Artix 5.6.3

Artix Command Reference, C++ Runtime

2015-02-17

Contents

Preface

What is Covered in This Book

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

Who Should Read This Book

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

The Artix ESB Documentation Library

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

Further Information and Product Support

Additional technical information or advice is available from several sources.

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

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

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

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

Information We Need

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

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

Contact information

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

Additional technical information or advice is available from several sources.

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

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

You may want to check these URLs in particular:

- http://www.microfocus.com/products/corba/orbix/orbix-6.aspx (trial software download and Micro Focus Community files)
- https://supportline.microfocus.com/productdoc.aspx (documentation updates and PDFs)

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

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

Prerequisites

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

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

- Run the **artix_env** script (**artix_env.bat** on Windows**)** located in `*InstallDir*/bin`.

Generating WSDL

Artix ESB C++ Runtime provides a number of command line tools for generating WSDL.

This chapter describes the following tools:

- idltowsdl
- coboltowsdl
- xsdtowsdl

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] [-I *`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
idlfile	Specifies the name of the IDL file.

Optional Arguments

The command has the following optional arguments:

Option	Interpretation
-usetypes	Generate rpc style messages. rpc style messages have parts defined using XML Schema types instead of XML elements.

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

Option	Interpretation
-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.
-fasttrack	Use the fasttrack wizard. You must also use the `-interface` and `-soapaddr` 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 *name*	Specifies the IDL interface for which WSDL will be generated by the fasttrack wizard.
-soapaddr *port*	Specifies the SOAP address to use in the generated `port` element when using the fasttrack wizard.
-e encoding	Specifies the value for the generated WSDL document's xml encoding attribute. The default is UTF-8.
-L file	Specifies the location of your license file. The default is `IT_PRODUCT_DIR\etc\license.txt`.
-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.

coboltowsdl

Generates a WSDL document with a fixed binding from a COBOL copybook.

Synopsis

coboltowsdl {-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*] [-qui-et] [-h] [-v] [-verbose]

Required Arguments

The command has the following required arguments:

Option	Interpretation
-b binding	Specifies the name for the generated binding.
-op operation	Specifies the name for the generated operation.
-im [inmessage:]incopybook	Specifies the name of the input message and the copybook file from which the data defining the message is taken. The input message name, *`inmessage`*, 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

Option	Interpretation
-om [outmessage:]outcopybook	Specifies the name of the output message and the copybook file from which the data defining the message is taken. The output message name, *`outmessage`*, 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.
-fm [faultmessage:]faultcopybook	Specifies the name of a fault message and the copybook file from which the data defining the message is taken. The fault message name, *`faultmessage`*, 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 fault message. You can specify more than one fault message
-i portType	Specifies the name of the port type in the generated WSDL. Defaults to a *binding*PortType . (If *binding* ends in Binding or binding, it is stripped off before being used in any of the default names.)
-t target	Specifies the target namespace for the generated WSDL. Defaults to `http://www.iona.com/`*`binding`*.
-x schema_name	Specifies the namespace for the schema in the generated WSDL. Defaults to `http://www.iona.com/`*`binding`*`/types`.

Option	Interpretation
-useTypes	Specifies that the generated WSDL will use `type` elements. Default is to generate `element` elements for schema types.
-oneway	Specifies that the operation does not have a response message.
-qualified	Specifies that the schemaelement in the generated WSDL has its elementFormDefault and attributeFormDefault attributes set to qualified.
-o file	Specifies the name of the generated WSDL file. Defaults to *binding*.wsdl.
-L file	Specifies the location of your license file. The default is IT_PRODUCT_DIR\etc\license.txt.
-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.

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
-t namespace	Specifies the target namespace for the generated contract. The default is to use the Artix target namespace.
-n name	Specifies the name for the generated contract and is the value of the `name` attribute in the contract's root `definitions` element. The default is to use the schema document's file name.
-d *dir*	Specifies the output directory for the generated contract.
-o file	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 `maxwell.xsd` the resulting contract will be `maxwell.wsdl`.
-L file	Specifies the location of your license file. The default is `IT_PRODUCT_DIR\etc\license.txt`.
-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.
xsdurl	Specifies the URL of the XML Schema Document.

Adding Bindings

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

This chapter describes the following tools:

- wsdltosoap
- wsdltocorba -corba

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
-i *portType*	Specifies the name of the `portType` element being mapped to a SOAP binding.
-n *namespace*	Specifies the namespace to use for the SOAP binding.
wsdlurl	Specifies the WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
-soapversion {1.1 | 1.2}	Specifies the SOAP version of the generated binding. Defaults to `1.1`.
-style {document | rpc}	Specifies the encoding style to use in the SOAP binding. Defaults to `document`.
-use {literal | encoded}	Specifies how the data is encoded. Default is `literal`.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-d *dir*	Specifies the output directory for the generated contract.
-L file	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.

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.

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
-i portType	Specifies the name of the port type for which the CORBA binding is generated.
wsdl	Specifies the WSDL document to which the binding is added.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
-idl	Specifies that an IDL file will be generated for the generated CORBA binding. You must also use the `-b` flag in conjunction with this flag.
-d *dir*	Specifies the directory into which the new WSDL document is written.
-b binding	Specifies the name of the generated CORBA binding. The default is *`portType`*`Binding`.
-o file	Specifies the name of the generated WSDL document. The default is *`wsdl_file`*`-corba.wsdl`.

Option	Interpretation
-props namespace	Specifies the namespace to use for the generated CORBA typemap.
-wrapped	Specifies that the generated binding uses wrapped types.
-L file	Specifies the location of your Artix license file. The default behavior is to check `IT_PRODUCT_DIR\etc\license.txt`.
-h	Displays the tool’s usage statement.
-v	Displays the tool's version.
-quiet	Specifies that the tool is to run in quiet mode.
-verbose	Specifies that the tool is to run in verbose mode.

Adding Endpoints

Artix ESB C++ Runtime provides command line tools for adding endpoints to WSDL documents.

This chapter describes the following tools:

- wsdltoservice -transport http/soap
- wsdltoservice -transport corba
- wsdltoservice -transport iiop
- wsdltoservice -transport mq
- wsdltoservice -transport tuxedo

wsdltoservice – transport http/soap

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] [-hsct contentType] [-hscc serverCacheControl] [-hsscse supressClientSendErrors] [-hsscre supressClientReceiveErrors] [-hshka honorKeepAlive] [-hsmps serverMultiplexPoolSize] [-hsrurl redirectURL] [-hscl contentLocation] [-hsce contentEncoding] [-hsst serverType] [-hssc serverCentificate] [-hsscc serverCentificateChain] [-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] [-hcck cookie] [-hcbt browserType] [-hcr referer] [-hcps proxyServer] [-hcpun proxyUserName] [-hcpp proxyPassword] [-hcpat proxyAuthorizationType] [-hcpa proxyAuthorization] [-hccce 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
wsdlurl	Specifies the WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
-transport soap/http	If the payload being sent over the wire is SOAP, use `-transport soap`. For all other payloads use `-transport http`.
-e service	Specifies the name of the generated service.
-t port	Specifies the value of the `name` attribute of the generated `port` element.
-b binding	Specifies the name of the binding for which the service is generated.
-a address	Specifies the value used in the `address` element of the port.
-hssdt serverSendTimeout	Specifies the number if milliseconds that the server can continue to try to send a response to the client before the connection is timed out.
-hscvt serverReceiveTimeout	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.
-hstrc trustedRootCertificates	Specifies the full path to the X509 certificate for the certificate authority.
-hsuss useSecureSockets	Specifies if the server uses secure sockets. Valid values are `true` or `false`.
-hsct contentType	Specifies the media type of the information being sent in a server response.
-hscc serverCacheControl	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.
-hsscse	Specifies whether exceptions are thrown

Option	Interpretation
supressClientSendErrors	when an error is encountered on receiving a client request. Valid values are `true` or `false`.
-hsscre supressClientReceiveErrors	Specifies whether exceptions are thrown when an error is encountered on sending a response to a client. Valid values are `true` or `false`.
-hshka honorKeepAlive	Specifies if the server honors client keep-alive requests. Valid values are trueor false.
-hsmps serverMultiplexPoolSize	
-hsrurl redirectURL	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.
-hscl contentLocation	Specifies the URL where the resource being sent in a server response is located.
-hsce contentEncoding	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.
-hsst serverType	Specifies what type of server is sending the response to the client.
-hssc serverCentificate	Specifies the full path to the X509 certificate issued by the certificate authority for the server.
-hsscc serverCentificateChain	Specifies the full path to the file that contains all the certificates in the chain.
-hsspk serverPrivateKey	Specifies the full path to the private key that corresponds to the X509 certificate specified by *`serverCertificate`*.
-hsspkp serverPrivateKeyPassword	Specifies a password that is used to decrypt the private key.
-hcst clientSendTimeout	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.
-hccvt clientReceiveTimeout	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.

Option	Interpretation
-hctrc trustedRootCertificates	Specifies the full path to the X509 certificate for the certificate authority.
-hcuss ueSecureSockets	Specifies if the client uses secure sockets. Valid values are `true` or `false`.
-hcct contentType	Specifies the media type of the data being sent in the body of the client request.
-hccc clientCacheControl	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.
-hcar autoRedirect	Specifies if the server should automatically redirect client requests.
-hcun userName	Specifies the username the client uses to register with servers.
-hcp password	Specifies the password the client uses to register with servers.
-hcat clientAuthorizationType	Specifies the authorization mechanisms the client uses when contacting servers.
-hca clientAuthorization	Specifies the authorization credentials used to perform the authorization.
-hca accept	Specifies what media types the client is prepared to handle.
-hcal acceptLanguage	Specifies what language the client prefers for the purposes of receiving a response.
-hcae acceptEncoding	Specifies what content codings the client is prepared to handle.
-hch *host*	Specifies the internet host and port number of the resource on which the client request is being invoked.
-hccn clientConnection	Specifies if the client will open a new connection for each request or if it will keep the original one open. Valid values are `close` and `Keep-Alive`.
-hcck *cookie*	Specifies a static cookie to be sent to the server.
-hcbt browserType	Specifies information about the browser from which the client request originates.
-hcr referer	Specifies the value for the client's referring entity.

Option	Interpretation
-hcps proxyServer	Specifies the URL of the proxy server, if one exists along the message path.
-hcpun proxyUserName	Specifies the username that the client uses to authorize with proxy servers.
-hcpp proxyPassword	Specifies the password that the client uses to authorize with proxy servers.
-hcpat proxyAuthorizationType	Specifies the authorization mechanism the client uses with proxy servers.
-hcpa proxyAuthorization	Specifies the actual data that the proxy server should use to authenticate the client.
-hccce clientCertificate	Specifies the full path to the X509 certificate issued by the certificate authority for the client.
-hcccc clientCertificateChain	Specifies the full path to the file that contains all the certificates in the chain.
-hcpk clientPrivateKey	Specifies the full path to the private key that corresponds to the X509 certificate specified by *`clientCertificate`*.
-hcpkp clientPrivateKeyPassword	Specifies a password that is used to decrypt the private key.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-d *dir*	Specifies the output directory for the generated contract.
-L file	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.

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] `wsdlurl`

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
wsdlurl	The WSDL document from which to base the generated WSDL document.

Optional Arguments

The tool has the following optional arguments:

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

Option	Interpretation
-pst persists	Specifies the value of the persistence policy. Valid values are `true` and `false`.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-d *dir*	Specifies the output directory for the generated contract.
-L file	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.

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] *wsdlurl*

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**
-e service	Specifies the name of the generated CORBA service.
-t port	Specifies the value of the `name` attribute of the generated `port` element.
-b binding	Specifies the name of the binding for which the service is generated.
-a address	Specifies the value used in the `iiop:address` element of the port.
-poa poaName	Specifies the value of the POA name policy.
-sid serviceId	Specifies the value of the ID assignment policy.
-pst persists	Specifies the value of the persistence policy. Valid values are `true` and `false`.
-paytype *payload*	Specifies the type of data being sent in the message payloads. Valid values are `string`, `octets`, `imsraw`, `imsraw_binary`, `cicsraw`, and `cicsraw_binary`.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-d *dir*	Specifies the output directory for the generated contract.
-L file	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.

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] [-cqn 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.

Option	Interpretation
-e service	Specifies the name of the generated service.
-t port	Specifies the value of the `name` attribute of the generated `port` element.
-b binding	Specifies the name of the binding for which the service is generated.
-sqm queueManager	Specifies the name of the server's queue manager.
-sqn *queue*	Specifies the name of the server's request queue.
-srqm queueManager	Specifies the name of the server's reply queue manager.
-srqn *queue*	Specifies the name of the server's reply queue.

Option	Interpretation
-smqn modelQueue	Specifies the name of the server's model queue.
-sus usageStyle	Specifies the value of the server's `UsageStyle` attribute. Valid values are `Peer`, `Requester`, or `Responder`.
-scs correlationStyle	Specifies the value of the server's `CorrelationStyle` attribute. Valid values are `messageId`, `correlationId`, or `messageId copy`.
-sam accessMode	Specifies the value of the server's `AccessMode` attribute. Valid values are `peek`, `send`, `receive`, `receive exclusive`, or `receive shared`.
-sto timeout	Specifies the value of the server's `Timeout` attribute.
-sme expiry	Specifies the value of the server's `MessageExpiry` attribute.
-smp priority	Specifies the value of the server's `MessagePriority` attribute.
-smi messageId	Specifies the value of the server's `MessageId` attribute.
-sci correlationId	Specifies the value of the server's `CorrelationId` attribute.
-sd delivery	Specifies the value of the server's `Delivery` attribute.
-st transactional	Specifies the value of the server's `Transactional` attribute. Valid values are `none`, `internal`, or `xa`.
-sro reportOption	Specifies the value of the server's `ReportOption` attribute. Valid values are `none`, `coa`, `cod`, `exception`, `expiration`, or `discard`.
-sf format	Specifies the value of the server's `Format` attribute.
-sad applicationData	Specifies the value of the server's `ApplicationData` attribute.
-sat accountingToken	Specifies the value of the server's `AccountingToken` attribute.
-scn connectionName	Specifies the name of the connection by which the adapter connects to the queue.
-sc convert	Specifies if the messages in the queue need to be converted to the system's native encoding. Valid values are `true` or `false`.
-scr reusable	Specifies the value of the server's `ConnectionReusable` attribute. Valid values are `true`or `false`.
-scfp fastPath	Specifies the value of the server's `ConnectionFastPath` attribute. Valid values are `true`or `false`.
-said *idData*	Specifies the value of the server's `ApplicationIdData` attribute.

Option	Interpretation
-saod originData	Specifies the value of the server's `ApplicationOriginData` attribute.
-cqm queueManager	Specifies the name of the client's queue manager.
-cqn *queue*	Specifies the name of the client's request queue.
-crqm queueManager	Specifies the name of the client's reply queue manager.
-crqn *queue*	Specifies the name of the client's reply queue.
-cmqn modelQueue	Specifies the name of the client's model queue.
-cus usageStyle	Specifies the value of the client's `UsageStyle` attribute. Valid values are `Peer`, `Requester`, or `Responder`.
-ccs correlationStyle	Specifies the value of the client's `CorrelationStyle` attribute. Valid values are `messageId, correlationId`, or `messageId copy`.
-cam accessMode	Specifies the value of the client's `AccessMode` attribute. Valid values are `peek`, `send`, `receive`, `receive exclusive`, or `receive shared`.
-cto timeout	Specifies the value of the client's `Timeout` attribute.
-cme expiry	Specifies the value of the client's `MessageExpiry` attribute.
-cmp priority	Specifies the value of the client's `MessagePriority` attribute.
-cmi messageId	Specifies the value of the client's `MessageId` attribute.
-cci correlationId	Specifies the value of the client's `CorrelationId` attribute.
-cd delivery	Specifies the value of the client's `Delivery` attribute.
-ct transactional	Specifies the value of the client's `Transactional` attribute. Valid values are `none`, `internal`, or `xa`.
-cro reportOption	Specifies the value of the client's `ReportOption` attribute. Valid values are `none`, `coa`, `cod`, `exception`, `expiration`, or `discard`.
-cf format	Specifies the value of the client's `Format` attribute.
-cad applicationData	Specifies the value of the client's `ApplicationData` attribute.
-cat accountingToken	Specifies the value of the client's `AccountingToken` attribute.
-ccn connectionName	Specifies the name of the connection by which the adapter connects to the queue.

Option	Interpretation
-cc convert	Specifies if the messages in the queue need to be converted to the system’s native encoding. Valid values are `true` or `false`.
-ccr reusable	Specifies the value of the client’s `ConnectionReusable` attribute. Valid values are `true` or `false`.
-ccfp fastPath	Specifies the value of the client’s `ConnectionFastPath` attribute. Valid values are `true` or `false`.
-caid *idData*	Specifies the value of the client’s `ApplicationIdData` attribute.
-caod originData	Specifies the value of the client’s `ApplicationOriginData` attribute.
-caqn *queue*	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 userId	Specifies the value of the client’s `UserIdentification` attribute.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-L file	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.
-d *dir*	Specifies the output directory for the generated contract.
wsdlurl	Specifies the name of the WSDL file to process.

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] *wsdlurl*

Description

wsdltoservice -transport tuxedo adds an 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
-e service	Specifies the name of the generated service.
-t port	Specifies the value of the `name` attribute of the generated `port` element.
-b binding	Specifies the name of the binding for which the service is generated.
-tsn tuxService	Specifies the name the service uses to register with the Tuxedo bulletin board.
-tfn tuxService:tuxFunction	Specifies the name of the function to be used on the specified Tuxedo bulletin board.
-ton tuxService:operation	Specifies the WSDL operation that is handled by the specified Tuxedo endpoint.
-o file	Specifies the filename for the generated contract. The default is to append `-service` to the name of the imported contract.
-d *dir*	Specifies the output directory for the generated contract.
-L file	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.

Option	Interpretation
-verbose	Specifies that the tool runs in verbose mode.
-h	Displays the tool's usage statement.
-v	Displays the tool's version.
wsdlurl	Specifies the name of the WSDL file to process.

Adding Routes

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

This chapter describes the following tool:

- wsdltorouting

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 the ***Artix Router Guide C++***.

Arguments

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

Option	Interpretation
-rn *name*	Specifies the name of the generated route. If no name is given a unique name will be generated for the route.
-ssn service	Specifies the name of the service to use as the source of the route.
-spn *port*	Specifies the name of the port to use as the source of the route. The port must correspond to a `port` element in the specified service.
-dsn service	Specifies the name of the service to use as the destination of the route.
-dpn *port*	Specifies the name of the port to use as the destination of the route. The port must correspond to a `port` element in the specified service.
-on operation	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.
-ta attribute	Specifies a transport attribute to use in defining the route.

Option	**Interpretation**
-d *dir*	Specifies the output directory for the generated contract.
-o file	Specifies the filename for the generated contract.
-L file	Specifies the location of your Artix license file. The default behavior is to check `IT_PRODUCT_DIR\etc\license.txt`.
-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.
wsdl	Specifies the name of the WSDL document to which the route is added.

Validating WSDL

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

This chapter describes the following tool:

- schemavalidator

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 *wsi-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 schema-directory	Specifies the directory used to search for schemas. This switch can appear multiple times.
-s schema-url	Specifies the URL of a user specific schema to be included in the validation of the contract. This switch can appear multiple times.
-w WSDL_XSD_URL	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 wsi-test-tools.home	Specifies the base directory of wsi-test-tools.

Argument	Interpretation
-tad BasicProfileAssertions	Specifies the URL of the of `BasicProfileTestAssertions.xml` used in wsi-test-tools.
-L file	Specifies the location of your Artix license file. The default behavior is to check `IT_PRODUCT_DIR\etc\license.txt`.
-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 ESB C++ Runtime includes a command line driven XSLT processor for transforming XML documents.

This chapter describes the following tool:

- xslttransform

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 controling the transformation are reviewed in the following table.

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

Generating Code from WSDL

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

This chapter describes the following tools:

- wsdlgen
- wsdltocpp

wsdlgen

Generates application code based on JavaScript templates.

Synopsis

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

Description

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

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

Arguments

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

Option	Interpretation
-G ApplicationType	Specifies the type of application to generate. The following application types are defined by default: • cxx—for generating C++ code
-T TemplateID	Specifies the template ID that governs code generation. See Template IDs on page 67 for details.
-C ConfigFile	Specifies the location of a configuration file to be used by the code generator.

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

Template IDs

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

Option	Interpretation
impl	Generate the stub and skeleton code require to implement the interface defined by the specified WSDL portType element.
server	Generate a simple main() for a standalone service that will host an implementation of the interface defined by the specified WSDL portType element. Stub code is also generated.
client	Generate a C++ file class that invokes all of the operations defined by the specified WSDL portType element. Stub code is also generated.
plugin	If generating C++, generate all of the code needed to implement the interface defined by the specified WSDL portType element as an Artix plug-in.
all	For C++, generate a client, a server, and an Artix plug-in.
make	Generate a make file for a C++ application.

wsdltocpp

Generates C++ stubs and skeletons for the services defined in a WSDL document.

Synopsis

wsdltocpp [-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 *version*] [-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

wsdltocpp 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
wsdlurl	The WSDL document from which the code is generated.

Optional Arguments

The tool uses the following optional arguments:

Option	Interpretation
-i port_type	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.
-e web_service_name[:port_list]	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.

Option	Interpretation
-b binding_name	Specifies the name of the binding to use when generating code. The default is the first binding listed in the contract.
-d output_dir	Specifies the directory to which the generated code is written. The default is the current working directory.
-n [URI=]C++namespace	Maps an XML namespace to a C++ namespace. The C++ stub code generated from the XML namespace (*URI*) is put into the specified C++ namespace. This switch can appear multiple times.
-nexclude URI[=C++namespace]	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.
-ninclude URI[=C++namespace]	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.
-nimport C++namespace	Specifies the C++ namespace to use for the code generated from imported schema.
-impl	Generates the skeleton code for implementing the server defined by the contract.
-m {NMAKE |UNIX}:[executable | library]	Used in combination with `-impl` to generate a makefile for the specified platform (`NMAKE` for Windows or `UNIX` for UNIX). You can specify that the generated makefile builds an executable, by appending `:executable`, or a library, by appending `:library`.
-libv version	Used in combination with either `-m NAME:library` or `-m UNIX:library` 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.
-f	*Deprecated*—Was needed to support routing in earlier versions.
-server	Generates code for a sample implementation of a server.
-client	Generates code for a sample implementation of a client.

Option	Interpretation
-sample	Generates code for a sample implementation of a client and a server (equivalent to `-server -client`).
-plugin[:plugin_name]	Generates servant registration code as a bus plug-in. You can optionally specify the plug-in name by appending `:`*`plugin_name`* to this option. If no plug-in name is specified, the default name is *`ServiceNamePortTypeName`*. The service name is specified by the `-e` option.
-deployable	(Used with `-plugin`.) Generates a deployment descriptor file, `deploy`*`ServiceName`*`.xml`, which is needed to deploy a plug-in into the Artix ESB C++ Runtime container.
-global	(Used with `-plugin`.) In the generated plug-in code, instantiate the plug-in using a `GlobalBusORBPlugIn` object instead of a `BusORBPlugIn` object. A `GlobalBusORBPlugIn` initializes the plug-in automatically, as soon as it is constructed (suitable approach for plug-ins that are linked directly with application code). A `BusORBPlugIn` is not initialized unless the plug-in is either listed in the `orb_plugins` list or deployed into an Artix ESB C++ Runtime container (suitable approach for dynamically loading plug-ins).
-license	Displays the currently available licenses.
-declspec *declspec*	Creates Visual C++ declaration specifiers for `dllexport` and `dllimport`. 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	Dislays 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.
-user_reserved_words word1[:wordn...]	Specifies a colon-separated list of words to be treated as reserved. For example, -user_reserved_words SEC:MILLISECwould generate a header file including '`class _SEC' instead of 'class SEC`'.

Option	Interpretation
-L file	Specifies the location of your Artix license file. The default behavior is to check IT_PRODUCT_DIR\etc\license.txt.
-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.

Generating Support Files

Artix ESB C++ Runtime provides tools to generate a number of support files.

This chapter describes the following tools:

- wsdltocorba -idl
- wsdd
- wsdltoacl

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 an Artix ESB C++ Runtime CORBA binding.

Required Arguments

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

Option	Interpretation
-b binding	Specifies the name of the CORBA binding for which the IDL is generated.
wsdl	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
-corba	Specifies that a CORBA binding is to be generated.
-i portType	Specifies the name of the port type for which the CORBA binding is generated.
-d *dir*	Specifies the directory into which the new WSDL document is written.
-o file	Specifies the name of the generated WSDL document. The default is `wsdl_file-corba.wsdl`.
-props namespace	Specifies the namespace to use for the generated CORBA typemap.
-wrapped	Specifies that the generated binding uses wrapped types.
-L file	Specifies the location of your Artix license file. The default behavior is to check `IT_PRODUCT_DIR\etc\license.txt`.
-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.

wsdd

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

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

Description

wsdd 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
-service *QName*	Specifies the QName of the plug-in's service as given in its contract.
-pluginName *name*	Specifies the name of the plug-in as specified in the Artix ESB C++ Runtime configuration file.
-pluginType {Cxx | Java}	Specifies if the plug-in is implemented in C++ or Java.

Optional Arguments

The tool has the following optional arguments:

Option	Interpretation
-pluginImpl *name*	Specifies the library/class name of the plug-in's implementation.
-pluginURL *dir*	Specifies the directory where the plug-in's implementation is located.
-wsdlurl *URL*	Specifies the location of the contract defining the service implemented by the plug-in.
-provider *namespace*	Specifies the namespace under which your plug-in's `ServantProvider` is registered with the bus.
-file *file*	Specifies the name of the generated deployment descriptor.
-d *dir*	Specifies the directory where the generated file will be written.
-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.

wsdltoacl

Generates a starting point ACL file from a WSDL document.

Synopsis

wsdltoacl {-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 wsdltoacl 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
-s server	Specifies the name of the server. Typically this is the ORB name of the server.
WSDL-URL	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
-i interface	Specifies the `portType` for which ACL data will be generated. The default is to generate information for all port types defined in the contract.
-r default_role	Specifies the role name to use in the generated ACL document. The default is `IONAUserRole`.
-d output_dir	Specifies the directory where the generated file will be written.
-o output_file	Specifies the name of the generated ACL file. The default is to use the name of the WSDL file with a `.acl` extension.

Option	Interpretation
-props props_file	Specifies the properties file listing the roles for each operation.
-L license	Specifies the location of your Artix ESB license file. The default behavior is to check `IT_PRODUCT_DIR\etc\license.txt`.
-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.