



# Artix™

---

## Glossary

Version 4.2, March 2007

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

IONA, IONA Technologies, the IONA logos, Orbix, Artix, Making Software Work Together, Adaptive Runtime Technology, Orbacus, IONA University, and IONA XMLBus are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

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

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

---

#### COPYRIGHT NOTICE

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

Copyright © 1999-2007 IONA Technologies PLC. All rights reserved.

All products or services mentioned in this publication are covered by the trademarks, service marks, or product names as designated by the companies that market those products.

Updated: March 18, 2007

# Preface

## **What is Covered in this Book**

This book provides definitions for terms used in the Artix and Artix Registry/Repository documentation libraries, 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 Documentation Library**

For information on the organization of the Artix library, the document conventions used, and finding additional resources, see [Using the Artix Library](#).

## PREFACE

# Glossary

## Artix-specific glossary

This glossary defines terms in the context of the development and deployment of services using Artix, and in the context of governance of a SOA network using [Artix Registry/Repository](#). 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**

An Artix service that gives [service consumers](#) a mechanism to discover the runtime location of [service provider](#) endpoints.

**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 Orchestration**

An add-on package for Artix that allows you to coordinate the long-running interactions between two or more [service providers](#), making the orchestrated services appear as a single service. This [orchestration](#) is implemented in terms of the [BPEL](#) language.

**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 provider](#) 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 Registry/Repository**

A suite of tools for defining, specifying, modeling, persisting, and deploying the components of a [SOA network](#). Artix Registry/Repository includes a database into which the model of your SOA network is stored, GUI tools for visualizing and validating your network, and tools for applying [business rules](#) in the form of policies to your network's services.

**Artix session manager**

An Artix service implemented as group of Artix [plug-ins](#) that work together to limit or control concurrent access by [service consumers](#) to one or more [service providers](#). The session manager can be used to make sure that a given instance is used by only one client at a time, which is useful for service-enabling single-threaded applications.

**Artix transformer**

In Artix for Windows, Linux, and UNIX, the Artix transformer is 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.

**assembly**

In [Artix Registry/Repository](#), a logical set of [SOA network participants](#) ([services](#) and [consumers](#)), which performs some business function. An assembly is a unit of [deployment](#).

---

**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](#).

**business rule**

In general, an enterprise-specific statement that defines or constrains some aspect of business operation. Business rules are unique to each enterprise or organization.

In [Artix Orchestration](#), a business rule is an expression of an interaction between services, implemented as a [BPEL](#) construct.

Compare with [governance rule](#).

---

**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++.

**Celtix Enterprise**

A supported IONA [ESB](#) product, based on the open-source Apache Incubator CXF project, along with related open-source projects.

**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 make sure 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. In the context of a [SOA network](#), a client process is called a [consumer](#) or [service consumer](#).

**composite policy**

In [Artix Registry/Repository](#), a [policy](#) formed by combining two or more policies.

**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, you can provide different settings for individual buses, or provide common settings for groups of buses. Any individual 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](#) and [JEE](#) 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**

---

### **datatype**

An XML data type as defined in the [XML Schema](#) Definition language ([XSD](#)).

**deployment**

The process of propagating a [service](#) into an environment so that it is ready to use.

In Artix, deployment refers to the activation of development artifacts in an Artix [container](#), with the presumption that the artifacts have been distributed and are available locally to the container.

In [Artix Registry/Repository](#), deployment refers to the instantiation of a [service](#) associated with a [container](#) and an [installation](#).

**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 `wSDLtoC++`, `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.

**DOM**

Document Object Model (DOM), an API for accessing and manipulating XML documents as tree structures.

**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](http://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**

Enterprise Service Bus (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****governance**

The process of controlling and enforcing rules about a [SOA network](#).

**governance rule**

In [Artix Registry/Repository](#), a design-time rule that applies to a [SOA network](#), usually implemented as a [policy](#). For example: "apply the [WS-RM](#) specification across all Artix services in this network." Compare with [business rule](#).

**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****i18n**

An abbreviation for internationalization, used in the context of preparing products, especially software and documentation, for use in more than one national locale and language. The abbreviation is constructed from the first and last letters of internationalization, with 18 substituting for the number of letters between. Use of the abbreviation avoids the issue of American versus British spellings of the word. See also [i10n](#).

**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.

### **implementation**

In [Artix Registry/Repository](#), any unit of executable code or configuration that implements business logic. For example, an implementation might be a C++ binary or library, a Java mainline or library, a BPEL script, or a routing configuration.

### **IMS**

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

### **infrastructure**

In general, the basic, underlying framework or features of a system or organization. In [Artix Registry/Repository](#), an infrastructure is a named collection of [SOA network participants](#), [installations](#), [products](#), and [containers](#). For example, a site might have a test infrastructure and a production infrastructure.

### **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](#).

### **installation**

In [Artix Registry/Repository](#), a physical machine on which a [package](#) is deployed. An installation is described with attributes such as an installation name, the host name and its platform, the network pathname to any files, the enclosing [infrastructure](#), and the product type.

### **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 provider](#) 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. The J2EE specification for Java 1.4 was updated to [JEE](#) for Java 5.

**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.

### **JAXP**

Java API for XML Processing (JAXP), an API for processing XML documents. JAXP supports the [SAX](#), [DOM](#), and [XSLT](#) standards.

### **JAXR**

Java API for XML Registries (JAXR), an API for accessing an XML [registry](#).

### **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 4 bases its Java API and data type mappings. For further information, see <http://java.sun.com/xml/jaxrpc/overview.html>.

### **JAX-WS**

Java API for XML Web Services (JAX-WS), an open-source programming model based on a specification from Sun Microsystems. JAX-WS is a newly rearchitected API for Web services, and is designed to take the place of [JAX-RPC](#) in Web services and Web applications. JAX-WS is the standard on which [Celtix Enterprise](#) and future versions of Artix base their Java API and data type mappings.

### **JB1**

Java Business Integration (JB1), a specification for a standards-based, vendor-neutral architecture, based on [SOA](#) principles, for the integration of disparate applications, service providers, and service consumers. JB1-compliant components are expected to plug in and interoperate with other JB1-compliant components. This frees vendors to concentrate on supplying components that implement their particular area of expertise without worrying about implementing the other necessary portions of a complete solution. JB1 also frees end-users to pick and choose among many JB1-compliant components to assemble a SOA network sized to their needs, without locking in to one vendor's approach. The JB1 specification was developed by the Java Community Process. Compare with [SCA](#).

**JCP**

Java Community Process (JCP), a consortium of vendors who propose, review, and agree on standards and specifications for Java technologies. See [jcp.org](http://jcp.org).

**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.

**JEE**

Java Platform, Enterprise Edition 5 (JEE or JEE 5), a specification and toolkit from Sun Microsystems for the development and deployment of enterprise applications. JEE is the Java 5 version of [J2EE](#).

**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.

**JNI**

Java Native Interface (JNI), a standard programming interface for writing Java native methods and embedding the Java virtual machine (JVM) into native applications. The primary goal is binary compatibility of native method libraries across all JVM implementations on a given platform.

## 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

**l10n**

An abbreviation for localization, used in the context of preparing products, especially software and documentation, for use in more than one national locale and language. Localization is the process of translating the elements of a product for a particular locale and language. See also [i18n](#).

**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 provider](#) 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.

**middleware**

A software communications layer that manages the interaction of disparate applications across heterogeneous hardware and network environments.

**MQseries**

The former name of an IBM [middleware](#) technology that allows independent and potentially non-concurrent applications on a distributed system to communicate with each other. MQseries is currently known as [WebSphere MQ](#).

---

**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.



## O

**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.

---

**OASIS**

An international consortium that drives the development, convergence, and adoption of Web services standards. See [www.oasis-open.org](http://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](http://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 [service providers](#) 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 provider](#) 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

**package**

In [Artix Registry/Repository](#), a zip or tar file containing an [assembly](#) that is ready to deploy.

**packaging**

In [Artix Registry/Repository](#), the creation of a package for deployment.

**participant**

A member of a SOA network, whether [service provider](#), [service consumer](#), or [intermediary](#).

**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.

**policy**

A collection of configuration settings applied to a [participant](#) in a [SOA network](#) that results in a defined behavior. For example, a security policy applied to a [service provider](#) might specify that any connection from a [service consumer](#) must meet minimum security standards or be rejected. In [Artix Registry/Repository](#), a policy is a means of implementing a [governance rule](#).

**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 the [services](#) 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](#).

**product**

In [Artix Registry/Repository](#), an attribute of an [installation](#) that identifies the vendor product name and version number that implements a given [participant](#). For example, the product attribute of an installation might be shown as Artix 4.2.

**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 of the specification where that resource is defined.

QNames are composed of:

- A URI representing the namespace of the resources's definition.
- The name of the resource, usually called the localPart.
- Some QName formats also include 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` is 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

In Artix, 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.

In [Artix Registry/Repository](#), any association between a [service provider](#), [service consumer](#), or [intermediary](#).

### registry

An infrastructure that enables the building, deployment, and discovery of Web services.

**reply**

A message returned by a [service](#) to a [service consumer](#) in response to a request from that consumer. See also [output message](#).

**repository**

In general, a central place where data is stored and maintained.

In [Artix Registry/Repository](#), a repository is a comprehensive system that describes all aspects of a [SOA network](#). It acts as a centrally-managed data store for hosts, [installations](#), [containers](#), [service providers](#), [service consumers](#), and [governance rules](#), and provides an audit trail of change events.

**request**

A message sent from a [service consumer](#) to a [service provider](#) 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](#).

**REST**

Representational State Transfer (REST), a model for Web services based solely on HTTP. REST takes the view that the Web already has everything necessary for Web services, without having to add extra specifications such as SOAP and UDDI. The theory holds that any object can be represented and made available at a URI, and, subject to the necessary permissions, can be fully manipulated using one of the four simple HTTP verbs: GET, PUT, POST, and DELETE.

**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

### SAAJ

SOAP with Attachments API for Java (SAAJ), an API for creating and populating a SOAP message.

**SAX**

Simple API for XML (SAX), an event-driven Java interface in which the parser invokes one of several methods supplied by the caller when a parsing event occurs. Events include recognizing an XML tag, finding an error, encountering a reference to an external entity, or processing a DTD specification.

**SCA**

Service Component Architecture (SCA), is a set of specifications that describe a model for building applications and systems using a Service-Oriented Architecture. SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services. SCA is developed by a consortium of companies, including IONA Technologies. Compare with [JBI](#).

**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](#). The more exact term in the context of a service-oriented network is [service provider](#).

**service bus**

The infrastructure that allows [service providers](#) 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**

In [Artix Registry/Repository](#), the WSDL [interface](#) to a [service](#) that contains only logical elements (type, [operation](#), [message](#), and [portType](#)), and which may not contain physical elements ([binding](#), [port](#), or [service](#)).

**service intermediary**

See [intermediary](#).

**service-level agreement**

See [SLA](#).

**service provider**

A [contract](#)-defined collection of [operations](#) that perform a useful set of functions in a network, access to which is implemented as an [endpoint](#). This term is often shortened to [service](#).

**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**

See [Artix session manager](#).

**SLA**

Service-level agreement (SLA), the portion of a service contract in which a certain level of service is agreed. The agreed level of service varies widely by service type, but might include items such as the percentage of server uptime, or the average time to resolve an issue.

**SOA**

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

**SOA governance**

The process of controlling and enforcing rules about a [SOA network](#).

**SOA infrastructure**

See [infrastructure](#).

**SOA lifecycle**

The stages involved in the overall design, development, testing, deployment, and management of reusable Web service applications.

**SOA network**

A collection of reusable [service providers](#), [service consumers](#), [implementations](#), and [assemblies](#) on a network.

**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.

**SOA repository**

A centrally-managed data store for all [SOA governance](#) policies and metadata.

**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](#).

**STP**

A top-level [Eclipse](#) project dedicated to providing a generic, extensible, standards-based tool platform for producing [SOA](#) applications.

**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 interoperability 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.

**WebSphere MQ**

The current name of an IBM network [middleware](#) technology that allows independent and potentially non-concurrent applications on a distributed system to communicate with each other. WebSphere MQ was formerly known as [MQseries](#).

**WS-Addressing**

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

## GLOSSARY

# Index

## A

- abstract contract 5
- abstract head element 5
- anyType 5
- APPC 5
- application server 6
- ART 6
- Artix bus 6
- Artix chain builder 6
- Artix Designer 6
- Artix locator 7
- Artix message context 7
- Artix Orchestration 7
- Artix reference 7
- Artix Registry/Repository 7
- Artix session manager 7
- Artix transformer 8
- assembly 8

## B

- binding 8
- binding element 8
- BMS map sets 8
- BMS parser 8
- BPEL 9
- bridge 9
- bus 9
- business rule 9

## C

- CDT 9
- Celtix Enterprise 9
- choice complex type 9
- CICS 9
- classloader 10
- classloader firewall 10
- client 10
- composite policy 10
- concrete contract 10
- configuration domain 10
- configuration file 10
- configuration scope 11

- connection 11
- connection factory 11
- consumer 11
- container 11
- contract 11
- CORBA 12
- CORBA naming service 12
- CSlv2 12

## D

- datatype 12
- deployment 13
- deployment descriptor 13
- discriminator 13
- DOM 13
- dynamic proxy 13

## E

- EAI 14
- EAR file 14
- Eclipse 14
- EIS 14
- EJB 14
- EMS 14
- endpoint 14
- endpoint reference 15
- enterprise service 15
- EPR 15
- ESB 15
- EXCI 15

## F

- facet 15
- factory pattern 15
- fault element 15
- fault message 16
- firewall classloader 16
- fixed binding 16
- FML 16

## G

- governance 16

governance rule 16  
green screen 16

**H**

Handler 17  
high availability 17  
host 17

**I**

i18n 17  
IDL 17  
IIOP 17  
implementation 18  
IMS 18  
infrastructure 18  
input element 18  
input message 18  
installation 18  
interceptor 18  
interface 19  
intermediary 19  
IOR 19

**J**

J2EE 19  
J2EE Connector Architecture 19  
JAXB 19  
JAXP 20  
JAXR 20  
JAX-RPC 20  
JAX-WS 20  
JBI 20  
JCP 21  
JDBC 21  
JDT 21  
JEE 21  
JMS 21  
JMX 21  
JNDI 21  
JNI 21

**K**

Knowledge Module 22

**L**

l10n 22  
list type 22

location domain 22  
locator 22  
locator daemon 22  
logical contract 23  
login service 23

**M**

marshaling 23  
message 23  
message context 23  
message element 23  
message handler 24  
message-level handler 24  
message-level interceptor 24  
MFS maps 24  
MFS parser 24  
middleware 24  
MQseries 24

**N**

naming service 24  
nillable 24  
notification operation 25

**O**

OASIS 25  
object reference 25  
OMG 25  
one-way operation 25  
operation 25  
operation element 26  
ORB 26  
orchestration 26  
OTMA 26  
output element 26  
output message 26

**P**

package 27  
packaging 27  
participant 27  
payload format 27  
peer manager 27  
physical contract 27  
plug-in 28  
POA 28  
policy 28

- port 28
- port element 28
- portType 28
- portType element 29
- product 29
- proxification 29
- proxy 29
- proxy object 29

**Q**

- QName 29
- QName interface 30

**R**

- RAR 30
- reference 30
- registry 30
- reply 31
- repository 31
- request 31
- request-level handler 31
- request-level interceptor 31
- request-response operation 31
- resource adapter 31
- response 31
- REST 32
- RMI 32
- router 32
- routing 32
- RPC 33

**S**

- SAAJ 33
- SAX 34
- SCA 34
- screen scraping 34
- servant 34
- server 34
- service 34
- service bus 35
- service consumer 35
- service contract 35
- service element 35
- service interface 35
- service intermediary 35
- service-level agreement 35
- service provider 35
- service proxy 36

- service template 36
- servlet 36
- session manager 36
- SLA 36
- SOA 36
- SOA governance 36
- SOA infrastructure 36
- SOA lifecycle 36
- SOA network 37
- SOAP 37
- SOA repository 37
- solicit-response operation 37
- SSL 37
- SSL handshake 37
- static servant 38
- STP 38
- Stub interface 38
- substitution group 38
- switch 38

**T**

- tagged binding 38
- TLS 38
- transaction isolation level 39
- transient servant 39
- transport 39
- transport plug-in 39
- type factory 39
- types element 39

**U**

- UDDI 39

**W**

- W3C 40
- WebSphere MQ 40
- WS-Addressing 40
- WSDL 40
- WS-RM 40
- WSS 40

**X**

- XML Schema 41
- XSD 41
- XSL 41
- XSLT 41

