one 01 levels, you will be asked to choose the one you want to use as the input message.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-om [<i>outmessage</i>:]<i>outcopybook</i></code>	Specifies the name of the output message and the copybook file from which the data defining the message is taken. The output message name, <i>outmessage</i> , is optional. However, if the copybook has more than one 01 levels, you will be asked to choose the one you want to use as the output message.

Option	Interpretation
-fm [<i>faultmessage</i> :] <i>faultcopybook</i>	Specifies the name of a fault message and the copybook file from which the data defining the message is taken. The fault message name, <i>faultmessage</i> , is optional. However, if the copybook has more than one <code>01</code> levels, you will be asked to choose the one you want to use as the fault message. You can specify more than one fault message.
-i <i>portType</i>	Specifies the name of the port type in the generated WSDL. Defaults to <i>bindingPortType</i> ^a .
-t <i>target</i>	Specifies the target namespace for the generated WSDL. Defaults to <code>http://www.iona.com/binding</code> .
-x <i>schema_name</i>	Specifies the namespace for the schema in the generated WSDL. Defaults to <code>http://www.iona.com/binding/types</code> .
-useTypes	Specifies that the generated WSDL will use <code>type</code> elements. Default is to generate <code>element</code> elements for schema types.
-oneway	Specifies that the operation does not have a response message.
-qualified	Specifies that the <code>schema</code> element in the generated WSDL has its <code>elementFormDefault</code> and <code>attributeFormDefault</code> attributes set to <code>qualified</code> .
-o <i>file</i>	Specifies the name of the generated WSDL file. Defaults to <i>binding.wsdl</i> .
-L <i>file</i>	Specifies the location of your license file. The default is <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-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.

^aIf *binding* ends in `Binding` or `binding`, it is stripped off before being used in any of the default names.

Name

`artix xsd2wsdl` — generates a WSDL document containing the types defined in an XML Schema document

Synopsis

```
artix xsd2wsdl {-t target-namespace} [-n wsdl-name] [-d
output-directory] [-o output-file] [-v] [[-verbose] | [-quiet]] xsdurl
```

Description

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

Required Arguments

The command has the following required arguments:

Option	Interpretation
<code>-t <i>target-namespace</i></code>	Specifies the target namespace for the generated WSDL.
<code><i>xsdurl</i></code>	The path and name of the existing XML Schema file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-n <i>wsdl-name</i></code>	Specifies the value of the generated <code>definition</code> element's <code>name</code> attribute.
<code>-d <i>output-directory</i></code>	Specifies the directory in which the generated WSDL is placed.
<code>-o <i>output-file</i></code>	Specifies the name of the generated WSDL file.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.
<code>-quiet</code>	Suppresses comments during the code generation process.

Using Ant

[Example 2 on page 31](#) 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="artix_java.classpath"/>
  </classpath>
</java>
```

Name

`xsdtoWSDL` — generates a WSDL document containing the types defined in an XML Schema document

Synopsis

```
xsdtoWSDL [ -t namespace ] [ -n name ] [ -d dir ] [ -o file ] [ -L file ]
[ -h ] [ -v ] [ [ -quiet ] | [ -verbose ] ] xsdurl
```

Description

xsdtoWSDL imports an XML Schema document and generates a WSDL contract containing a types element populated by the types defined in the XML Schema document. The rest of the contract will be empty.

Arguments

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

Option	Interpretation
<code>-t namespace</code>	Specifies the target namespace for the generated contract. The default is to use the Artix target namespace.
<code>-n name</code>	Specifies the name for the generated contract and is the value of the <code>name</code> attribute in the contract's root <code>definitions</code> element. The default is to use the schema document's file name.
<code>-d dir</code>	Specifies the output directory for the generated contract.
<code>-o file</code>	Specifies the filename for the generated contract. Defaults to the filename of the imported schema document. For example, if the imported schema document is stored in <code>maxwell.xsd</code> the resulting contract will be <code>maxwell.wsdl</code> .
<code>-L file</code>	Specifies the location of your license file. The default is <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.

Option	Interpretation
<code>-quiet</code>	Suppresses comments during the code generation process.
<code>xsdurl</code>	Specifies the URL of the XML Schema document.

Name

`artix dbconfig2wsdl` — generates a WSDL document from an Artix ESB Java Runtime database configuration file

Synopsis

```
artix dbconfig2wsdl [-a address] [-d output-dir] [-servicename
service-name] [-jdbctypemappings jdbc-type-mapping-file] [-mp {
element | type } ] [-t target-namespace] [-o output-file] [-logical] [-v]
[[-verbose] | [-quiet]] dbconfigurl
```

Description

artix dbconfig2wsdl imports an Artix ESB Java Runtime database configuration document and generates an Artix ESB Java Runtime contract defining a service that represents the database operations defined in the document.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>dbconfigurl</code>	Specifies the URL of the database configuration file.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-a address</code>	Specifies the value of generated <code>soap:address</code> element's <code>location</code> attribute.
<code>-d output-dir</code>	Specifies the folder into which the generated WSDL is placed.
<code>-servicename name</code>	Specifies the value of the generated <code>service</code> element's <code>name</code> attribute. The default is <code>DataService</code> .
<code>-jdbctypemappings jdbc-type-mapping-file</code>	Specifies the name of the file containing the mappings between JDBC types and XSD types.

Option	Interpretation
-mp {element type}	Specifies if the generated message parts should be types or elements. The default is elements.
-t <i>target-namespace</i>	Specifies the target namespace for the generated contract.
-o <i>output-file</i>	Specifies the name of the generated WSDL document.
-logical	Specifies the tool only generates the logical portion of the 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.

Name

`dbconfigtowsdl` — generates a WSDL document from an Artix ESB C++ Runtime database configuration file

Synopsis

```
dbconfigtowsdl [-a bindingAddress] [-fasttrack] [-plugin] [-d dir] [-source
dir] [-h] [-v] [[-quiet] | [-verbose]] dbconfigurl
```

Description

dbconfigtowsdl imports an Artix ESB C++ Runtime database configuration document and generates an Artix ESB C++ Runtime contract defining a service that represents the database operations defined in the document.

Required Arguments

The command has the following required arguments:

Option	Interpretation
<code>dbconfigurl</code>	Specifies the URL of the database configuration file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-t <i>bindingAddress</i></code>	Specifies the address to use in the port element of the generated WSDL. This flag is only valid when <code>-fasttrack</code> is also used. The default is <code>http://localhost:9000/DBConnection</code> .
<code>-fasttrack</code>	Specifies that the tool will generate a default SOAP binding and HTTP endpoint for the database operations. In addition, the tool will generate the code for the intermediary required to expose the operations as a service.
<code>-plugin</code>	Specifies that the intermediary is generated as an Artix ESB C++ Runtime plug-in. This flag is only valid when <code>-fasttrack</code> is also used.

Option	Interpretation
<code>-d dir</code>	Specifies the output directory for the generated WSDL file. The default is the local directory. When <code>-fasttrack</code> is used, the default is <code>etc</code> .
<code>-source dir</code>	Specifies the output directory for the generated code. This flag is only valid when <code>-fasttrack</code> is also used. The default is <code>java</code> .
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.
<code>-quiet</code>	Suppresses comments during the code generation process.

Adding Bindings

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

artix wsdl2soap	40
artix wsdl2xml	42
artix wsdl2idl -corba	44
wsdl2soap	46
wsdl2corba -corba	48

Name

`artix wsdl2soap` — generates a WSDL document containing a valid SOAP/HTTP endpoint definition based on a `portType` element

Synopsis

```
artix wsdl2soap {-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

artix wsdl2soap generates a new WSDL file with a SOAP binding from an existing WSDL file containing one or more `portType` elements.

Required Arguments

The tool has the following required arguments:

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

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-b binding-name</code>	Specifies the name of the generated SOAP binding.
<code>-soap12</code>	Specifies that the generated binding will use SOAP 1.2.
<code>-d output-directory</code>	Specifies the directory to place generated WSDL file.
<code>-o output-file</code>	Specifies the name of the generated WSDL file.
<code>-n soap-body-namespace</code>	Specifies the SOAP body namespace when the style is RPC.

Option	Interpretation
-style {document rpc}	Specifies the encoding style (document or RPC) to use in the SOAP binding. The default is <code>document</code> .
-use {literal encoded}	Specifies the binding use (encoded or literal) to use in the SOAP binding. The default is <code>literal</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.

Using Ant

[Example 3 on page 41](#) shows the `java` task to execute this command.

Example 3. Generating a SOAP Binding From Ant

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

Name

`artix wsdl2xml` — generates a WSDL document containing an XML binding based on a `portType` element

Synopsis

```
artix wsdl2xml [-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

artix wsdl2xml generates an XML binding from an existing WSDL document containing a `portType` definition. In addition to an XML binding, the tool adds an HTTP endpoint to the generated WSDL.



Tip

The `http:address` element will be empty unless you specify an address using the `-a` argument.

Required Arguments



The tool has the following required arguments:

Option	Interpretation
<code>-i <i>port-type-name</i></code>	Specifies the <code>portType</code> element to use.
<code><i>wSDLurl</i></code>	The path and name of the existing WSDL file.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-b <i>binding-name</i></code>	Specifies the name of the generated XML binding.

Option	Interpretation
<code>-e service-name</code>	Specifies the value of the generated <code>service</code> element's <code>name</code> attribute.
<code>-p port-name</code>	Specifies the value of the generated <code>port</code> element's <code>name</code> attribute.  Tip To specify multiple <code>port</code> elements, separate the names by a space.
<code>-a address</code>	Specifies the value used in the <code>http:address</code> element of the generated <code>port</code> element.  Note If you do not provide a value the generated <code>http:address</code> element's <code>location</code> attribute will have an empty value.
<code>-d output-directory</code>	Specifies the directory to place generated WSDL file.
<code>-o output-file</code>	Specifies the name of the generated WSDL file.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.
<code>-quiet</code>	Suppresses comments during the code generation process.

Using Ant

[Example 4 on page 43](#) 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

`artix wsdl2idl -corba` — adds an Artix ESB Java Runtime CORBA binding to a WSDL document

Synopsis

```
artix wsdl2idl {-corba} {-i portType} [-idl] [-b binding] [-d dir] [-w
wsdlOut] [-o idlOut...] [-props namespace] [-wrapped] [-a address] [-f
address-file] [-L file] [[-quiet] | [-verbose]] [-v] [-h] wsdl
```

Description

artix 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
<code>-corba</code>	Specifies that the tool will generate a new WSDL file with a CORBA binding.
<code>-i portType</code>	Specifies the name of the interface for which the CORBA binding is generated.
<code>wsdl</code>	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 <code>-b</code> flag in conjunction with this flag.
-b <i>binding</i>	Specifies the name of the generated CORBA binding.
-d <i>dir</i>	Specifies the directory into which the new WSDL document is written.
-w <i>wsdlOut</i>	Specifies the name of the WSDL document containing the generated CORBA binding.
-o <i>idlOut</i>	Specifies the name of the generated IDL file.
-props <i>namespace</i>	Specifies the namespace to use for the generated CORBA typemap.
-wrapped	Specifies that the generated binding uses wrapped types.
-a <i>address</i>	Specifies the value of the generated binding's <code>corba:address</code> element's <code>location</code> attribute.
-f <i>address-file</i>	Specifies the name of a file whose contents are to be used as the value of the generated binding's <code>corba:address</code> element's <code>location</code> attribute.
-L <i>file</i>	Specifies the location of your Artix license file.
-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.

Name

`wsdltosoap` — generates a WSDL document containing an Artix ESB C++ Runtime SOAP binding

Synopsis

```
wsdltosoap {-i portType} {-n namespace} [-soapversion [ 1.1 | 1.2 ]]
[-style [ document | rpc ]] [-use [ literal | encoded ]] [-b binding] [-o file]
[-d dir] [-L file] [[-quiet] | [-verbose]] [-h] [-v] wsdlurl
```

Description

wsdltosoap adds a Artix ESB C++ Runtime SOAP binding to a WSDL document based on the values provided as arguments to the tool.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>-i portType</code>	Specifies the name of the <code>portType</code> element being mapped to a SOAP binding.
<code>-n namespace</code>	Specifies the namespace to use for the SOAP binding.
<code>wsdlurl</code>	Specifies the WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-soapversion {1.1 1.2}</code>	Specifies the SOAP version of the generated binding. Defaults to 1.1.
<code>-style {document rpc}</code>	Specifies the encoding style to use in the SOAP binding. Defaults to <code>document</code> .
<code>-use {literal encoded}</code>	Specifies how the data is encoded. Default is <code>literal</code> .
<code>-o file</code>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.

Option	Interpretation
-d <i>dir</i>	Specifies the output directory for the generated contract.
-L <i>file</i>	Specifies the location of your Artix license file. The default behavior is to check IT_PRODUCT_DIR\etc\license.txt.
-quiet	Specifies that the tool runs in quiet mode.
-verbose	Specifies that the tool runs in verbose mode.
-h	Displays the tool's usage statement.
-v	Displays the tool's version.

Name

`wsdltocorba -corba` — adds an Artix ESB C++ Runtime CORBA binding to a WSDL document

Synopsis

```
wsdltocorba -corba {-i portType} [-idl] [-d dir] [-b binding] [-o file]
[-props namespace] [-wrapped] [-L file] [[-quiet] | [-verbose]] [-h] [-v] wsdl
```

Description

wsdltocorba -corba adds a Artix ESB C++ Runtime CORBA binding to an existing WSDL document. The generated WSDL file will also contain a Artix ESB C++ 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
<code>-i portType</code>	Specifies the name of the port type for which the CORBA binding is generated.
<code>wsdl</code>	Specifies the WSDL document to which the binding is added.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-idl</code>	Specifies that an IDL file will be generated for the generated CORBA binding. You must also use the <code>-b</code> flag in conjunction with this flag.
<code>-d dir</code>	Specifies the directory into which the new WSDL document is written.

Option	Interpretation
<code>-b <i>binding</i></code>	Specifies the name of the generated CORBA binding. The default is <code>portTypeBinding</code> .
<code>-o <i>file</i></code>	Specifies the name of the generated WSDL document. The default is <code>wsdl_file-corba.wsdl</code> .
<code>-props <i>namespace</i></code>	Specifies the namespace to use for the generated CORBA typemap.
<code>-wrapped</code>	Specifies that the generated binding uses wrapped types.
<code>-L <i>file</i></code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.
<code>-quiet</code>	Specifies that the tool is to run in quiet mode.
<code>-verbose</code>	Specifies that the tool is to run in verbose mode.

Adding Endpoints

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

<code>artix wsdl2service -transport http</code>	52
<code>artix wsdl2service -transport jms</code>	54
<code>wsdl2service -transport http/soap</code>	56
<code>wsdl2service -transport corba</code>	61
<code>wsdl2service -transport iiop</code>	63
<code>wsdl2service -transport mq</code>	65
<code>wsdl2service -transport tibrv</code>	70
<code>wsdl2service -transport tuxedo</code>	74

Name

`artix wsdl2service -transport http` — generates a WSDL document containing a valid HTTP endpoint definition from a `binding` element.

Synopsis

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

Description

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

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>-n binding-name</code>	Specifies the binding used to generate the service.
<code>wsdlurl</code>	The path and name of the existing WSDL file.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-e service-name</code>	Specifies the value of the generated <code>service</code> element's <code>name</code> attribute.
<code>-p port-name</code>	Specifies the value of the generated <code>port</code> element's <code>name</code> attribute. To specify multiple port elements, separate the names by a space.
<code>-a address</code>	Specifies the value used in the <code>address</code> element of the port.
<code>-soap12</code>	Specifies that the SOAP version to use is 1.2.

Option	Interpretation
-o <i>output-file</i>	Specifies the name of the generated WSDL file.
-d <i>output-directory</i>	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.

Using Ant

[Example 5 on page 53](#) shows the `java` task to execute this command.

Example 5. Generating an HTTP Binding From Ant

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

Name

`artix wsdl2service -transport jms` — generates a WSDL document containing a valid JMS endpoint definition from a `binding` element.

Synopsis

```
artix wsdl2service -transport jms [-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

`artix wsdl2service -transport jms` creates a new WSDL file containing an JMS service definition from an existing WSDL document containing a binding element.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>-n <i>binding-name</i></code>	Specifies the binding used to generate the service.
<code><i>wsdlurl</i></code>	The path and name of the existing WSDL file.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-e <i>service-name</i></code>	Specifies the value of the generated <code>service</code> element's <code>name</code> attribute.
<code>-p <i>port-name</i></code>	Specifies the value of the generated <code>port</code> element's <code>name</code> attribute. To specify multiple port elements, separate the names by a space.

Option	Interpretation
-jds {queue topic}	Specifies the JMS destination style.
-jpu <i>jndi-provider-URL</i>	Specifies the URL of the JMS JNDI provider.
-jcf <i>initial-context-factory</i>	Specifies the JMS initial context factory.
-jfn <i>jndi-connection-factory-name</i>	Specifies the JMS JNDI connection factory name.
-jdn <i>jndi-destination-name</i>	Specifies the JMS JNDI destination name.
-jmt {text binary}	Specifies the JMS message type.
-jmc {true false}	Specifies if the <code>MessageID</code> is used as the <code>CorrelationID</code> .
-jsn <i>durable-subscriber-name</i>	Specifies an optional durable subscriber name.
-o <i>output-file</i>	Specifies the name of the generated WSDL file.
-d <i>output-directory</i>	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.

Using Ant

[Example 6 on page 55](#) shows the `java` task to execute this command.

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="artix_java.classpath"/>
  </classpath>
</java>
```

Name

`wsdltoservice -transport http/soap` — generates a WSDL document containing an Artix ESB C++ Runtime HTTP endpoint

Synopsis

```
wsdltoservice -transport soap/http [-e service] [-t port] [-b
binding] [-a address] [-hssdt serverSendTimeout] [-hscvt
serverReceiveTimeout] [-hstrc trustedRootCertificates] [-hsuss
useSecureSockets] [-hscst contentType] [-hscsc serverCacheControl]
[-hsscsc supressClientSendErrors] [-hsscscrc
supressClientReceiveErrors] [-hshka honorKeepAlive] [-hsmpps
serverMultiplexPoolSize] [-hsrurl redirectURL] [-hscsl
contentLocation] [-hsce contentEncoding] [-hssst serverType] [-hssc
serverCertificate] [-hsscsc serverCertificateChain] [-hsspk
serverPrivateKey] [-hsspkp serverPrivateKeyPassword] [-hcst
clientSendTimeout] [-hccvt clientReceiveTimeout] [-hctr
trustedRootCertificates] [-hcuss useSecureSockets] [-hcct
contentType] [-hccc clientCacheControl] [-hcar autoRedirect] [-hcun
userName] [-hcp password] [-hcat clientAuthorizationType] [-hca
clientAuthorization] [-hca accept] [-hcal acceptLanguage] [-hcae
acceptEncoding] [-hch host] [-hccn clientConnection] [-hccc
cookie]
[-hcbt browserType] [-hcr referer] [-hcps proxyServer] [-hcpun
proxyUserName] [-hcpp proxyPassword] [-hcpat
proxyAuthorizationType] [-hcpa proxyAuthorization] [-hcccce
clientCertificate] [-hcccc clientCertificateChain] [-hcpk
clientPrivateKey] [-hcpkp clientPrivateKeyPassword] [-o file] [-d
dir] [-L file] [[-quiet] | [-verbose]] [-h] [-v] wSDLurl
```

Description

`wsdltoservice -transport http/soap` adds a Artix ESB C++ Runtime HTTP endpoint to a WSDL document based on the values provided as arguments to the tool.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>wsdurl</code>	Specifies the WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-transport soap/http</code>	If the payload being sent over the wire is SOAP, use <code>-transport soap</code> . For all other payloads use <code>-transport http</code> .
<code>-e service</code>	Specifies the name of the generated service.
<code>-t port</code>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
<code>-b binding</code>	Specifies the name of the binding for which the service is generated.
<code>-a address</code>	Specifies the value used in the <code>address</code> element of the port.
<code>-hssdt serverSendTimeout</code>	Specifies the number of milliseconds that the server can continue to try to send a response to the client before the connection is timed out.
<code>-hscvt serverReceiveTimeout</code>	Specifies the number of milliseconds that the server can continue to try to receive a request from the client before the connection is timed out.
<code>-hstrc trustedRootCertificates</code>	Specifies the full path to the X509 certificate for the certificate authority.
<code>-hsuss UseSecureSockets</code>	Specifies if the server uses secure sockets. Valid values are <code>true</code> or <code>false</code> .
<code>-hsct contentType</code>	Specifies the media type of the information being sent in a server response.
<code>-hsc serverCacheControl</code>	Specifies directives about the behavior that must be adhered to by caches involved in the chain comprising a request from a client to a server.
<code>-hsscse supressClientSendErrors</code>	Specifies whether exceptions are thrown when an error is encountered on receiving a client request. Valid values are <code>true</code> or <code>false</code> .
<code>-hsscre supressClientReceiveErrors</code>	Specifies whether exceptions are thrown when an error is encountered on sending a response to a client. Valid values are <code>true</code> or <code>false</code> .

Adding Endpoints

Option	Interpretation
<code>-hshka honorKeepAlive</code>	Specifies if the server honors client keep-alive requests. Valid values are <code>true</code> or <code>false</code> .
<code>-hsmps serverMultiplexPoolSize</code>	
<code>-hsrurl redirectURL</code>	Specifies the URL to which the client request should be redirected if the URL specified in the client request is no longer appropriate for the requested resource.
<code>-hscl contentLocation</code>	Specifies the URL where the resource being sent in a server response is located.
<code>-hsce contentEncoding</code>	Specifies what additional content codings have been applied to the information being sent by the server, and what decoding mechanisms the client therefore needs to retrieve the information.
<code>-hsst serverType</code>	Specifies what type of server is sending the response to the client.
<code>-hssc serverCertificate</code>	Specifies the full path to the X509 certificate issued by the certificate authority for the server.
<code>-hsscc serverCertificateChain</code>	Specifies the full path to the file that contains all the certificates in the chain.
<code>-hsspk serverPrivateKey</code>	Specifies the full path to the private key that corresponds to the X509 certificate specified by <code>serverCertificate</code> .
<code>-hsspkp serverPrivateKeyPassword</code>	Specifies a password that is used to decrypt the private key.
<code>-hcst clientSendTimeout</code>	Specifies the number of milliseconds that the client can continue to try to send a request to the server before the connection is timed out.
<code>-hccvt clientReceiveTimeout</code>	Specifies the number of milliseconds that the client can continue to try to receive a response from the server before the connection is timed out.
<code>-hctrc trustedRootCertificates</code>	Specifies the full path to the X509 certificate for the certificate authority.
<code>-hcuss ueSecureSockets</code>	Specifies if the client uses secure sockets. Valid values are <code>true</code> or <code>false</code> .
<code>-hcct contentType</code>	Specifies the media type of the data being sent in the body of the client request.
<code>-hccc clientCacheControl</code>	Specifies directives about the behavior that must be adhered to by caches involved in the chain comprising a request from a client to a server.
<code>-hcar autoRedirect</code>	Specifies if the server should automatically redirect client requests.
<code>-hcun userName</code>	Specifies the username the client uses to register with servers.

Option	Interpretation
<code>-hcp password</code>	Specifies the password the client uses to register with servers.
<code>-hcat clientAuthorizationType</code>	Specifies the authorization mechanisms the client uses when contacting servers.
<code>-hca clientAuthorization</code>	Specifies the authorization credentials used to perform the authorization.
<code>-hca accept</code>	Specifies what media types the client is prepared to handle.
<code>-hcal acceptLanguage</code>	Specifies what language the client prefers for the purposes of receiving a response.
<code>-hcae acceptEncoding</code>	Specifies what content codings the client is prepared to handle.
<code>-hch host</code>	Specifies the internet host and port number of the resource on which the client request is being invoked.
<code>-hccn clientConnection</code>	Specifies if the client will open a new connection for each request or if it will keep the original one open. Valid values are <code>close</code> and <code>Keep-Alive</code> .
<code>-hcck cookie</code>	Specifies a static cookie to be sent to the server.
<code>-hcbt browserType</code>	Specifies information about the browser from which the client request originates.
<code>-hcr referer</code>	Specifies the value for the client's referring entity.
<code>-hcps proxyServer</code>	Specifies the URL of the proxy server, if one exists along the message path.
<code>-hcpun proxyUserName</code>	Specifies the username that the client uses to authorize with proxy servers.
<code>-hcpp proxyPassword</code>	Specifies the password that the client uses to authorize with proxy servers.
<code>-hcpat proxyAuthorizationType</code>	Specifies the authorization mechanism the client uses with proxy servers.
<code>-hcpa proxyAuthorization</code>	Specifies the actual data that the proxy server should use to authenticate the client.
<code>-hccce clientCertificate</code>	Specifies the full path to the X509 certificate issued by the certificate authority for the client.
<code>-hcccc clientCertificateChain</code>	Specifies the full path to the file that contains all the certificates in the chain.
<code>-hcpk clientPrivateKey</code>	Specifies the full path to the private key that corresponds to the X509 certificate specified by <code>clientCertificate</code> .
<code>-hcpkp clientPrivateKeyPassword</code>	Specifies a password that is used to decrypt the private key.

Adding Endpoints

Option	Interpretation
-o <i>file</i>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.
-d <i>dir</i>	Specifies the output directory for the generated contract.
-L <i>file</i>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-quiet	Specifies that the tool runs in quiet mode.
-verbose	Specifies that the tool runs in verbose mode.
-h	Displays the tool's usage statement.
-v	Displays the tool's version.

Name

`wsdltoservice -transport corba` — generates a WSDL document containing an Artix ESB C++ Runtime CORBA endpoint

Synopsis

```
wsdltoservice -transport corba [-e service] [-t port] [-b binding]
[-a address] [-poa poaName] [-sid serviceId] [-pst persists] [-o file]
[-d dir] [-L file] [[-quiet] | [-verbose]] [-h] [-v] wsdurl
```

Description

wsdltoservice -transport corba adds a Artix ESB C++ Runtime CORBA endpoint to a WSDL document based on the values provided as arguments to the tool.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>wsdurl</code>	The WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-e service</code>	Specifies the name of the generated CORBA service.
<code>-t port</code>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
<code>-b binding</code>	Specifies the name of the binding for which the service is generated.
<code>-a address</code>	Specifies the value used in the <code>corba:address</code> element of the port.
<code>-poa poaName</code>	Specifies the value of the POA name policy.
<code>-sid serviceId</code>	Specifies the value of the ID assignment policy.

Adding Endpoints

Option	Interpretation
<code>-pst persists</code>	Specifies the value of the persistence policy. Valid values are <code>true</code> and <code>false</code> .
<code>-o file</code>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.
<code>-d dir</code>	Specifies the output directory for the generated contract.
<code>-L file</code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-quiet</code>	Specifies that the tool runs in quiet mode.
<code>-verbose</code>	Specifies that the tool runs in verbose mode.
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.

Name

`wsdltoservice -transport iiop` — generates a WSDL document containing an Artix ESB C++ Runtime IIOp tunnel endpoint

Synopsis

```
wsdltoservice -transport iiop [-e service] [-t port] [-b binding]
[-a address] [-poa poaName] [-sid serviceId] [-pst persists]
[-paytype payload] [-o file] [-d dir] [-L file] [[-quiet] |
[-verbose]] [-h] [-v] wsdurl
```

Description

wsdltoservice -transport iiop adds a Artix ESB C++ Runtime IIOp tunnel endpoint to a WSDL document based on the values provided as arguments to the tool.

Arguments

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

Option	Interpretation
<code>-e service</code>	Specifies the name of the generated CORBA service.
<code>-t port</code>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
<code>-b binding</code>	Specifies the name of the binding for which the service is generated.
<code>-a address</code>	Specifies the value used in the <code>iiop:address</code> element of the port.
<code>-poa poaName</code>	Specifies the value of the POA name policy.
<code>-sid serviceId</code>	Specifies the value of the ID assignment policy.
<code>-pst persists</code>	Specifies the value of the persistence policy. Valid values are <code>true</code> and <code>false</code> .
<code>-paytype payload</code>	Specifies the type of data being sent in the message payloads. Valid values are <code>string</code> , <code>octets</code> , <code>imsraw</code> , <code>imsraw_binary</code> , <code>cicsraw</code> , and <code>cicsraw_binary</code> .

Adding Endpoints

Option	Interpretation
<code>-o file</code>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.
<code>-d dir</code>	Specifies the output directory for the generated contract.
<code>-L file</code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-quiet</code>	Specifies that the tool runs in quiet mode.
<code>-verbose</code>	Specifies that the tool runs in verbose mode.
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.

Name

`wsdltoservice -transport mq` — generates a WSDL document containing an Artix ESB C++ Runtime WebSphere MQ endpoint

Synopsis

```
wsdltoservice -transport mq [-e service] [-t port] [-b binding] [-sqm
queueManager] [-sqn queue] [-srqm queueManager] [-srqn queue] [-smqn
modelQueue] [-sus usageStyle] [-scs correlationStyle] [-sam
accessMode] [-sto timeout] [-sme expiry] [-smp priority] [-smi
messageId] [-sci correlationId] [-sd delivery] [-st transactional]
[-sro reportOption] [-sf format] [-sad applicationData] [-sat
accountingToken] [-scn connectionName] [-sc convert] [-scr reusable]
[-scfp fastPath] [-said idData] [-saod originData] [-cqm queueManager]
[-cq n queue] [-crqm queueManager] [-crqn queue] [-cmqn modelQueue]
[-cus usageStyle] [-ccs correlationStyle] [-cam accessMode] [-cto
timeout] [-cme expiry] [-cmp priority] [-cmi messageId] [-cci
correlationId] [-cd delivery] [-ct transactional] [-cro reportOption]
[-cf format] [-cad applicationData] [-cat accountingToken] [-ccn
connectionName] [-cc convert] [-ccr reusable] [-ccfp fastPath] [-caid
idData] [-caod originData] [-caqn queue] [-cui userId] [-o file] [-d dir]
[-L file] [[-quiet] | [-verbose]] [-h] [-v] wsdLurl
```

Description

wsdltoservice -transport mq adds a Artix ESB C++ Runtime WebSphere MQ endpoint to a WSDL document based on the values provided as arguments to the tool.

Arguments

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

Adding Endpoints

Option	Interpretation
-e <i>service</i>	Specifies the name of the generated service.
-t <i>port</i>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
-b <i>binding</i>	Specifies the name of the binding for which the service is generated.
-sqm <i>queueManager</i>	Specifies the name of the server's queue manager.
-sqn <i>queue</i>	Specifies the name of the server's request queue.
-srqm <i>queueManager</i>	Specifies the name of the server's reply queue manager.
-srqn <i>queue</i>	Specifies the name of the server's reply queue.
-smqn <i>modelQueue</i>	Specifies the name of the server's model queue.
-sus <i>usageStyle</i>	Specifies the value of the server's <code>UsageStyle</code> attribute. Valid values are <code>Peer</code> , <code>Requester</code> , or <code>Responder</code> .
-scs <i>correlationStyle</i>	Specifies the value of the server's <code>CorrelationStyle</code> attribute. Valid values are <code>messageId</code> , <code>correlationId</code> , or <code>messageId copy</code> .
-sam <i>accessMode</i>	Specifies the value of the server's <code>AccessMode</code> attribute. Valid values are <code>peek</code> , <code>send</code> , <code>receive</code> , <code>receive exclusive</code> , or <code>receive shared</code> .
-sto <i>timeout</i>	Specifies the value of the server's <code>Timeout</code> attribute.
-sme <i>expiry</i>	Specifies the value of the server's <code>MessageExpiry</code> attribute.
-smp <i>priority</i>	Specifies the value of the server's <code>MessagePriority</code> attribute.
-smi <i>messageId</i>	Specifies the value of the server's <code>MessageId</code> attribute.
-sci <i>correlationId</i>	Specifies the value of the server's <code>CorrelationId</code> attribute.
-sd <i>delivery</i>	Specifies the value of the server's <code>Delivery</code> attribute.
-st <i>transactional</i>	Specifies the value of the server's <code>Transactional</code> attribute. Valid values are <code>none</code> , <code>internal</code> , or <code>xa</code> .
-sro <i>reportOption</i>	Specifies the value of the server's <code>ReportOption</code> attribute. Valid values are <code>none</code> , <code>coa</code> , <code>cod</code> , <code>exception</code> , <code>expiration</code> , or <code>discard</code> .
-sf <i>format</i>	Specifies the value of the server's <code>Format</code> attribute.

Option	Interpretation
-sad <i>applicationData</i>	Specifies the value of the server's <code>ApplicationData</code> attribute.
-sat <i>accountingToken</i>	Specifies the value of the server's <code>AccountingToken</code> attribute.
-scn <i>connectionName</i>	Specifies the name of the connection by which the adapter connects to the queue.
-sc <i>convert</i>	Specifies if the messages in the queue need to be converted to the system's native encoding. Valid values are <code>true</code> or <code>false</code> .
-scr <i>reusable</i>	Specifies the value of the server's <code>ConnectionReusable</code> attribute. Valid values are <code>true</code> or <code>false</code> .
-scfp <i>fastPath</i>	Specifies the value of the server's <code>ConnectionFastPath</code> attribute. Valid values are <code>true</code> or <code>false</code> .
-said <i>idData</i>	Specifies the value of the server's <code>ApplicationIdData</code> attribute.
-saod <i>originData</i>	Specifies the value of the server's <code>ApplicationOriginData</code> attribute.
-cqm <i>queueManager</i>	Specifies the name of the client's queue manager.
-cqn <i>queue</i>	Specifies the name of the client's request queue.
-crqm <i>queueManager</i>	Specifies the name of the client's reply queue manager.
-crqn <i>queue</i>	Specifies the name of the client's reply queue.
-cmqn <i>modelQueue</i>	Specifies the name of the client's model queue.
-cus <i>usageStyle</i>	Specifies the value of the client's <code>UsageStyle</code> attribute. Valid values are <code>Peer</code> , <code>Requester</code> , or <code>Responder</code> .
-ccs <i>correlationStyle</i>	Specifies the value of the client's <code>CorrelationStyle</code> attribute. Valid values are <code>messageId</code> , <code>correlationId</code> , or <code>messageId copy</code> .
-cam <i>accessMode</i>	Specifies the value of the client's <code>AccessMode</code> attribute. Valid values are <code>peek</code> , <code>send</code> , <code>receive</code> , <code>receive exclusive</code> , or <code>receive shared</code> .
-cto <i>timeout</i>	Specifies the value of the client's <code>Timeout</code> attribute.
-cme <i>expiry</i>	Specifies the value of the client's <code>MessageExpiry</code> attribute.
-cmp <i>priority</i>	Specifies the value of the client's <code>MessagePriority</code> attribute.
-cmi <i>messageId</i>	Specifies the value of the client's <code>MessageId</code> attribute.

Adding Endpoints

Option	Interpretation
-cci <i>correlationId</i>	Specifies the value of the client's <code>CorrelationId</code> attribute.
-cd <i>delivery</i>	Specifies the value of the client's <code>Delivery</code> attribute.
-ct <i>transactional</i>	Specifies the value of the client's <code>Transactional</code> attribute. Valid values are <code>none</code> , <code>internal</code> , or <code>xa</code> .
-cro <i>reportOption</i>	Specifies the value of the client's <code>ReportOption</code> attribute. Valid values are <code>none</code> , <code>coa</code> , <code>cod</code> , <code>exception</code> , <code>expiration</code> , or <code>discard</code> .
-cf <i>format</i>	Specifies the value of the client's <code>Format</code> attribute.
-cad <i>applicationData</i>	Specifies the value of the client's <code>ApplicationData</code> attribute.
-cat <i>accountingToken</i>	Specifies the value of the client's <code>AccountingToken</code> attribute.
-ccn <i>connectionName</i>	Specifies the name of the connection by which the adapter connects to the queue.
-cc <i>convert</i>	Specifies if the messages in the queue need to be converted to the system's native encoding. Valid values are <code>true</code> or <code>false</code> .
-ccr <i>reusable</i>	Specifies the value of the client's <code>ConnectionReusable</code> attribute. Valid values are <code>true</code> or <code>false</code> .
-ccfp <i>fastPath</i>	Specifies the value of the client's <code>ConnectionFastPath</code> attribute. Valid values are <code>true</code> or <code>false</code> .
-caid <i>idData</i>	Specifies the value of the client's <code>ApplicationIdData</code> attribute.
-caod <i>originData</i>	Specifies the value of the client's <code>ApplicationOriginData</code> attribute.
-caqn <i>queue</i>	Specifies the remote queue to which a server will put replies if its queue manager is not on the same host as the client's local queue manager.
-cui <i>userId</i>	Specifies the value of the client's <code>UserIdentification</code> attribute.
-o <i>file</i>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.
-L <i>file</i>	Specifies the location of your license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-quiet	Specifies that the tool runs in quiet mode.
-verbose	Specifies that the tool runs in verbose mode.

Option	Interpretation
-h	Displays the tool's usage statement.
-v	Displays the tool's version.
-d <i>dir</i>	Specifies the output directory for the generated contract.
<i>wSDLurl</i>	Specifies the name of the WSDL file to process.

Name

`wsdltoservice -transport tibrv` — generates a WSDL document containing an Artix ESB C++ Runtime Tibco Rendezvous endpoint

Synopsis

```
wsdltoservice -transport tibrv [-e service] [-t port] [-b binding]
[-tss subject] [-tcst subject] [-tbt bindingType] [-tcl callbackLevel]
[-trdt timeout] [-tts transportService] [-ttn transportNetwork] [-ttbm
batchMode] [-tqp priority] [-tqlp queueLimitPolicy] [-tqme
queueMaxEvents] [-tqda queueDiscardAmount] [-tcs cmSupport] [-tctsn
cmTransportServerName] [-tctcn cmTransportClientName] [-tctro
cmTransportRequestOld] [-tctln cmTransportLedgerName] [-tctsl
cmTransportSyncLedger] [-tctra cmTransportRelayAgent] [-tctdtl
cmTransportDefaultTimeLimit] [-tclca cmListenerCancelAgreements]
[-tcqtsn cmQueueTransportServerName] [-tcqtcn
cmQueueTransportClientName] [-tcqqtww
cmQueueTransportWorkerWeight] [-tcqqtsw
cmQueueTransportWorkerTasks] [-tcqqtsw
cmQueueTransportSchedulerWeight] [-tcqqtsh
cmQueueTransportSchedulerHeartbeat] [-tcqqtsw
cmQueueTransportSchedulerActivation] [-tcqqtct
cmQueueTransportCompleteTime] [-tmnfv messageNameFieldValue]
[-tmnfp messageNameFieldPath] [-tbfi bindingFieldId] [-tbfn
bindingFieldName] [-o file] [-d dir] [-L file] [[-quiet] | [-verbose]] [-h]
[-v] wsdurl
```

Description

wsdltoservice -transport tibrv adds a Artix ESB C++ Runtime Tibco Rendezvous endpoint to a WSDL document based on the values provided as arguments to the tool.

Arguments

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

Option	Interpretation
<code>-e service</code>	Specifies the name of the generated service.
<code>-t port</code>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
<code>-b binding</code>	Specifies the name of the binding for which the service is generated.
<code>-tss subject</code>	Specifies the subject to which the server listens.
<code>-tbt bindingType</code>	Specifies the message binding type. Valid vales are <code>msg</code> , <code>xml</code> , <code>opaque</code> , or <code>string</code> .
<code>-tcl callbackLevel</code>	Specifies the server-side callback level when TIB/RV system advisory messages are received. Valid values are <code>INFO</code> , <code>WARN</code> , or <code>ERROR</code> .
<code>-trdt timeout</code>	Specifies the client-side response receive dispatch time-out.
<code>-tts transportService</code>	Specifies the UDP service name or port for TibrvNetTransport.
<code>-ttn transportNetwork</code>	Specifies the binding network addresses for TibrvNetTransport.
<code>-ttbm batchSize</code>	Specifies if the TIB/RV transport uses batch mode to send messages. Valid values are <code>DEFAULT_BATCH</code> and <code>TIMER_BATCH</code> .
<code>-tqp priority</code>	Specifies the queue priority.
<code>-tqlp queueLimitPolicy</code>	Valid values are <code>DISCARD_NONE</code> , <code>DISCARD_NEW</code> , <code>DISCARD_FIRST</code> , or <code>DISCARD_LAST</code> .
<code>-tqme queueMaxEvents</code>	Specifies the queue max events.
<code>-tqda queueDiscardAmount</code>	Specifies the queue discard amount.
<code>-tcs cmSupport</code>	Specifies if Certified Message Delivery support is enabled. Valid values are <code>true</code> or <code>false</code> .
<code>-tctsn cmTransportServerName</code>	Specifies the server's TibrvCmTransport correspondent name.
<code>-tctcn cmTransportClientName</code>	Specifies the client TibrvCmTransport correspondent name.

Adding Endpoints

Option	Interpretation
<code>-tctro cmTransportRequestOld</code>	Specifies if the endpoint can request old messages on start-up. Valid values are <code>true</code> or <code>false</code> .
<code>-tctlm cmTransportLedgerName</code>	Specifies the TibrvCmTransport ledger file.
<code>-tctsl cmTransportSyncLedger</code>	Specifies if the endpoint uses a synchronous ledger. Valid values are <code>true</code> or <code>false</code> .
<code>-tctra cmTransportRelayAgent</code>	Specifies the endpoint's TibrvCmTransport relay agent.
<code>-tctdtl cmTransportDefaultTimeLimit</code>	Specifies the default time limit for a Certified Message to be delivered.
<code>-tclca cmListenerCancelAgreements</code>	Specifies if Certified Message agreements are canceled when the endpoint disconnects. Valid values are <code>true</code> or <code>false</code> .
<code>-tcqtsn cmQueueTransportServerName</code>	Specifies the server's TibrvCmQueueTransport correspondent name.
<code>-tcqtcn cmQueueTransportClientName</code>	Specifies the client's TibrvCmQueueTransport correspondent name.
<code>-tcqtwc cmQueueTransportWorkerWeight</code>	Specifies the endpoint's TibrvCmQueueTransport <code>worker weight</code> parameter.
<code>-tcqtws cmQueueTransportWorkerTasks</code>	Specifies the endpoint's TibrvCmQueueTransport <code>worker tasks</code> parameter.
<code>-tcqtsw cmQueueTransportSchedulerWeight</code>	Specifies the TibrvCmQueueTransport <code>scheduler weight</code> parameter.
<code>-tcqtsh cmQueueTransportSchedulerHeartbeat</code>	Specifies the endpoint's TibrvCmQueueTransport <code>scheduler heartbeat</code> parameter.
<code>-tcqtsa cmQueueTransportSchedulerActivation</code>	Specifies the TibrvCmQueueTransport <code>scheduler activation</code> parameter.
<code>-tcqtct cmQueueTransportCompleteTime</code>	Specifies the TibrvCmQueueTransport <code>complete time</code> parameter.
<code>-tmnfv messageNameFieldValue</code>	Specifies the message name field value.
<code>-tmnfp messageNameFieldPath</code>	Specifies the message name field path.
<code>-tbfi bindingFieldId</code>	Specifies the binding field id.
<code>-tbfn bindingFieldName</code>	Specifies the binding field name.
<code>-o file</code>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.

Option	Interpretation
<code>-d <i>dir</i></code>	Specifies the output directory for the generated contract.
<code>-L <i>file</i></code>	Specifies the location of your license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-quiet</code>	Specifies that the tool runs in quiet mode.
<code>-verbose</code>	Specifies that the tool runs in verbose mode.
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.
<code><i>wSDLurl</i></code>	Specifies the name of the WSDL file to process.

Name

`wsdltoservice -transport tuxedo` — generates a WSDL document containing an Artix ESB C++ Runtime Tuxedo endpoint

Synopsis

```
wsdltoservice -transport tuxedo [-e service] [-t port] [-b binding]
[-tsn tuxService] [-tfn tuxService:tuxFunction] [-ton
tuxService:operation] [-o file] [-d dir] [-L file] [[-quiet] | [-verbose]]
[-h] [-v] wsdurl
```

Description

wsdltoservice -transport tuxedo adds a Artix ESB C++ Runtime Tuxedo endpoint to a WSDL document based on the values provided as arguments to the tool.

Arguments

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

Option	Interpretation
<code>-e service</code>	Specifies the name of the generated service.
<code>-t port</code>	Specifies the value of the <code>name</code> attribute of the generated <code>port</code> element.
<code>-b binding</code>	Specifies the name of the binding for which the service is generated.
<code>-tsn tuxService</code>	Specifies the name the service uses to register with the Tuxedo bulletin board.
<code>-tfn tuxService:tuxFunction</code>	Specifies the name of the function to be used on the specified Tuxedo bulletin board.
<code>-ton tuxService:operation</code>	Specifies the WSDL operation that is handled by the specified Tuxedo endpoint.
<code>-o file</code>	Specifies the filename for the generated contract. The default is to append <code>-service</code> to the name of the imported contract.
<code>-d dir</code>	Specifies the output directory for the generated contract.

Option	Interpretation
-L <i>file</i>	Specifies the location of your license file. The default behavior is to check IT_PRODUCT_DIR\etc\license.txt.
-quiet	Specifies that the tool runs in quiet mode.
-verbose	Specifies that the tool runs in verbose mode.
-h	Displays the tool's usage statement.
-v	Displays the tool's version.
<i>wSDLurl</i>	Specifies the name of the WSDL file to process.

Adding Routes

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

wsdltorouting 78

Name

`wsdltorouting` — adds a route to a WSDL document

Synopsis

```
wsdltorouting [-rn name] [-ssn service] [-spn port] [-dsn service]
[-dpn port] [-on operation] [-ta attribute] [-d dir] [-o file] [-L file]
[[-quiet] | [-verbose]] [-h] [-v] {wSDL}
```

Description

wsdltorouting adds a route to the provided WSDL document. Routes are used by the Artix ESB router to direct messages between endpoints. For more information see [Router Guide](#) [./routing/index.htm].

Arguments

The arguments for controlling the generated route are reviewed in the following table.

Option	Interpretation
<code>-rn <i>name</i></code>	Specifies the name of the generated route. If no name is given a unique name will be generated for the route.
<code>-ssn <i>service</i></code>	Specifies the name of the service to use as the source of the route.
<code>-spn <i>port</i></code>	Specifies the name of the port to use as the source of the route. The port must correspond to a <code>port</code> element in the specified service.
<code>-dsn <i>service</i></code>	Specifies the name of the service to use as the destination of the route.
<code>-dpn <i>port</i></code>	Specifies the name of the port to use as the destination of the route. The port must correspond to a <code>port</code> element in the specified service.
<code>-on <i>operation</i></code>	Specifies the name of the operation to use for the route. If the route is port-based, you do not need to use this flag.
<code>-ta <i>attribute</i></code>	Specifies a transport attribute to use in defining the route.
<code>-d <i>dir</i></code>	Specifies the output directory for the generated contract.
<code>-o <i>file</i></code>	Specifies the filename for the generated contract.

Option	Interpretation
<code>-L file</code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.
<code>-quiet</code>	Specifies that the tool is to run in quiet mode.
<code>-verbose</code>	Specifies that the tool is to run in verbose mode.
<code>wSDL</code>	Specifies the name of the WSDL document to which the route is added.

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.

artix validator	82
schemavalidator	83

Name

artix validator — validates a WSDL document

Synopsis

```
artix validator [-s schema-url...] [-w] [-v] [[-verbose] | [-quiet]] wSDLurl
```

Description

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

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<i>wSDLurl</i>	Specifies the WSDL file to validate.

Optional Arguments

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

Option	Interpretation
<code>-s <i>schema-url</i></code>	Specifies the URL of a user specific schema to be included in the validation of the contract. This switch can appear multiple times.
<code>-w</code>	Specifies that warnings are suppressed.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the validation.
<code>-quiet</code>	Suppresses comments during the validation.

Using Ant

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

Name

schemavalidator — validates WSDL documents and checks if they meet the WS-I basic profile

Synopsis

```
schemavalidator [ -d schema-directory ... ] [ -s schema-url ... ] { -w  
WSDL_XSD_URL } [ -deep ] [ -wsi ] [ -wh ws-test-tools.home ] [ -tad  
BasicProfileAssertions ] [ -L file ] [[-quiet] | [-verbose]] [ -h ] [  
-v ]
```

Description

schemavalidator validates that a WSDL document is well-formed. In addition, it can test the WSDL document for conformance to the WS-I basic profile.

Arguments

The arguments used to manage WSDL validation are described below.

Argument	Interpretation
-d <i>schema-directory</i>	Specifies the directory used to search for schemas. This switch can appear multiple times.
-s <i>schema-url</i>	Specifies the URL of a user specific schema to be included in the validation of the contract. This switch can appear multiple times.
-w <i>WSDL_XSD_URL</i>	Specifies the URL of the document to be validated.
-deep	Specifies that the validator is to check all WSDL imports and all WSDL semantics. When using this switch, the tool will also validate the imported WSDL.
-wsi	Specifies that the tool is to use the wsi-test-tools from wsi.org to validate the contract.
-wh <i>wsi-test-tools.home</i>	Specifies the base directory of wsi-test-tools.
-tad <i>BasicProfileAssertions</i>	Specifies the URL of the of <i>BasicProfileTestAssertions.xml</i> used in wsi-test-tools.
-L <i>file</i>	Specifies the location of your Artix license file. The default behavior is to check <i>IT_PRODUCT_DIR\etc\license.txt</i> .

Argument	Interpretation
-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.

Transforming XML

Artix includes a command line driven XSLT processor for transforming XML documents.

xslttransform 86

Name

`xslttransform` — transforms an XML document based on an XSLT stylesheet

Synopsis

```
xslttransform {-IN inputXMLURL} {-OUT outputXMLURL} {-XS XSLTURL}
[-PARAM name value...]
```

Description

xslttransform transforms an XML document based on an XSLT stylesheet. The command uses the Artix ESB transformer which is implemented as part of the Artix ESB C++ Runtime. To use it you must source the **artix_env** script located in `InstallDirCxx_java/bin`.

Arguments

The arguments for controlling the transformation are reviewed in the following table.

Option	Interpretation
<code>-IN <i>inputXMLURL</i></code>	Specifies the URL of the source XML document.
<code>-OUT <i>outputXMLURL</i></code>	Specifies the URL of the transformed XML document.
<code>-XS <i>XSLTURL</i></code>	Specifies the URL of the XSLT stylesheet.
<code>-PARAM <i>name value</i></code>	Specifies a name/value pair that corresponds to a parameter in the XSLT stylesheet.

Generating Code from WSDL

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

artix wsdlgen	88
artix wsdl2cpp	90
artix wsdl2java	94
wsdltojava	99
artix wsdl2dbservice	102
wsdltodbservice	105

Name

artix wsdlgen — generates application code based on JavaScript templates

Synopsis

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

Description

artix 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 C++ and Java code if you do not require any customization.

For more information see [WSDLGen Guide](#) [../wsdlgen/index.htm].

Arguments

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

Option	Interpretation
-G <i>ApplicationType</i>	Specifies the type of application to generate. The following application types are defined by default: <ul style="list-style-type: none"> • <i>cxx</i>—for generating C++ code • <i>jaxrpc</i>—for generating JAX-RPC code • <i>jaxws</i>—for generator JAX-WS code
-T <i>TemplateID</i>	Specifies the template ID that governs code generation. See Template IDs on page 89 for details.
-C <i>ConfigFile</i>	Specifies the location of a configuration file to be used by the code generator.
-D <i>name=value</i>	Specifies the value, <i>value</i> , of a JavaScript property, <i>name</i> . Typically you will use this option to specify a value for the portType property. This instructs the code generator the WSDL <code>portType</code> element for which code is to be generated.

Option	Interpretation
<code>WSDLFile</code>	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
<code>impl</code>	Generate the stub and skeleton code require to implement the interface defined by the specified WSDL <code>portType</code> element.
<code>server</code>	Generate a simple <code>main()</code> for a standalone service that will host an implementation of the interface defined by the specified WSDL <code>portType</code> element. Stub code is also generated.
<code>client</code>	Generate a C++ file or Java class that invokes all of the operations defined by the specified WSDL <code>portType</code> element. Stub code is also generated.
<code>plugin</code>	If generating C++ or JAX-RPC code, generate all of the code needed to implement the interface defined by the specified WSDL <code>portType</code> element as an Artix plug-in.
<code>all</code>	For C++ and JAX-RPC, generate a client, a server, and an Artix plug-in. For JAX-WS, generate a client and a server. For example, specifying <code>-G cxx -T all</code> is equivalent to <code>-G cxx -T impl -T plugin -T client</code> .
<code>ant</code>	Generate an Apache Ant build file for a Java application.
<code>make</code>	Generate a make file for a C++ application.

Name

`artix wsdl2cpp` — generates C++ stubs and skeletons for the services defined in a WSDL document

Synopsis

```
artix wsdl2cpp [-e web_service_name[:port_list]] [-b binding_name]
[-i port_type...] [-d output-dir] [-n URI=C++namespace...] [-nexclude
URI=C++namespace...] [-ninclude URI=C++namespace...] [-nimport
C++namespace] [-impl] [-m { NMAKE | UNIX } : [ executable | library ]] [-libv
ersion] [-jp plugin_class] [-f] [-server] [-client] [-sample]
[-plugin[:plugin_name]] [-deployable] [-global] [-license] [-declspec
declspec] [-all] [-flags] [[-upper] | [-lower] | [-minimal] | [-mapper class]]
[-reflect] [-user_reserved_words word1 [:wordn...]] [-L file] [[-quiet] |
[-verbose]] [-h] [-v] wsdlurl
```

Description

artix wsdl2cpp generates C++ skeletons for the services defined in a WSDL document. It can also generate starting point code for your server and client applications.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>wsdlurl</code>	The WSDL document from which the code is generated.

Optional Arguments

The tool uses the following optional arguments:

Option	Interpretation
<code>-i port_type</code>	Specifies the name of the port type for which the tool will generate code. The default is to use the first port type listed in the contract. This switch can appear multiple times.
<code>-e web_service_name[:port_list]</code>	Specifies the name of the service for which the tool will generate code. The default is to use the first service listed in the contract. You can optionally specify a comma separated list of port names to activate. The default is to activate all of the service's ports.
<code>-b binding_name</code>	Specifies the name of the binding to use when generating code. The default is the first binding listed in the contract.
<code>-d output_dir</code>	Specifies the directory to which the generated code is written. The default is the current working directory.
<code>-n [URI=]C++namespace</code>	Maps an XML namespace to a C++ namespace. The C++ stub code generated from the XML namespace (<i>URI</i>) is put into the specified C++ namespace. This switch can appear multiple times.
<code>-nexclude URI[=C++namespace]</code>	Do not generate C++ stub code for the specified XML namespace. You can optionally map the XML namespace to a C++ namespace in case it is referenced by the rest of the XML Schema/WSDL document. This switch can appear multiple times.
<code>-nininclude URI[=C++namespace]</code>	Generates C++ stub code for the specified XML namespace. You can optionally map the XML namespace to a C++ namespace. This switch can appear multiple times.
<code>-nimport C++namespace</code>	Specifies the C++ namespace to use for the code generated from imported schema.
<code>-impl</code>	Generates the skeleton code for implementing the server defined by the contract.
<code>-m {NMAKE UNIX}: [executable library]</code>	Used in combination with <code>-impl</code> to generate a makefile for the specified platform (<code>NMAKE</code> for Windows or <code>UNIX</code> for UNIX). You can specify that the generated makefile builds an executable, by appending <code>:executable</code> , or a library, by appending <code>:library</code> .
<code>-libv version</code>	Used in combination with either <code>-m NAME:library</code> or <code>-m UNIX:library</code> to specify the version number of the library built by the makefile. This version number is for your own convenience, to help you keep track of your own library versions.
<code>-f</code>	<i>Deprecated</i> —Was needed to support routing in earlier versions.

Option	Interpretation
-server	Generates code for a sample implementation of a server.
-client	Generates code for a sample implementation of a client.
-sample	Generates code for a sample implementation of a client and a server (equivalent to <code>-server -client</code>).
-plugin[: <i>plugin_name</i>]	Generates servant registration code as a bus plug-in. You can optionally specify the plug-in name by appending <code>:<i>plugin_name</i></code> to this option. If no plug-in name is specified, the default name is <code>ServiceNamePortTypeName</code> . The service name is specified by the <code>-e</code> option.
-deployable	(Used with <code>-plugin</code> .) Generates a deployment descriptor file, <code>deployServiceName.xml</code> , which is needed to deploy a plug-in into the Artix ESB C++ Runtime container.
-global	(Used with <code>-plugin</code> .) In the generated plug-in code, instantiate the plug-in using a <code>GlobalBusORBPlugIn</code> object instead of a <code>BusORBPlugIn</code> object. A <code>GlobalBusORBPlugIn</code> initializes the plug-in automatically, as soon as it is constructed (suitable approach for plug-ins that are linked directly with application code). A <code>BusORBPlugIn</code> is not initialized unless the plug-in is either listed in the <code>orb_plugins</code> list or deployed into an Artix ESB C++ Runtime container (suitable approach for dynamically loading plug-ins).
-license	Displays the currently available licenses.
-declspec <i>declspec</i>	Creates Visual C++ declaration specifiers for <code>dllexport</code> and <code>dllimport</code> . This option makes it easier to package Artix stubs in a DLL library.
-all	Generate stub code for all of the port types and the types that they use. This option is useful when multiple port types are defined in a WSDL contract.
-flags	Displays detailed information about the options.
-reflect	Enables reflection on the generated classes.
-wrapped	When used with document/literal wrapped style, generates function signatures with wrapped parameters, instead of unwrapping into separate parameters.

Option	Interpretation
<code>-user_reserved_words</code> <code>word1[:wordn...]</code>	Specifies a colon-separated list of words to be treated as reserved. For example, <code>-user_reserved_words SEC:MILLISEC</code> would generate a header file including 'class _SEC' instead of 'class SEC'.
<code>-L file</code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.
<code>-quiet</code>	Suppresses comments during the code generation process.

Name

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

Synopsis

```
artix 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] [-xCargs] [-noAddressBinding]
[-validate] [-v] [[-verbose] | [-quiet]] wsdlfile
```

Description

artix 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, **artix 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	

Option	Interpretation
<code>-fe <i>frontend</i></code>	Specifies the front end used by the code generator. The default is <code>jaxws</code> . ^a
<code>-db <i>databinding</i></code>	Specifies the data binding used by the code generator. The default is <code>jaxb</code> . ^b
<code>-wv <i>wSDLVersion</i></code>	Specifies the WSDL version expected by the tool. The default is <code>1.1</code> . ^c
<code>-p [<i>wSDLNamespace=</i>]<i>PackageName</i></code>	Specifies zero, or more, package names to use for the generated code. Optionally specifies the WSDL namespace to package name mapping.
<code>-b <i>bindingName</i></code>	Specifies zero, or more, JAXWS or JAXB binding files. Use spaces to separate multiple entries.
<code>-sn <i>serviceName</i></code>	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.
<code>-d <i>output-directory</i></code>	Specifies the directory into which the generated code files are written.
<code>-catalog <i>catalogUrl</i></code>	Specifies the URL of an XML catalog to use for resolving imported schemas and WSDL documents.
<code>-compile</code>	Compiles generated Java files.
<code>-classdir <i>compile-class-dir</i></code>	Specifies the directory into which the compiled class files are written.

Option	Interpretation
-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 <code>build.xml</code> file.
-ant	Generates the Ant <code>build.xml</code> file.
-keep	Instructs the tool to not overwrite any existing files.
-defaultValues[= <i>DefaultValueProvider</i>]	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 <code>RandomValueProvider</code> class is used.
-nexclude <i>schema-namespace</i> [= <i>java-packagename</i>]	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 extended soap

Option	Interpretation
	header message binding. Default is <code>false</code> .
<code>-dns (true/false)</code>	Enables or disables the loading of the default namespace package name mapping. Default is <code>true</code> .
<code>-dex (true/false)</code>	Enables or disables the loading of the default excludes namespace mapping. Default is <code>true</code> .
<code>-wsdlLocation wsdlLocation</code>	Specifies the value of the <code>@WebService</code> annotation's <code>wsdlLocation</code> property.
<code>-xjcargs</code>	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 <code>-xjc-X</code> .
<code>-noAddressBinding</code>	Instructs the tool to use the Artix ESB proprietary WS-Addressing type instead of the JAX-WS 2.1 compliant mapping.
<code>-validate</code>	Instructs the tool to validate the WSDL document before attempting to generate any code.
<code>-v</code>	Displays the version number for the tool.
<code>-verbose</code>	Displays comments during the code generation process.

Option	Interpretation
<code>-quiet</code>	Suppresses comments during the code generation process.
<code>wSDLfile</code>	The path and name of the WSDL file to use in generating the code.

^aCurrently, Artix ESB only provides the JAX-WS front end for the code generator.

^bCurrently, Artix ESB only provides the JAXB data binding for the code generator.

^cCurrently, 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.wsdltto.WSDLToJava`.

[Example 7 on page 98](#) shows the **java** task to execute this command.

Example 7. Generating a Java Code From Ant

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

Name

`wSDLtojava` — generates JAX-RPC compliant Java code stubs and skeletons for the services defined in a WSDL document

Synopsis

```
wSDLtojava [-e service:port...] [-b binding] [-i portType] [-d  
output_dir] [-p [namespace=]package] [-impl] [-server] [-client] [-plugin]  
[-servlet] [-types] [-call] [-interface] [-sample] [-all] [-ant] [-datahandlers]  
[-merge] [-deployable] [-nextclude namespace [=package]] [-ninclude  
namespace[=package]] [-ser] [-stub] [-typehandlers] [-L file] [[-quiet] |  
[-verbose]] [-h] [-v] wSDLurl
```

Description

wSDLtojava generates JAX-RPC compliant Java code stubs and skeletons for the services defined in the specified WSDL document. It can also generate starting point code for your server and client applications. The default behavior of **wSDLtojava** is to generate all of the Java code needed to develop a client and server.



Note

The JAX-RPC APIs are implemented as part of the Artix ESB C++ Runtime using a JNI layer.

Required Arguments


The tool has the following required attributes:

Option	Interpretation
<code>wSDLurl</code>	Specifies the WSDL document for which the code is generated.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-e service:port</code>	Specifies the name of the service, and optionally the port, for which the tool will generate code. The default is to use the first service listed in the contract. Specifying multiple services results in the generation of code for all the named service/port combinations. If no port is given, all ports defined in a service will be activated.
<code>-b binding</code>	Specifies the name of the binding to use when generating code. The default is to use the first binding listed in the contract.
<code>-i portType</code>	Specifies the name of a portType for which code will be generated. You can specify this flag for each portType for which you want code generated. The default is to use the first portType in the contract.
<code>-d out_dir</code>	Specifies the directory to which the generated code is written. The default is the current working directory.
<code>-p</code> <code>[namespace=]package</code>	Specifies the name of the Java package to use for the generated code. You can optionally map a WSDL namespace to a particular package name if your contract has more than one namespace.
<code>-impl</code>	Generates the skeleton class for implementing the server defined by the contract.
<code>-server</code>	Generates a simple main class for the server.
<code>-client</code>	Generates only the Java interface and code needed to implement the complex types defined by the contract. This flag is equivalent to specifying <code>-interface -types</code> .
<code>-plugin</code>	Generate a bus plug-in with the appropriate servant registration code for the generated service implementation.
<code>-servlet</code>	Generates a bus plug-in with the additional information needed to deploy it as a servlet.
<code>-types</code>	Generates the code to implement the complex types defined by the contract.
<code>-call</code>	Generates a sample client the uses the <code>Call</code> interface to invoke on the remote service.
<code>-interface</code>	Generates the Java interface for the service.
<code>-sample</code>	Generates a sample client that can be used to test your Java server.
<code>-all</code>	Generates code for all portTypes in the contract.
<code>-ant</code>	Generate an ant build target for the generated code.
<code>-datahandlers</code>	When a service uses SOAP w/ attachments as its payload format, generate code that uses <code>javax.activation.DataHandler</code> instead of the standard Java classes specified in the JAX-RPC specification.
<code>-merge</code>	Merge any user changes into the generated code.

Option	Interpretation
-deployable	Generate a deployment descriptor to deploy the generated plug-in into an Artix ESB C++ Runtime container.
-nexclude <i>namespace[=package]</i>	Instructs the code generator to skip the specified XML Schema namespace when generating code. You can optionally specify a package name to use for the types that are not generated.
-ninclude <i>namespace[=package]</i>	Instructs the code generator to generate code for the specified XML Schema namespace. You can optionally specify a package name to use for the types in the specified namespace.
-ser	Specifies that the generated classes for the types defined in a contract should be serializable.
-stub	Specifies that the tool will generate the stub code for a client and a server.
-typehandlers	<p>Specifies that the tool will generate C++ style type handlers for complex types.</p> <p> Warning</p> <p>Not all types are supported by the type handler mechanism.</p> <p>Type handlers are incompatible with standard JAX-WS code in some cases.</p>
-L <i>file</i>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-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

`artix wsdl2dbservice` — generates an Artix ESB Java Runtime intermediary for a database service

Synopsis

```
artix wsdl2dbservice [-jdbctypemappings jdbc-type-mapping-file]
[-db data binding name...] [-wv wsdl version...] [-p [wsdl namespace=]Package Name...] [-sn [wsdl namespace=]Package Name]
[-b binding-name...] [-d output-directory] [-compile] [-classdir compile-classes-directory] [-impl] [-server] [-client] [-all] [-ant]
[-nexclude schema namespace [= java packagename]...] [-exsh (true|false)]
[-dns (true|false)] [-dex (true|false)] [-validate] [-wsdlLocation wsdlLocation attribute] [-v] [[-verbose] | [-quiet]] wsdlurl
```

Description

artix wsdl2dbservice takes a WSDL document and generates the code for a Artix ESB Java Runtime database service intermediary.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<i>wsdlurl</i>	Specifies the WSDL document for which the database service is generated.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-jdbctypemappings</code> <i>jdbc-type-mapping-file</i>	Specifies the location of the JDBC to XSD mapping file.
<code>-db <i>data binding name</i></code>	Specifies the data binding to use. The default is JAXB.

Option	Interpretation
<code>-wv wsdl version</code>	Specifies the WSDL version to use. The default is WSDL 1.1.
<code>-p [wsdl namespace=]Package Name</code>	Specifies the Java package name to use for the generated code. Optionally, you can specify the WSDL namespace mapping to a particular Java package name.
<code>-sn [wsdl namespace=]Package Name</code>	Specifies the service name to use for the generated code. Optionally, you can specify the WSDL namespace.
<code>-b binding-name</code>	Specifies an external JAXWS or JAXB binding files.
<code>-d output-directory</code>	Specifies the directory into which the generated code is placed.
<code>-compile</code>	Specifies that the generated code is compiled.
<code>-classdir compile-classes-directory</code>	Specifies the directory into which the compiled class files are placed.
<code>-impl</code>	Generates a dummy implementation class.
<code>-server</code>	Generates a server mainline for the service.
<code>-client</code>	Generates the code needed to deploy a client.
<code>all</code>	Generates all starting point code: types, service proxy, service interface, server mainline, client mainline, implementation object, and an Ant <code>build.xml</code> file.
<code>-ant</code>	Generates an Ant <code>build.xml</code> .
<code>-nexclude schema-namespace [=java-packagename]</code>	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).
<code>-exsh (true/false)</code>	Enables or disables processing of extended soap header message binding.
<code>-dns (true/false)</code>	Enables or disables the loading of the default namespace package name mapping. Default is <code>true</code> .
<code>-dex (true/false)</code>	Enables or disables the loading of the default excludes namespace mapping. Default is <code>true</code> .
<code>-validate</code>	Enables validating the WSDL before generating the code.
<code>-v</code>	Display's the tool's version.

Option	Interpretation
-quiet	Specifies that the tool suppresses most messages.
-verbose	Specifies that the tool displays verbose messages.

Name

`wsdltodbservice` — generates an Artix ESB C++ Runtime intermediary for a database service

Synopsis

```
wsdltodbservice [-d dir] [-source dir] [-plugin] [-h] [-v] [[-quiet] |  
[-verbose]] dbconfig wSDLurl
```

Description

wsdltodbservice takes a WSDL document and an Artix ESB C++ Runtime database configuration document and generates the code for the intermediary used expose the database operations. The generated Java code will need to be compiled before it can be deployed.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<code>dbconfig</code>	Specifies the name of the database configuration file to use when generating code.
<code>wSDLurl</code>	Specifies the WSDL document to use when generating code.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-d <i>dir</i></code>	Specifies the output directory for the generated DB service.
<code>-source <i>dir</i></code>	Specifies the output directory for the generated source code. The default is <code>java</code> .
<code>-plugin</code>	Specifies that the DB service is to be generated as a plug-in for deployment into an Artix ESB C++ Runtime container.
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.

Option	Interpretation
-quiet	Specifies that the tool is to run in quiet mode.
-verbose	Specifies that the tool is to run in verbose mode.

Generating Support Files

Artix provides tools to generate a number of support files.

artix wsdl2idl -idl	108
wsdltoCorba -idl	110
artix sql2dbconfig	112
wstd	117
artix wsdl2acl	119

Name

`artix wsdl2idl -idl` — generates an IDL file from a WSDL document containing an Artix ESB Java Runtime CORBA binding

Synopsis

```
artix wsdl2idl {-idl} {-b binding} [-corba] [-i portType] [-d dir] [-w
wsdlOut] [-o idlOut...] [-props namespace] [-wrapped] [-a address] [-f
address-file] [-L file] [[-quiet] | [-verbose]] [-v] [-h] wsdl
```

Description

artix wsdl2idl -idl generates an IDL file from a WSDL document containing an Artix ESB Java Runtime 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
<code>-idl</code>	Specifies that the tool is to generate IDL from the binding.
<code>-b <i>binding</i></code>	Specifies the name of the CORBA binding for which the IDL file is generated.
<code><i>wsdl</i></code>	Specifies the WSDL document to which the binding is added.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-corba</code>	Specifies that an CORBA binding will be added to the WSDL document. You must also use the <code>-i</code> flag in conjunction with this flag.
<code>-i <i>portType</i></code>	Specifies the name of the port type for which the CORBA binding is generated.
<code>-d <i>dir</i></code>	Specifies the directory into which the new IDL file is written.

Option	Interpretation
<code>-w wsdlOut</code>	Specifies the name of the WSDL document containing the generated CORBA binding.
<code>-o idlOut</code>	Specifies the name of the generated IDL file.
<code>-props namespace</code>	Specifies the namespace to use for the generated CORBA typemap.
<code>-wrapped</code>	Specifies that the generated binding uses wrapped types.
<code>-a address</code>	Specifies the value of the generated binding's <code>corba:address</code> element's <code>location</code> attribute.
<code>-f address-file</code>	Specifies the name of a file whose contents are to be used as the value of the generated binding's <code>corba:address</code> element's <code>location</code> attribute.
<code>-L file</code>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-v</code>	Displays the tool's version.
<code>-h</code>	Specifies that the tool will display a detailed usage statement.
<code>-quiet</code>	Specifies that the tool is to run in quiet mode.
<code>-verbose</code>	Specifies that the tool is to run in verbose mode.

Name

`wsdltocorba -idl` — generates an IDL file from a WSDL document containing an Artix ESB C++ Runtime CORBA binding

Synopsis

```
wsdltocorba -idl {-b binding} [-corba] [-i portType] [-d dir] [-o file]
[-L file] [[-quiet] | [-verbose]] [-h] [-v] wSDL
```

Description

wsdltocorba -idl generates an IDL file from a WSDL document containing a Artix ESB C++ Runtime CORBA binding.

Required Arguments

The required arguments for generating an IDL file are reviewed in the following table.

Option	Interpretation
<code>-b <i>binding</i></code>	Specifies the name of the CORBA binding for which the IDL is generated.
<code><i>wSDL</i></code>	Specifies the WSDL document to which the binding is added.

Optional Arguments

The optional arguments used to control the generated CORBA binding are explained in the following table.

Option	Interpretation
<code>-corba</code>	Specifies that a CORBA binding is to be generated.
<code>-i <i>portType</i></code>	Specifies the name of the port type for which the CORBA binding is generated.
<code>-d <i>dir</i></code>	Specifies the directory into which the new WSDL document is written.
<code>-o <i>file</i></code>	Specifies the name of the generated WSDL document. The default is <code><i>wSDL_file</i>-corba.wSDL</code> .
<code>-props <i>namespace</i></code>	Specifies the namespace to use for the generated CORBA typemap.

Option	Interpretation
-wrapped	Specifies that the generated binding uses wrapped types.
-L <i>file</i>	Specifies the location of your Artix license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
-h	Displays the tool's usage statement.
-v	Displays the tool's version.
-quiet	Specifies that the tool is to run in quiet mode.
-versbose	Specifies that the tool is to run in verbose mode.

Name

artix sql2dbconfig — generates a Artix ESB Java Runtime database service configuration file

Synopsis

```
artix sql2dbconfig [-d output-dir] [-new [-driver driver-class
] [-connectionurl connection-url] [-property property ...] [-pool
] [-maxactive pool-maxactive] [-maxidle pool-maxidle] [-
transaction transaction-level] [-autocommit auto-commit] [-
readonly read-only]] [-test] [-add [-name name] [-query query
] [-isprocedure] [-isupdate] [-oktoexecute] [-parametertype
parameter-type ...] [-parameterdirection parameter-direction ...]
[-parameternullable parameter-nullable ...] [-parametervalue
parameter-value ...] [-timeout timeout]] [-delete [-name name]]
[-v] [[-verbose] | [-quiet]] outfile
```



Important

Before running this command you should set the `JDBC_DRIVER_CP` environment variable to point to all required JDBC drivers.

Required Arguments

The tool has the following required arguments:

Option	Interpretation
<i>outfile</i>	Specifies the path and the name of the generated database service configuration file. The default is <code>./dbconfig.xml.db</code> .

Optional Arguments

The the tool has the following optional arguments:

Option	Interpretation
-d <i>output-directory</i>	Specifies the directory into which the generated configuration file is placed.
-new	<p>Specifies that a new configuration file is to be generated. Only connection information will be added to the newly created configuration file. Users can add operations to the configuration file in subsequent commands by specifying the <code>-add</code> option. The target output file should not exist in the filesystem. The following options can be used with the <code>new</code> option:</p> <ul style="list-style-type: none"> • <code>-driver</code> • <code>-connectionurl</code> • <code>-property</code> • <code>-pool</code> • <code>-maxactive</code> • <code>-maxidle</code> • <code>-transaction</code> • <code>-autocommit</code> • <code>-readonly</code>
-driver <i>driver-class</i>	Specifies the driver class for the new connection. This option can only be used with <code>-new</code> .
-connectionurl <i>connection-url</i>	Specifies the connection url for the new connection. This option can only be used with <code>-new</code> .
-property <i>property</i>	Specifies a connection property for the new connection. This option can only be used with <code>-new</code> .
--pool	Specifies that connection pooling should be enabled. This option can only be used with <code>-new</code> .
-maxactive <i>pool-maxactive</i>	Specifies the maximum active connections in the connection pool. This option can only be used when both <code>-new</code> and <code>-pool</code> are specified.

Option	Interpretation
-maxidle <i>pool-maxidle</i>	Specifies the maximum idle connections in the pool. This option can only be used when both <i>-new</i> and <i>-pool</i> are specified.
-transaction <i>transaction-level</i>	Specifies the transaction isolation level for the new connection. This option can only be used with <i>-new</i> . The value should be an integer value as defined in JDBC specification or one of the following: <i>none</i> , <i>read_committed</i> , <i>read_uncommitted</i> , <i>repeatable_read</i> , and <i>serializable</i> .
-autocommit <i>auto-commit</i>	Specifies the auto commit value for the connection. This option can only be used with <i>-new</i> .
-readonly <i>read-only</i>	Specifies the read only value for the connection. This option can only be used with <i>-new</i> .
-test	Test a connection using the connection information provided in an configuration.
-add	<p>Specified when adding a new operation to an existing database service configuration. This option cannot be used with <i>-new</i>. The following options can be used with <i>-add</i>:</p> <ul style="list-style-type: none"> • <i>-name</i> • <i>-query</i> • <i>-isprocedure</i> • <i>-isupdate</i> • <i>-oktoexecute</i> • <i>-parametertype</i> • <i>-parameterdirection</i> • <i>-parameternullable</i> • <i>-parametervalue</i> • <i>-timeout</i>

Option	Interpretation
<code>-name name</code>	Specifies the name of the operation.
<code>-query query</code>	Specifies the query or procedure call of the operation.
<code>-isprocedure</code>	Specifies that the operation is a stored procedure.
<code>-isupdate</code>	Specifies that the operation will write to the database. If both <code>-isupdate</code> and <code>-isprocedure</code> are specified, then <code>-oktoexecute</code> must be specified. If the operation requires IN/INOUT parameters, users must provide parameter values for executing the operation. The reason is that a stored procedure can return multiple results. The command will need to execute the stored procedure to obtain the result metadata. However, if only <code>-isupdate</code> is specified, the command will not need to execute the operation and the result is assumed to be an update count.
<code>-oktoexecute</code>	Specifies that it is OK to execute the operation to get resultset metadata.
<code>-parametertype parameter-type</code>	Specifies the JDBC type for a parameter by position. Positions start at 1. The value should be the integer or string value of the JDBC type as defined in the JDBC specification. For example <code>-parametertype 1=3</code> specifies that the first parameter is a JDBC decimal.
<code>-parameterdirection parameter-direction</code>	Specifies the direction for a parameter by position. Positions start at 1. The value should be one of IN, INOUT, and OUT. For example, <code>-parameterdirection 2=IN</code> specifies that the second parameter is an input parameter.
<code>-parameternullable parameter-nullable</code>	Specifies whether a parameter is nullable by position. Positions start at 1. For example, <code>-parameternullable 1=true</code> specifies that the first parameter can be null.
<code>-parametervalue parameter-value</code>	Specifies a parameter's value for by position. Positions start at 1. This option is necessary for parameterized operation when <code>-oktoexecute</code> is used. For example, <code>-parametervalue 1=12.0</code> specifies the first parameter's value is 12.0.
<code>-timeout timeout</code>	Specifies the number of seconds before the operation times out.

Generating Support Files

Option	Interpretation
-delete	Specifies that an operation is to be deleted from the database service configuration file.
-name <i>name</i>	Specifies the name of the operation to delete.
-v	Displays the version number for the tool.
-verbose	Displays comments during the generation.
-quiet	Suppresses comments during the generation.

Name

`wstd` — generates a deployment descriptor that can be used to deploy a Artix ESB C++ Runtime plug-in into the Artix ESB C++ Runtime container

Synopsis

```
wstd {-service QName} {-pluginName name} {-pluginType { Cxx | Java }}  
[-pluginImpl name] [-pluginURL dir] [-wsdlurl URL] [-provider namespace]  
[-file file] [-d dir] [[-quiet] | [-verbose]] [-h] [-v]
```

Description

`wstd` generates a deployment descriptor that can be used to deploy and Artix ESB C++ Runtime plug-in into the Artix ESB C++ Runtime container.

Required Options

The tool has the following required options:

Option	Interpretation
<code>-service <i>QName</i></code>	Specifies the QName of the plug-in's service as given in its contract.
<code>-pluginName <i>name</i></code>	Specifies the name of the plug-in as specified in the Artix ESB C++ Runtime configuration file.
<code>-pluginType {Cxx Java}</code>	Specifies if the plug-in is implemented in C++ or Java.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
<code>-pluginImpl <i>name</i></code>	Specifies the library/class name of the plug-in's implementation.
<code>-pluginURL <i>dir</i></code>	Specifies the directory where the plug-in's implementation is located.
<code>-wsdlurl <i>URL</i></code>	Specifies the location of the contract defining the service implemented by the plug-in.
<code>-provider <i>namespace</i></code>	Specifies the namespace under which your plug-in's <code>ServantProvider</code> is registered with the bus.

Generating Support Files

Option	Interpretation
<code>-file <i>file</i></code>	Specifies the name of the generated deployment descriptor.
<code>-d <i>dir</i></code>	Specifies the directory where the generated file will be written.
<code>-h</code>	Displays the tool's usage statement.
<code>-v</code>	Displays the tool's version.
<code>-quiet</code>	Specifies that the tool is to run in quiet mode.
<code>-versbose</code>	Specifies that the tool is to run in verbose mode.

Name

`artix wsdl2acl` — generates a starting point ACL file from a WSDL document

Synopsis

```
artix wsdl2acl {-s server} {WSDL-URL} [-i interface] [-r  
default_role] [-d output_dir] [-O output_file] [-props props_file]  
[-L license] [[-quiet] | [-verbose]] [-v]
```

Description

artix wsdl2acl generates a starting point ACL file from a WSDL document. The generated ACL must be completed before it can be used.

Required Arguments

The command has the following required arguments:

Option	Interpretation
<code>-s <i>server</i></code>	Specifies the name of the server. Typically this is the ORB name of the server.
<code><i>WSDL-URL</i></code>	Specifies the name of the WSDL file from which the ACL file is generated.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
<code>-i <i>interface</i></code>	Specifies the <code>portType</code> for which ACL data will be generated. The default is to generate information for all port types defined in the contract.
<code>-r <i>default_role</i></code>	Specifies the role name to use in the generated ACL document. The default is <code>IONAUserRole</code> .
<code>-d <i>output_dir</i></code>	Specifies the directory where the generated file will be written.
<code>-o <i>output_file</i></code>	Specifies the name of the generated ACL file. The default is to use the name of the WSDL file with a <code>.acl</code> extension.
<code>-props <i>props_file</i></code>	Specifies the properties file listing the roles for each operation.

Option	Interpretation
<code>-L <i>license</i></code>	Specifies the location of your Artix ESB license file. The default behavior is to check <code>IT_PRODUCT_DIR\etc\license.txt</code> .
<code>-v</code>	Displays the tool's version.
<code>-quiet</code>	Specifies that the tool is to run in quiet mode.
<code>-verbose</code>	Specifies that the tool is to run in verbose mode.