Artix™

Glossary

Version 4.1, September 2006

Making Software Work Together™

Updated: September 26, 2006

Preface

What is Covered in this Book

This book provides definitions for terms used in the Artix documentation library, with special attention to terms with Artix-specific meanings.

Who Should Read this Book

This book is intended for all users of the Artix documentation library.

The Artix Library

The Artix documentation library is organized in the following sections:

- Getting Started
- Designing Artix Solutions
- Configuring and Managing Artix Solutions
- Using Artix Services
- Integrating Artix Solutions
- Integrating with Management Systems
- Reference
- Artix Orchestration

Getting Started

The books in this section provide you with a background for working with Artix. They describe many of the concepts and technologies used by Artix. They include:

- Release Notes contains release-specific information about Artix.
- Installation Guide describes the prerequisites for installing Artix and the procedures for installing Artix on supported systems.
- Getting Started with Artix describes basic Artix and WSDL concepts.

- Using Artix Designer describes how to use Artix Designer to build Artix solutions.
- Artix Technical Use Cases provides a number of step-by-step examples of building common Artix solutions.

Designing Artix Solutions

The books in this section go into greater depth about using Artix to solve real-world problems. They describe how to build service-oriented architectures with Artix and how Artix uses WSDL to define services:

- Building Service-Oriented Infrastructures with Artix provides an overview of service-oriented architectures and describes how they can be implemented using Artix.
- Writing Artix Contracts describes the components of an Artix contract. Special attention is paid to the WSDL extensions used to define Artix-specific payload formats and transports.

Developing Artix Solutions

The books in this section how to use the Artix APIs to build new services:

- Developing Artix Applications in C++ discusses the technical aspects of programming applications using the C++ API.
- Developing Advanced Artix Plug-ins in C++ discusses the technical aspects of implementing advanced plug-ins (for example, interceptors) using the C++ API.
- Developing Artix Applications in Java discusses the technical aspects of programming applications using the Java API.

Configuring and Managing Artix Solutions

This section includes:

- Configuring and Deploying Artix Solutions explains how to set up your Artix environment and how to configure and deploy Artix services.
- Managing Artix Solutions with JMX explains how to monitor and manage an Artix runtime using Java Management Extensions.

Using Artix Services

The books in this section describe how to use the services provided with Artix:

- Artix Router Guide explains how to integrate services using the Artix router.

- Artix Locator Guide explains how clients can find services using the Artix locator.
- Artix Session Manager Guide explains how to manage client sessions using the Artix session manager.
- Artix Transactions Guide, C++ explains how to enable Artix C++ applications to participate in transacted operations.
- Artix Transactions Guide, Java explains how to enable Artix Java applications to participate in transacted operations.
- Artix Security Guide explains how to use the security features in Artix.

Integrating Artix Solutions

The books in this section describe how to integrate Artix solutions with other middleware technologies.

- Artix for CORBA provides information on using Artix in a CORBA environment.
- Artix for J2EE provides information on using Artix to integrate with J2EE applications.

For details on integrating with Microsoft's .NET technology, see the documentation for Artix Connect.

Integrating with Management Systems

The books in this section describe how to integrate Artix solutions with a range of enterprise and SOA management systems. They include:

- IBM Tivoli Integration Guide explains how to integrate Artix with the IBM Tivoli enterprise management system.
- BMC Patrol Integration Guide explains how to integrate Artix with the BMC Patrol enterprise management system.
- CA-WSDM Integration Guide explains how to integrate Artix with the CA-WSDM SOA management system.
- AmberPoint Integration Guide explains how to integrate Artix with the AmberPoint SOA management system.

Reference

These books provide detailed reference information about specific Artix APIs, WSDL extensions, configuration variables, command-line tools, and terms. The reference documentation includes:

- Artix Command Line Reference

- Artix Configuration Reference
- Artix WSDL Extension Reference
- Artix Java API Reference
- Artix C++ API Reference
- Artix .NET API Reference
- Artix Glossary

Artix Orchestration

These books describe the Artix support for Business Process Execution Language (BPEL), which is available as an add-on to Artix. These books include:

- Artix Orchestration Release Notes
- Artix Orchestration Installation Guide
- Artix Orchestration Administration Console Help.

Getting the Latest Version

The latest updates to the Artix documentation can be found at http://www.iona.com/support/docs.

Compare the version dates on the web page for your product version with the date printed on the copyright page of the PDF edition of the book you are reading.

Searching the Artix Library

You can search the online documentation by using the **Search** box at the top right of the documentation home page:

http://www.iona.com/support/docs

To search a particular library version, browse to the required index page, and use the **Search** box at the top right, for example:

http://www.iona.com/support/docs/artix/4.1/index.xml

You can also search within a particular book. To search within a HTML version of a book, use the **Search** box at the top left of the page. To search within a PDF version of a book, in Adobe Acrobat, select **Edit**|**Find**, and enter your search text.

Artix Online Help

Artix Designer and Artix Orchestration Designer include comprehensive online help, providing:

- Step-by-step instructions on how to perform important tasks
- A full search feature
- Context-sensitive help for each screen

There are two ways that you can access the online help:

- Select **Help|Help Contents** from the menu bar. The help appears in the contents panel of the Eclipse help browser.
- Press **F1** for context-sensitive help.

In addition, there are a number of cheat sheets that guide you through the most important functionality in Artix Designer and Artix Orchestration Designer. To access these, select **Help|Cheat Sheets**.

Artix Glossary

The Artix Glossary is a comprehensive reference of Artix terms. It provides quick definitions of the main Artix components and concepts. All terms are defined in the context of the development and deployment of Web services using Artix.

Additional Resources

The IONA Knowledge Base (http://www.iona.com/support/knowledge_base/index.xml) contains helpful articles written by IONA experts about Artix and other products.

The IONA Update Center (http://www.iona.com/support/updates/index.xml) contains the latest releases and patches for IONA products.

If you need help with this or any other IONA product, go to IONA Online Support (http://www.iona.com/support/index.xml).

Comments, corrections, and suggestions on IONA documentation can be sent to docs-support@iona.com .

Document Conventions

Typographical conventions

This book uses the following typographical conventions:

`Fixed width`	Fixed width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `IT_Bus::AnyType` class. Constant width paragraphs represent code examples or information a system displays on the screen. For example: `#include <stdio.h>`
`Fixed width italic`	Fixed width italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example: `% cd /users/`*`YourUserName`*
Italic	Italic words in normal text represent *emphasis* and introduce *new terms*.
Bold	Bold words in normal text represent graphical user interface components such as menu commands and dialog boxes. For example: the **User Preferences** dialog.

Keying Conventions

This book uses the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, the command prompt is not shown.
`%`	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
`#`	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
`>`	The notation > represents the MS-DOS or Windows command prompt.

`...`
 `.`
 `.`
 `.`
	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.

`[]`	Brackets enclose optional items in format and syntax descriptions.
`{}`	Braces enclose a list from which you must choose an item in format and syntax descriptions.
`|`	In format and syntax descriptions, a vertical bar separates items in a list of choices enclosed in `{}` (braces). In graphical user interface descriptions, a vertical bar separates menu commands (for example, select **File|Open**).

Glossary

Artix-specific glossary

This glossary defines terms in the context of the development and deployment of services using Artix. Some terms are used the same way in Artix as in the context of industry-standard Web services. Other terms are used in a narrow sense in the context of Web services, but in a broader sense in the extended context of Artix-enabled enterprise services.

Terms used by analogy

Some Artix terms (including port, router, and transport) are used in Artix by analogy with the similar terms used in the context of TCP/IP networking. In all cases, these Artix terms describe software-to-software interactions, not interactions between hardware nodes as in TCP/IP networking.

Glossary navigation

Click a letter to jump to that section of this glossary.

A B C D E F G H I J K L M N O P Q R S T U W X

A

abstract contract

See logical contract.

abstract head element

An XML Schema element that cannot appear in an instance document. When a substitution group's head element is declared as abstract with `abstract=true`, a member of that element's substitution group must be used instead.

anyType

The root type for all XML Schema type definition hierarchies. All primitive types are derivatives of this type, as are all user-defined complex types.

APPC

Advanced Program-to-Program Communication (APPC), an IBM protocol for communicating between various application programs, including CICS or IMS applications in a z/OS mainframe environment. The Artix transformer on z/OS can use APPC to communicate with the IMS back end. APPC is also the protocol used by z/OS-based Web service clients to communicate with the Artix transformer on z/OS.

application server

A software platform that provides the services and infrastructure required to develop and deploy middle-tier applications. Middle-tier applications implement the business logic necessary to provide web clients with access to enterprise information systems. In a multi-tier architecture, an application server sits beside a web server or between a web server and enterprise information systems. Application servers provide the middleware for enterprise systems. JBoss, WebLogic and WebSphere are J2EE application servers.

ART

Adaptive Runtime Technology (ART) is IONA's modular framework for constructing distributed systems, based on a lightweight core and an open-ended set of plug-ins. ART supports flexible, modular deployment and configuration of services and application code. ART provides the software foundation for Artix and other IONA products.

Artix bus

An internal component of the Artix system. The Artix bus coordinates the passage of messages through the messaging chain of both services and service consumers, and is responsible for loading plug-ins into the Artix container.

Artix chain builder

An Artix service that enables you to link together a series of services in a multi-part process. This is useful if you have processes that require a set order of steps to complete, or if you wish to link together a number of smaller service modules into a complex service.

Artix Designer

A suite of GUI tools for creating and editing WSDL contracts, and for generating Java or C++ code to implement the consumer and server sides of the WSDL contract. Artix Designer is integrated into the Eclipse development environment.

In Artix for z/OS, Artix Designer can generate deployment descriptor files for various z/OS integration solutions, and can generate COBOL or PL/I code for z/OS-based Web service application development.

Artix locator

A Web service that provides Web service clients with a mechanism to discover service endpoints at runtime.

Artix message context

A container for metadata about a message. Artix uses the message context to store and transmit transport details and message header information. In the Artix Java API, the message context is an extension of the JAX-RPC message context. In the Artix C++ API, message contexts are part of the core implementation.

Artix reference

An object in an Artix-defined format that fully describes a running service. References can be passed between Artix services or between a service and its consumer as operation parameters. As of Artix 4.0, the Artix reference format is deprecated in favor of the endpoint reference format, as defined by the WS-Addressing standard.

Artix transformer

An Artix service that processes messages based on XSLT scripts and returns the result to the requesting application.

In Artix for z/OS, the Artix transformer is an IONA-supplied server application installed on the mainframe as part of an Artix for z/OS installation. The transformer serves as a broker for messages between distributed Web service or CORBA clients and CICS or IMS applications on z/OS. It also serves as a broker for messages between z/OS-based Web service clients and distributed Web services. The transformer can use the EXCI, APPC, or OTMA protocol for communication with z/OS-based applications, depending on the solution involved.

B

binding

A description of the message format and protocol details for a set of operations and messages. Bindings are created based on the information specified in a WSDL binding element.

binding element
The element in a WSDL contract that maps the messages defined for a specific `portType` to a payload format that will be sent over the wire. For example, a WSDL contract might bind HelloWorldPortType to the SOAP payload format.

BMS map sets
Basic Mapping Support (BMS) is a component of the CICS subsystem on z/OS. BMS map sets specify the screen layout details and presentation logic for CICS applications for use with green screen terminals.

BMS parser
An IONA-supplied application installed on the mainframe as part of an Artix for z/OS installation, used to generate deployment descriptor files from BMS map sets.

BPEL
Business Process Execution Language (BPEL), an XML-based language for specifying interaction and process flow between Web Services to implement a set of business rules. See also orchestration.

bridge
See router.

bus
See Artix bus and service bus.

C

CDT
C/C++ Development Tools (CDT), a subsystem of the Eclipse development environment that automates the writing and testing of applications in C and C++.

choice complex type
An XML Schema construct defined using the `choice` element to constrain the possible elements in a complex data type. When using a choice complex type, only one of the elements defined in the complex type can be present at a time.

CICS

Customer Information Control System (CICS), an IBM database and transaction management subsystem for z/OS and other platforms. CICS is sometimes pronounced *kicks*.

classloader

The portion of the Java virtual machine (JVM) responsible for finding and loading Java class files.

classloader firewall

An Artix extension that provides a way to ensure that the Artix Java runtime loads a particular set of Java classes by blocking the runtime from loading classes on the host system's classpath.

client

An application or process that requests services from other applications known as servers. The server processes may be running on the same or a different machine. See also consumer and service consumer.

concrete contract

See physical contract.

configuration domain

A collection of the configuration information for a given Artix or Orbix environment, containing all the configuration properties and values that services and applications use in that environment. Artix configuration domains are specified in a configuration file. Configuration domains might be used in a large-scale Artix implementation to organize configuration information into manageable groups.

configuration file

A file that contains configuration information for Artix or Orbix components within a specific configuration domain.

configuration scope

A subset of an Artix or Orbix configuration domain, which corresponds to an Artix bus name. By organizing configuration properties into various scopes, different settings can be provided for individual buses, or common settings provided for groups of buses. Any Artix service can be run under its own configuration scope.

connection

In Artix, an established communication link between a service consumer and an endpoint, or between any two endpoints.

connection factory

In the context of J2EE programming, an object used for creating a connection to a resource adapter.

consumer

The end user of a service, also called a *client* for that service. The more exact term in the context of a service-oriented network is *service consumer*.

container

A server executable or process into which you can deploy and manage services.

You can write service implementations as Artix C++ or Java plug-ins that you deploy as services in an Artix container. Using the container eliminates the need to write your own C++ or Java server mainline. Instead, you can deploy your service by passing the location of a generated deployment descriptor to the Artix container's administration client. This provides a powerful programming model where the code is location-independent.

contract

A description of the messages and formats accepted and generated by a service. A service’s contract is specified in a WSDL document that defines the interface and all connection-related information for that interface. A WSDL contract contains two sets of components: logical (or abstract) and physical (or concrete).

The logical components of the contract are those that describe the data types, message formats, operations, and interfaces for the services defined in the contract. Logical components are specified with the WSDL elements `types`, `message`, `portType`, and `operation`.

The physical components of the contract are those that define the payload format, middleware transport, service groupings, and the mappings between these items and the `portType` operations. The physical contract could also describe the policies of the service, such as its security requirements. The physical components are specified with the WSDL elements `binding`, `port`, and `service`.

CORBA

Common Object Request Broker Architecture (CORBA) defines language-independent standards for interoperability and portability among distributed objects. CORBA is a robust, industry-accepted standard from the Object Management Group, deployed in thousands of mission-critical systems.

CORBA naming service

An implementation of the OMG Naming Service Specification that describes how applications can map object references to names. Servers can register object references by name with a naming service repository, and can advertise those names to clients. Clients, in turn, can resolve the desired objects in the naming service by supplying the appropriate name. The Orbix naming service is an example.

CSIv2

Common Secure Interoperability protocol, version 2 (CSIv2) is an OMG standard protocol that provides the basis for application-level security in both CORBA and J2EE applications. The IONA Security Framework implements CSIv2 to transmit usernames and passwords, and to assert identities between applications.

D

deployment

The process of propagating a service into an environment so that it is ready to use. For some systems, service deployment means distributing development artifacts to a container and then activating those artifacts in that container. In Artix, deployment refers to the activation of artifacts in a container, with the presumption that the artifacts have been distributed and are available locally to the container.

deployment descriptor

A generated XML file that describes the resources needed to deploy a service in an Artix container. These resources include: the service's name, the name of the plug-in that implements the service, and whether the plug-in is written in C++ or Java. Deployment descriptors are generated by the Artix `wsdltocpp`, `wsdltojava` and `wsdd` utilities.

In Artix for z/OS, deployment descriptor files map Web service or CORBA operation details to z/OS application details. They can also map WSDL and CORBA types to COBOL or PL/I types (and vice versa).

discriminator

A data element created to support the mapping of a choice complex type to an object. The discriminator element identifies the valid element in a choice complex type.

dynamic proxy

A special Java class created at runtime by the Java virtual machine, which implements a proxy interface. A proxy interface forces object method calls to occur indirectly through a proxy object, which acts as a surrogate or delegate for the underlying object being proxied. Artix uses the dynamic proxy method to connect to remote services, as specified in the JAX-RPC specification.

In Artix C++, dynamic proxy also refers to the DLL-style APIs that allow users to develop dynamic applications without linking in stub code.

E

EAI

Enterprise Application Integration (EAI), the use of software and architectural principles to integrate disparate enterprise applications.

EAR file

Enterprise Archive (EAR) file, a compressed (.zip) file that contains the classes and other files of a J2EE application.

Eclipse

An open source application development framework provided by the Eclipse Foundation. Artix Designer is delivered as a set of Eclipse plug-ins. For more on Eclipse, see eclipse.org.

EIS

Enterprise Information System (EIS), the set of applications that constitute an enterprise's existing information infrastructure for handling company-wide information. Examples of enterprise information systems include enterprise resource planning systems, mainframe transaction processing systems, and legacy database systems.

EJB

Enterprise JavaBeans (EJB), Sun Microsystems' component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications. EJB enables the implementation of a multi-tier, distributed object architecture.

EMS

Enterprise Management System (EMS), a set of integrated tools that enable system administrators to manage large-scale production environments. Example Enterprise Management Systems are BMC Patrol™, IBM Tivoli™, HP OpenView™, and CA Unicenter™. These systems give a top-to-bottom view of every part of the network infrastructure, and enable administrators to track key server metrics and to automate recovery actions if a server crashes.

endpoint

The point of contact that a service provides for its consumers.

endpoint reference

A self-contained object that describes the network contact and policy information for an endpoint, as defined in the WS-Addressing standard. Starting with release 4.0, Artix supports WS-Addressing endpoint references as its native reference type. Compare with Artix reference.

enterprise service

A service deployed in an enterprise network. The term is used to distinguish the narrow term *Web services* from services in general. *Web services* usually refers to request-reply services deployed over a SOAP-over-HTTP transport. By contrast, Artix-enabled enterprise services might be intermediaries as well as request-reply services, and might be deployed over many other protocols and transports.

EPR

An endpoint reference.

ESB

See service bus.

EXCI

External Call Interface (EXCI), a z/OS-based protocol for communication between CICS applications. The Artix transformer on z/OS uses EXCI to communicate with the CICS back end.

F

facet

A rule in an XML Schema definition used in the derivation of user-defined simple types. Common facets include `length`, `pattern`, `totalDigits`, and `fractionDigits`.

factory pattern

A usage pattern for services in Artix where one service creates and manages instances of another service. Typically, the factory service returns references to the services it creates.

fault element

The element in a WSDL contract that defines a fault message for a portType.

fault message

A message containing error or exception information passed between a service and its consumers. Fault messages are defined using the fault element in a WSDL contract. See also request-response operation and solicit-response operation.

firewall classloader

See classloader firewall.

fixed binding

An Artix WSDL extension used to represent fixed record length data, usually when communicating with mainframe systems or COBOL-based applications, or with C language structures containing fixed-length strings.

FML

Field Manipulation Language (FML), a language for dealing with self-describing buffers, and a library of C functions that implements the language. FML is part of the proprietary Tuxedo middleware system offered by BEA Systems, Inc.

G

green screen

A monochrome CRT computer display for mainframe computers, having fixed-size characters and simple block graphics, that communicates with the host computer one screen page at a time. Examples of green screen technologies include the IBM 3270 terminal series and 3270 terminal emulation software for PCs. The name refers to the green phosphor used in early examples of green screen terminals. See also screen scraping.

H

handler

A Java message handling interface defined in the JAX-RPC standard, with methods for processing both request and response messages. Artix provides a `GenericHandler` class to provide a template for implementing message handlers. Compare with interceptor.

high availability

The ability of a system to remain operational despite catastrophic failure of one or more of its components. This is achieved in Artix using service replication, where multiple copies of a service run concurrently and operate as identical copies of each other.

host

Any computer or device on a network that is a repository for services available to other computers or devices on the network.

I

IDL

Interface Definition Language (IDL), the standard language for defining the interfaces to all CORBA objects. An IDL file defines the public API that CORBA objects expose in a server application. Clients use these interfaces to access server objects across a network. IDL interfaces are independent of operating systems and programming languages.

IIOP

Internet Inter-ORB Protocol (IIOP), the CORBA-standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. IIOP is defined as a protocol layer above the transport layer, TCP/IP.

IMS

Information Management System (IMS), an IBM database and transaction management subsystem for z/OS.

input element

The element in a WSDL contract that defines an input message for a portType.

input message

A message passed from a service consumer to a service. When mapped into Java or C++, the parts of an input message are mapped into a method's parameter list. Input messages are defined using the input element in a WSDL contract. See also request-response operation, solicit-response operation, and one-way operation.

interceptor

A C++ message handling interface with methods for processing both request and response messages. Compare with handler.

interface

The external touch point between applications to collaborate or share functional behavior. Interfaces are completely described by the combination of logical and physical portions of a WSDL contract.

Once defined in a contract, an interface is the abstract boundary that a service exposes. A service's interface is the set of message types and message exchange patterns through which service consumers can interact with that service. In a WSDL contract, interfaces are defined using the WSDL portType element.

intermediary

A service whose main role is to process all received messages in a value-added way, such as converting them from one data format to another, or routing them to another service. An intermediary has characteristics of both a service and a service consumer. Most intermediaries have an intermediary contract, which is similar in form to a service contract, except that it includes rules for processing messages.

IOR

Interoperable Object Reference (IOR), a data structure associated with a CORBA object that contains enough information to locate that object from anywhere on the network.

J

J2EE

Java 2 Platform, Enterprise Edition (J2EE), an environment for developing and deploying enterprise applications. The J2EE platform consists of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications.

J2EE Connector Architecture

An architecture specified by Sun Microsystems for integrating J2EE products with enterprise information systems. See EIS.

JAXB

Java Architecture for XML Binding (JAXB), an API that provides a way to bind an XML schema to a representation in Java code. JAXB is part of Sun Microsystems' Java Web Services Developer Pack.

JAX-RPC

Java API for XML-Based RPC (JAX-RPC), a programming model based on a specification from Sun Microsystems. The JAX-RPC specification defines APIs and conventions for supporting XML-based remote procedure calls in the Java platform. JAX-RPC is the standard on which Artix bases its Java API and data type mappings. For further information, see http://java.sun.com/xml/jaxrpc/overview.html.

JDBC
Java Database Connectivity (JDBC), an API specified in Java technology that provides Java applications with access to databases and other data sources.

JDT
Java Development Tools (JDT), a subsystem of the Eclipse development environment that automates the writing and testing of applications in Java.

JMS
Java Message Service (JMS), a Java API implementing a Sun Microsystems messaging standard that allows application components based on J2EE to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

JMX
Java Management eXtensions (JMX), a Java technology that supplies tools for managing and monitoring applications, system objects, devices, and service-oriented networks.

JNDI
Java Naming and Directory Interface (JNDI), a set of APIs specified in Java technology that assists Java applications with interfacing to multiple naming and directory services.

K

Knowledge Module
A pre-built loadable library that enables connections to the BMC Patrol EMS. The IONA Knowledge Module (KM) enables connections for Artix and Orbix applications. The IONA KM conforms to the standard BMC Software Knowledge Module design and operation.

L

list type
A data type defined in an XML Schema definition as a space-separated list of primitive type elements, defined using the `xsd:list` element.

location domain

A collection of Orbix servers under the control of a single locator daemon. The location domain can span any number of hosts across a network, and can be dynamically extended with new hosts. In Artix, this term primarily occurs in the context of connecting Artix to Orbix or other CORBA services.

locator

See Artix locator.

locator daemon

An Orbix server host facility that manages an implementation repository and acts as a control center for a location domain. Orbix clients use the locator daemon, often in conjunction with a naming service, to locate the objects they seek. In Artix, this term primarily occurs in the context of connecting Artix to Orbix or other CORBA services.

Artix also provides a separate Artix locator service, which is not related to the locator daemon.

logical contract

The abstract portion of a WSDL contract that defines the data types, message types, and the interfaces for the services defined in the contract. The logical contract answers questions such as:

- What kinds of data will this service work with?
- What kinds of data are grouped together for processing?
- What operations are related and what are their interfaces?

WSDL elements used in the logical contract include: portType element, operation element, message element, and types element. Compare with physical contract.

login service

A central Artix service that authenticates username and password combinations.

M

marshaling

The process in data communications of packing one or more items of data into a message buffer prior to transmitting that message buffer over a communication channel. In Artix, data packing is performed according to the rules of the binding element, and the communication channel is defined by the port element.

message

Any data passed between a service and a service consumer, or between two endpoints. Messages are defined in an Artix contract using the WSDL message element. See also fault message, input message, and output message.

message context

See Artix message context.

message element

The element in a WSDL contract that defines the abstract structure for a particular type of message. For example, a message might consist of a text string that can be tokenized into the parameter arguments for an operation. Another message type might contain an invoice, an account history, or a query string.

message handler

A Java class responsible for intercepting a message along the message chain and performing some processing on the raw message data. See also handler.

message-level handler

A message handler that processes messages as they pass between the binding and the transport.

message-level interceptor

The equivalent of a message-level handler, but used with Artix C++.

MFS maps

Message Format Services (MFS) is a component of the IMS subsystem on z/OS. MFS maps specify the screen layout details and presentation logic for IMS applications for use with green screen terminals.

MFS parser

An IONA-supplied application installed on the mainframe as part of an Artix for z/OS installation, used to generate deployment descriptor files from MFS maps.

N

naming service

See CORBA naming service.

nillable

In an XML Schema definition, an attribute of an element that specifies that the element is optional within a complex type.

notification operation

One type of WSDL-defined abstract operation, in which the service endpoint sends a message, but does not expect a return message. Artix WSDL-to-code generation tools do not support notification operations.

O

OASIS

An international consortium that drives the development, convergence, and adoption of Web services standards. See www.oasis-open.org.

object reference

A reference that uniquely identifies a local or remote object instance. The reference can be stored in a CORBA naming service, in a file, or in a URL. In the context of CORBA programming, this is also known as an interoperable object reference (IOR). Object references are a CORBA-specific feature used by Artix only when interfacing with a CORBA system. Contrast with endpoint reference and Artix reference.

OMG

Object Management Group (OMG), an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications, including CORBA. See www.omg.com.

one-way operation

One type of WSDL-defined abstract operation, in which the service endpoint receives a message, but does not provide a return message. One-way operations specify only input message types. Artix WSDL-to-code generation tools support one-way operations.

operation

A message interaction between a service and a service consumer. The WSDL specification provides for four types of operations:

- one-way operation
- request-response operation
- solicit-response operation
- notification operation

Artix WSDL-to-code generation tools support one-way and request-response operations. Operations are defined using the operation element in a WSDL contract.

operation element

The element in a WSDL contract that provides an abstract definition of a specific interaction between a service and a service consumer. A WSDL operation element is defined in terms of input messages, output messages, and fault messages.

ORB

Object Request Broker (ORB), the key CORBA component that manages the interaction between clients and servers, using the Internet Inter-ORB Protocol (IIOP). An ORB enables clients to make requests and receive replies from servers in a distributed computer environment.

orchestration

The coordination of a process flow between two or more Web services to implement a set of business rules, using the BPEL XML language.

OTMA

Open Transaction Manager Access (OTMA), an IBM protocol for communicating with IMS applications in the z/OS mainframe environment. The Artix transformer can use OTMA to communicate with the IMS back end.

output element

The element in a WSDL contract that defines an output message for a portType.

output message

A message passed from a service to a service consumer. When mapped into Java or C++, the parts of an output message are mapped to a method's output parameter list, including any return value. Output messages are defined using the output element in a WSDL contract. See also request-response operation, solicit-response operation, and notification operation.

P

payload format

The on-the-wire structure of messages over a given transport. Artix supports several payload formats, including SOAP, TibMsg, and fixed-record-length data. Most payload formats are independent of the transport that carries them, and could be carried over several transports. Some payload formats are transport-specific by design (CORBA) or by convention (FML).

peer manager

An Artix service that pings the endpoints of services registered with the Artix locator and Artix session manager to verify that these endpoints are still running.

physical contract

The concrete portion of a WSDL contract that defines the bindings and transport details used by the services defined by that contract. The physical contract answers questions such as:

- How is message traffic formatted on the wire?
- How and where does message traffic travel?
- Is there more than one option for transmitting a request?

WSDL elements used in the physical contract include: binding element, service element, operation element, and port element. Compare with logical contract.

plug-in

A well-defined Artix component that can be independently loaded into an application to provide a set of features. Artix defines a platform-independent framework for loading plug-ins dynamically, using dynamic linking implementations such as shared libraries, DLLs, or Java classes.

POA

A Portable Object Adapter (POA) maps abstract CORBA objects to their actual implementations, or servants. Depending on the policies you set on a POA, object-servant mappings can be static or dynamic. POA policies also determine the threading model in use, and whether object references are persistent or transient.

port

The physical mechanism used to access a service. Ports are created based on the information specified in a WSDL port element.

port element

The element in a WSDL contract that specifies the details needed to contact a service defined in the contract. The contact details might include location information and policy details. For example, a `port` element for an HTTP endpoint might specify a URL and its MIME encoding types and timeout policies. A `port` element for an MQ endpoint might specify a queue name.

portType

A named set of abstract operations along with the abstract messages involved with those operations. A portType is defined in a WSDL portType element.

portType element

The element in a WSDL contract that represents the logical interface for the service defined in the contract. A `portType` element is a collection of abstract operations supported by one or more endpoints. A `portType` is mapped to one or more transports using one or more bindings.

proxification

A feature of the Artix router wherein a reference of a certain type (for example, a CORBA reference) that passes through the router is automatically converted to a reference of another type (for example, a SOAP reference).

proxy

An object that models an interface as a class in the programming language of choice, and encapsulates physical interface implementation details.

proxy object

In Artix client code, a stand-in object that represents a particular service and port of an enterprise service. See also service proxy.

Q

QName

Industry-standard abbreviation for qualified name, as defined in the XML namespace specification. A QName is resource name that incorporates the namespace representing the specification where that resource is defined. QNames are composed of a URI representing the namespace of the resources's definition, plus the name of the resource, usually called the localPart. Some QName formats also use an alias for the namespace called the prefix.

QNames can be found in several formats. The canonical format for QNames in Artix code and Artix configuration files is the one specified in `javax.xml.namespace.QName`, which is the namespace URI enclosed in braces, followed immediately (with no punctuation) by the localPart. For, example: `{http://www.iona.com/FixedBinding}SOAPHTTPService`.

Another format is used in a self-contained document such as a WSDL contract, where a qualified name is in the form `prefix:localPart`. The `prefix` is declared in an `xmlns` statement in an XML namespace declaration in the same document. For example, `ls:SOAPHTTPService` is a qualified name, where the prefix `ls` was defined in the statement `xmlns:ls="http://www.iona.com/FixedBinding"` earlier in the same document, and `SOAPHTTPService` is a resource defined in the specification at that location.

QName interface

A programming interface that manages QNames in canonical format. For Java, Artix uses `javax.xml.namespace.QName`. For C++, Artix provides `IT_Bus::QName`.

R

RAR
Resource Adapter Archive (RAR), a compressed (.zip) file that contains the classes and other files required to run a J2EE Connector Architecture resource adapter.

reference
A self-contained object that fully describes a service. References can be passed between services or between a service and its consumers as operation parameters. Starting with release 4.0, Artix uses the endpoint reference format for references, as defined by the WS-Addressing standard. Previous versions of Artix used the Artix reference format.

reply
A message returned by a service to a service consumer in response to a request from that consumer. See also output message.

request
A message sent from a service consumer to a service asking for the service to perform an action. See also input message.

request-level handler
A Java message handler that processes messages between the Artix binding and the user's application code.

request-level interceptor
The equivalent of a request-level handler, but used with Artix C++.

request-response operation
One type of WSDL-defined abstract operation, in which the service endpoint receives a message and returns a correlated message. Request-response operations specify input message, output message, and fault message types. Artix WSDL-to-code generation tools support request-response operations.

resource adapter

A system-level software driver used by a J2EE application server to connect to an enterprise information system (EIS). The driver plugs into an application server and provides connectivity between the EIS, the application server, and the enterprise application. The Artix J2EE Connector is a resource adapter that connects J2EE to Artix.

response

See reply.

RMI

Remote Method Invocation (RMI), a Java API for performing remote procedure calls.

router

An Artix service that redirects messages based on rules defined in the router's contract. An Artix router can be used to bridge operation invocations between different communication protocols.

routing

The redirection of a message from one WSDL port to another. Artix supports the following types of routing defined in WSDL contracts:

- **Port based routing** (also known as topic based routing), which routes all messages on an inbound WSDL port to a single outbound WSDL port. This is useful for protocol conversion or proxy use cases because the overhead is minimal. For example, all messages that arrive on a URL can be forwarded to a single MQSeries queue.
- **Operation based routing** (also known as subject based routing), which routes different messages that arrive on the same WSDL port to different outbound WSDL ports. This is useful for creating unified facades for functionality implemented across different hosts. For example, for the Customer Service portType, CustomerSearch messages are sent to the mainframe using MQSeries, while TroubleTicket messages are sent using CORBA to a different host.
- **Context based routing**, which routes messages that arrive on the same WSDL port to different destinations based on values in the middleware headers. This allows for modifying application behavior based on

sender attributes. For example, messages can be sent to different servers based on the user-agent field of the HTTP header, which allows for optimizing implementations for different SOAP stacks in the client base. In another example, MQ messages where the Application Identity Data field is set to “sales” are routed to one host, and all other messages are sent to another host.

- **Failover routing**, which normally tries to route messages to one host, but then under fault or timeout conditions automatically tries other hosts. This is a simple form of fault tolerance that requires no failover server infrastructure. (More robust failover capabilities are provided by the Artix locator service.)
- **Fanout routing**, which routes messages to several hosts in parallel. This provides distribution list capabilities that are centrally manageable via WSDL changes, yet do not require a publish-subscribe server infrastructure. (More robust message distribution capabilities are provided by the Artix notification service.)
- **Content based routing**, which routes messages that arrive on the same WSDL port to different destinations based on the application data contained in the message. Such rules are based on XPath expressions, even when payloads are not XML data (and without converting to XML data). This allows for changing the message destination based on application requirements. For example, customers with gold-level support contracts can be routed to one host, while all other messages are routed to another host.

RPC

Remote Procedure Call (RPC), a protocol used by a program to request a service from a program located on another computer in a network.

A SOAP message binding is specified in a WSDL contract as either an RPC style or Document style binding.

S

screen scraping

A legacy data extraction technique in which one program extracts data from the output display of another program. Sometimes used with green screen terminals and mainframe applications, screen scraping is considered brittle because it relies on precise row and column placement of data. By contrast, Artix for z/OS uses BMS map driven techniques on-host, which is more resilient because BMS map sets contain metadata that is independent of the terminal screen output.

servant

A Java or C++ object that implements the service operations specified in a WSDL contract. See also static servant and transient servant.

server

A process in which one or more Artix servants can be created and registered to handle incoming operation requests through the Artix bus object.

service

A collection of operations that perform a useful set of functions in a network, access to which is implemented as an endpoint. In a service-oriented network, services are defined by a service contract.

service bus

The infrastructure that allows services and service consumers to interact in a distributed environment. The bus handles the delivery of messages between different middleware systems, and provides management, monitoring, and mediation services such as routing, service discovery, or transaction processing. Also known as an Enterprise Service Bus, or ESB. The Artix product as a whole is an example of a standards-based ESB.

service consumer

The end user of a service, also called a client for that service. This term is sometimes shortened to *consumer*.

service contract

See contract.

service element

An enclosing element in a WSDL contract that contains one or more `port` elements. Each port element maps a binding to the transport details necessary to contact the service.

service interface
See interface.

service intermediary
See intermediary.

service proxy
A stand-in object created by an Artix client that allows it to connect to a remote service. See also dynamic proxy.

service template
A WSDL service definition that serves as the model for the clones created for a transient servant. Service templates must fully define all of the details of the transport used by the transient servant, except its address. The address provided in the service template must be a wildcard value.

servlet
A Java program that extends the functionality of a web server by generating dynamic content and interacting with Web applications using a request-reply protocol.

session manager
A group of Artix plug-ins that work together to control the number of clients that can access a group of services concurrently. The session manager can be used to ensure a given instance is used by only one client at a time, which is useful for service-enabling single-threaded applications.

SOA
Service-Oriented Architecture (SOA), a loosely-coupled distributed architecture in which services make resources available to service consumers in a standardized way. SOA is language and protocol independent.

SOAP
Simple Object Access Protocol (SOAP), a protocol intended for exchanging structured information in a decentralized, distributed environment. It defines, using XML, an extensible messaging framework containing a message construct that can be exchanged over a variety of underlying transport protocols.

solicit-response operation

One type of WSDL-defined abstract operation, in which the service endpoint sends a message and receives a correlated message. Artix WSDL-to-code generation tools do not support solicit-response operations.

SSL

Secure Socket Layer (SSL), a security protocol that provides private communication over the Internet. The protocol allows client-server applications to communicate in a way that cannot be eavesdropped on or tampered with. SSL-compliant servers are always authenticated, and SSL clients are optionally authenticated. See also TLS.

SSL handshake

An exchange of messages that begins an SSL communication session. The handshake allows a server to authenticate itself to the client using public-key encryption, and then allows the client and the server to co-operate in the creation of symmetric keys that are used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server, which is known as mutual authentication.

static servant

A servant that, when registered, is associated with a service appearing explicitly in its defining WSDL contract. Static servants are thus restricted to using a service from the fixed collection of services appearing in the WSDL contract. Static servants are useful when an Artix bus instance is only going to host a single instance of a servant, or when using references without using the WSDL publishing plug-in. Compare with transient servant.

Stub interface

A Java standard interface, `javax.xml.rpc.Stub`. As required by the JAX-RPC specification, all Artix proxies implement this interface, which provides access to a number of low-level properties used in connecting the proxy to the service implementation.

substitution group

A feature of XML Schema that allows you to define groups of elements that may be used interchangeably in instance documents. For example, a *vehicle* head element might be defined with *automobile*, *boat*, and *airplane* substitution elements, any of which could be used wherever the *vehicle* element might be used. A substitution group is defined using the `substitutionGroup` attribute of the XML Schema element. See also abstract head element.

switch

See router.

T

tagged binding

An Artix WSDL extension used to communicate with applications that use self-describing, or delimited, messages.

TLS

Transport Layer Security (TLS), an open standard from the Internet Engineering Task Force that is based on, and is the successor to, SSL. TLS provides transport-layer security for secure communications. See also SSL.

transaction isolation level

The degree to which a database transaction is protected from actions by other transactions. The SQL standard specifies four isolation levels: read uncommitted, read committed, repeatable reads, and serializable.

transient servant

A servant whose physical details are cloned from a `port` definition in the contract that defines a service. Transient servants are useful when an Artix bus will host several instances of a servant, such as when a service is a factory for other services. Compare with static servant.

transport

A standards-based network protocol, such as HTTP or IIOP, that defines how objects communicate over a network. The transport details for an endpoint are specified inside the WSDL `port` element.

transport plug-in

An Artix plug-in module that provides wire-level interoperation with a specific type of middleware. When configured with a given transport plug-in, Artix interoperates with the specified middleware at a remote location or in another process. The transport is specified in the port element of an Artix contract.

type factory

A Java class generated to support the use of XML Schema `anyTypes` and SOAP headers in Java.

types element

The enclosing element in a WSDL contract that contains data type definitions using a type system such as XSD.

U

UDDI

Universal Description, Discovery, and Integration (UDDI), an industry initiative to create a platform-independent, open framework and registry for describing services, discovering businesses, and integrating business services using the Internet. UDDI specifies a mechanism for Web service providers to advertise the existence of their Web services and for Web service consumers to locate Web services of interest. For further information, see http://www.uddi.org.

W

W3C

World Wide Web Consortium (W3C), an international consortium where member organizations, a full-time staff, and the public work together to develop Web standards.

WS-A

Web Services Addressing (WS-A or WS-Addressing), a specification that provides transport-neutral mechanisms to address Web services and messages. See the WS-A specification.

WSDL

Web Services Description Language (WSDL), an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL is the language used to express service contracts.

WSDL is similar to IDL, type libraries, and other previous interface definition languages, but WSDL is extensible so that it can uniquely model a physical contract. For further information see the WSDL specification.

WS-RM

Web Services Reliable Messaging (WS-RM), a specification that describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. Obtain the specification from IBM or Microsoft.

WSS

Web Services Security (WSS), an OASIS specification that describes enhancements to SOAP messaging to provide a means for applying security to Web services. For further details, see the WSS specification.

X

XML Schema

A language specification by the W3C that defines an XML vocabulary for defining the contents and structure of XML documents. XML Schema is a successor to XML Document Type Declarations (DTDs), but is more expressive and better designed for expressing a type system. XML Schema is used as the native type system for Artix.

For further information, see the XML Schema specification.

XSD

XML Schema Definition (XSD), an instance of an XML schema written in the XML Schema language. An XSD defines a type of XML document in terms of constraints upon what elements and attributes may appear, their relationship to each other, and what types of data may be in them.

In Artix, a schema can be a standalone resource, or it can be used as an import to define the types within a WSDL contract.

XSL

Extensible Stylesheet Language (XSL), a language for expressing stylesheets. It consists of two parts: a language for transforming XML documents, and an XML vocabulary for specifying formatting semantics. For further information, see the XSL specification.

XSLT

XSL Transformations (XSLT), an XML-based language used for the transformation of XML documents into other forms. XSLT is the stylesheet language subset of the XSL specification. For further information, see the XSLT specification.

Index