Artix™ ESB

Using the Artix Library

Version 5.1, December 2007

Making Software Work Together™

Updated: May 7, 2008

Contents

CHAPTER 1

Artix Library Overview

This chapter describes the contents of the Artix Library, how to get additional information, and the documentation conventions used.

In this chapter

This chapter includes the following topics

Artix Documentation Library

Overview

The Artix documentation library is organized into the following sections:

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

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.
- Using the Artix Library (this book) introduces the Artix documentation library, explains its conventions, and provides suggested reading paths.
- Getting Started with Artix describes basic Artix and WSDL concepts, and shows a simple example application.
- Artix Online Help Infocenter describes how to use the Artix Designer GUI tools 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 discuss how to use 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 SOA 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 WSDL contract. Special attention is paid to Artix-specific WSDL extensions.
- Artix Bindings and Transports, C++ Runtime describes the WSDL extensions used to define payload formats and transports for Artix services written in C++ or JAX-RPC.
- Artix Bindings and Transports, Java Runtime describes the Artix WSDL extensions used to define payload formats and transports for Artix services written in JAX-WS or JavaScript.

Developing Artix Solutions

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

- Developing Artix Applications in C++ explains how to implement services using the Artix C++ API.
- Developing Advanced Artix Plug-ins in C++ explains how to implement advanced Artix plug-ins (for example, interceptors) using the Artix C++ API.
- Developing Artix Applications with JAX-RPC explains how to implement services using the Artix Java API for XML-Based Remote Procedure Call.
- Developing Artix Applications with JAX-WS explains how to implement services using the Artix Java API for XML-Based Web Services.
- Developing Artix Applications with JavaScript explains how to implement services using the Artix JavaScript API.
- Artix WSDLGen Guide explains how to generate C++, JAX-RPC, and JAX-WS code using the Artix scripting tools.
- Developing Artix Database Services explains how to expose databases as Web services using Artix command-line tools and Artix Designer GUI tools. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.

Configuring and Managing Artix Solutions

This section includes:

- Configuring and Deploying Artix Solutions, C++ Runtime explains how to set up your Artix environment and how to configure and deploy Artix services written in C++ or JAX-RPC.
- Configuring and Deploying Artix Solutions, Java Runtime explains how to set up your Artix environment and how to configure and deploy Artix Java services (for example, written in JAX-WS or JavaScript).
- Managing Artix Solutions with JMX, C++ Runtime explains how to monitor and manage Artix services using Java Management Extensions. This guide applies to services written in C++ or JAX-RPC.
- Managing Artix Solutions with JMX, Java Runtime explains how to monitor and manage Artix Java services using Java Management Extensions. This applies to services written in JAX-WS or JavaScript.

For information on using third-party management systems, see Integrating with Management Systems.

Using Artix Services

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

- Artix Router Guide, C++ Runtime explains how to integrate and manage services using the Artix C++ router. This guide applies to services written in C++ or JAX-RPC.
- Artix Java Router Getting Started introduces the Artix Java router and describes how to create, build and run a simple example. This guide applies to services written in JAX-WS.
- Artix Java Router Deployment Guide explains how to deploy a standalone Artix Java router, and how to deploy into the Spring container. This guide applies to services written in JAX-WS.
- Artix Locator Guide explains how clients can find services using the Artix locator. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.
- Artix Session Manager Guide explains how to manage client sessions using the Artix session manager. This guide applies to services written in C++ or JAX-RPC.

- Artix C++ Transactions Guide explains how to enable Artix applications written in C++ to participate in transacted operations.
- Artix JAX-RPC Transactions Guide explains how to enable Artix applications written in JAX-RPC to participate in transacted operations.
- Artix Security Guide explains how to configure and develop secure Artix applications. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.

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. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.
- Artix for J2EE provides information on using Artix to integrate with J2EE applications. This guide applies to services written in JAX-RPC.

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 third-party enterprise and SOA management systems:

- IBM Tivoli Integration Guide explains how to integrate Artix services with the IBM Tivoli enterprise management system. This guide applies to services written in C++ or JAX-RPC.
- BMC Patrol Integration Guide explains how to integrate Artix services with the BMC Patrol enterprise management system. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.
- CA-WSDM Integration Guide explains how to integrate Artix services with the CA-WSDM SOA management system. This guide applies to services written in C++ or JAX-RPC.
- AmberPoint Integration Guide explains how to integrate Artix services with the AmberPoint SOA management system. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.
- Progress Actional Integration Guide explains how to integrate Artix services with the Progress Actional SOA management system. This guide applies to services written in JAX-WS or JavaScript.

Artix 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, C++ Runtime
- Artix Configuration Reference, Java Runtime
- Artix WSDL Extension Reference
- Artix JAX-RPC API Reference
- Artix JAX-WS API Reference
- Artix Security API Reference
- Artix C++ API Reference
- Artix .NET API Reference
- WSDLGen Java API Reference
- WSDLGen JavaScript API Reference
- Artix Glossary

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.

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

You can access the online help the following different ways:

- 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.
- See the Artix Infocenter available online.

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**.

Related documentation

For information on related IONA products, see the following documentation:

- Artix Registry/Repository
- Artix Mainframe
- Artix Orchestration
- Artix Data Services
- Artix Connect

Getting the Latest Version

The latest updates to the Artix documentation library 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/5.1/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.

Additional Resources

The IONA Knowledge Base contains helpful articles written by IONA experts about Artix and other products.

The IONA Update Center 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.

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

Documentation Conventions

Overview

This section shows the typographical and keying conventions used by the Artix documentation library.

Typographical conventions

The Artix library 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

The Artix library 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**).

CHAPTER 2

Suggested Reading Paths

This chapter describes suggested reading paths for different types of Artix users.

In this chapter

This chapter includes the following topics

SOA Architects

Overview

This section describes a suggested reading path for SOA architects, and includes suggestions for background reading.

SOA architect path

SOA architects should start with the following:

1. Building SOA with Artix presents an overview of SOA and ESBs, of how Artix fits into SOA, and of how Artix works.
2. Installation Guide. You must read the following sections about supported environments:
   i. Supported Systems and Compilers
   ii. Java, Compiler, and Artix Designer Requirements
3. Writing Artix Contracts includes the following information about basic WSDL concepts and how to write a service interface:
   i. *Introduction*. Overview of WSDL, the structure of a contract, and the steps involved in writing a service contract.
   ii. *Designing Logical Data Units*. How to create data types using XML Schema.
   iii. *Defining Logical Messages Used by a Service*. How to build the data types into the messages that a service will use to implement its operations.
   iv. *Defining Your Logical Interfaces*: How to create a service interface using the logical messages.
4. Artix Bindings and Transports, C++ Runtime describes the WSDL extensions used to define payload formats and transports for Artix services written in C++ or JAX-RPC:
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, Fixed, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, Tuxedo, or JMS).

5. Artix Bindings and Transports, Java Runtime describes the Artix WSDL extensions used to define payload formats and transports for Artix services written in JAX-WS or JavaScript.
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, CORBA, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, MQ, or JMS).
6. Artix Infocenter explains how to use the Eclipse-based Artix Designer GUI tools to design WSDL service contracts and to generate C++, JAX-RPC, and JAX-WS implementation code.

Background reading

In addition, the following publications provide useful background information on Web services, XML, and WSDL:

- *Understanding Web Services: XML, WSDL, SOAP, and UDDI*, by Eric Newcomer
- *Understanding SOA with Web Services*, by Eric Newcomer and Greg Lomow
- W3Schools online tutorials
  (see http://www.w3schools.com)
  - XML tutorial (http://www.w3schools.com/xml/default.asp)
  - XSD tutorial (http://www.w3schools.com/schema/default.asp)
  - XSLT tutorial (http://www.w3schools.com/xsl/default.asp)
- The W3C XML schema page
  (see www.w3.org/XML/Schema)
- THE W3C WSDL specification
  (see www.w3.org/TR/wsdl)

Administrators

Overview

This section describes a suggested reading path for Artix administrators in both C++ and Java runtime environments.

All Artix administrators

Administrators can approach the Artix library as follows:

1. Installation Guide describes all the prerequisites and procedures for installing Artix on supported systems. You must read the following:
   i. *Supported Systems and Compilers*
   ii. *Java, Compiler, and Artix Designer Requirements*
   iii. *Installing Artix*

C++ runtime administrators

Administrators working with applications written in C++ or JAX-RPC should read the following:

1. Configuring and Deploying Artix Solutions, C++ Runtime explains how to configure and deploy Artix services written in C++ or JAX-RPC. It explains Artix configuration, how to find contracts that control your Artix services, and how to deploy Artix applications. You should start with the following chapters:
   i. *Getting Started* explains how to set up an Artix environment using the `artix_env` script, and Artix system environment variables.
   ii. *Artix Configuration* explains concepts such as Artix `.cfg` configuration files, scopes, namespaces, and variables.
   iii. *Artix Logging* explains how to configure Artix logging for services and Artix subsystems.
   iv. *Deploying Services in an Artix Container* explains how to deploy and manage C++ and Java services using an Artix container.
   v. *Deploying High Availability* explains how to configure and deploy high availability in Artix, which is based on Berkeley DB.

vi. *Deploying Reliable Messaging* explains how to configure and deploy WS-RM and WS-Addressing for services in an Artix runtime environment.

vii. *Publishing WSDL Contracts* explains how to publish WSDL files that correspond to specific Web services, and enable clients to access the WSDL file and invoke on the service.

viii. *Accessing Contracts and References* shows how to specify the location of WSDL contracts and references in a configuration file and on the command line.

2. Artix Configuration Reference, C++ Runtime provides a comprehensive reference for the all configuration variables in an Artix configuration domain.

Java runtime administrators

Administrators working with applications written in JAX-WS or JavaScript should read the following:

1. Configuring and Deploying Artix Solutions, Java Runtime explains how to configure and deploy Artix Java services (for example, JAX-WS or JavaScript). You should start with the following chapters:

   i. *Getting Started* explains how to set your Artix Java runtime system environment.

   ii. *Artix Java Configuration* introduces the main concepts and components in the Artix Java runtime configuration.It also explains how to use Artix Java configuration files to manage your applications.

   iii. *Deploying to the Spring Container* outlines how to deploy and manage an Artix endpoint in the Spring container.

2. Artix Configuration Reference, Java Runtime provides a comprehensive reference for Artix configuration settings in Spring XML configuration files.

Security and management

Security administrators and administrators using management consoles should read the following:

1. Artix Security Guide provides detailed information on Artix security configuration and management. This guide applies to services written in C++, JAX-RPC, JAX-WS, or JavaScript.
2. Managing Artix Solutions with JMX, C++ Runtime explains how to monitor and manage Artix services using Java Management Extensions. This guide applies to services written in C++ or JAX-RPC.
3. Managing Artix Solutions with JMX, Java Runtime explains how to monitor and manage Artix Java services using Java Management Extensions. This applies to services written in JAX-WS or JavaScript.
4. Administrator's using third-party management tools should see the following guides for information on how to integrate Artix with third-party enterprise and SOA management systems:
   - Progress Actional Integration Guide
   - AmberPoint Integration Guide
   - BMC Patrol Integration Guide
   - CA WSDM Integration Guide
   - IBM Tivoli Integration Guide

Background reading

For background information on Web services, XML, and WSDL, see "Background reading" on page 17.

All Service Developers

Overview

This section describes an initial reading path for all types of service development use case. You should follow this path before writing any code.

All developers

All service developers should read the following path:

1. Building SOA with Artix presents an overview of SOA and ESBs, of how Artix fits into SOA, and of how Artix works.
2. Artix Installation Guide. You must read the following sections about supported environments:
   i. *Supported Systems and Compilers*
   ii. *Java, Compiler, and Artix Designer Requirements*
3. Writing Artix Contracts includes information about basic WSDL concepts and how to write a service interface.
   i. *Introduction*. Overview of WSDL, the structure of a contract, and the steps involved in writing a service contract.
   ii. *Designing Logical Data Units*. How to create data types using XML Schema
   iii. *Defining Logical Messages Used by a Service*. How to build the data types into the messages that a service will use to implement its operations.
   iv. *Defining Your Logical Interfaces*: How to create a service interface using the logical messages.
4. Artix Infocenter explains how to use the Eclipse-based Artix Designer GUI tools to design WSDL service contracts and to generate C++ and Java implementation code.

5. Configuring and Deploying Artix Solutions, C++ Runtime explains how to configure and deploy Artix services written in C++ or JAX-RPC. You must read the following chapters:
   i. *Getting Started* explains how to set up an Artix environment using the `artix_env` script, and system environment variables.
   ii. *Artix Configuration* explains concepts such as Artix configuration files, scopes, namespaces, and variables.
   iii. *Artix Logging* explains how to configure Artix logging for services and Artix subsystems.
6. Configuring and Deploying Artix Solutions, Java Runtime explains how to configure and deploy Artix Java services (for example, written in JAX-WS or JavaScript). You must read the following chapters:
   i. *Getting Started* explains how to set your Artix Java runtime system environment.
   ii. *Artix Java Configuration* introduces the main concepts and components in the Artix Java runtime configuration.It also explains how to use Artix Java configuration files to manage your applications.
   iii. *Deploying to the Spring Container* outlines how to deploy and manage an Artix endpoint in the Spring container.

Integration Use Case

Overview

This section describes the reading path for developing a service as a front-end for existing functionality.

Service integration

Service integrators should read the following books:

1. Artix Technical Use Cases. Read the following chapter:
   i. *Web Service Enabling Backend Services*. Walks through the steps for the integration use case.
2. Artix Router Guide, C++ Runtime. Read the following information about the C++ router service and how to make routes.
   i. *Introduction*. Overview of the C++ router and how it is used.
   ii. *Compatibility of Ports and Operations*. Explains the requirements for routing between interfaces.
   iii. *Creating Routes Using Artix Tools*. Introduces the GUI and command line tools that can be used to create routes.
3. Artix Java Router Getting Started. Read the following information about the C++ router service and how to make routes.
   i. *Introduction*. Overview of the Java router and how it is used.
   ii. *Tutorial*. Describes the Java router in more detail, explains the code for a sample application, and how to build and run the application.
   iii. *Defining Routes in Java DSL*. Explains how to define routing rules in Java in a domain specific language (DSL). This is the most flexible way to define rules.
   iv. *Defining Routes in XML*. Explains how to define routing rules in XML. This is not as flexible as Java DSL, but is easy to reconfigure at runtime.

4. Artix Bindings and Transports, C++ Runtime includes information about creating bindings and endpoints for Artix services written in C++ or JAX-RPC:
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, Fixed, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, Tuxedo, or JMS).
5. Artix Bindings and Transports, Java Runtime includes information about creating bindings and endpoints for Artix services written in JAX-WS or JavaScript.
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, CORBA, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, MQ, or JMS).
6. Artix Router Guide, C++ Runtime. Read the following information about how to define routes between endpoints:
   i. *Creating a Basic Route*. This is required reading. It describes the minimum needed to create a route in Artix.
   ii. *Adding Operation-Based Rules to a Route*: This is optional. It expands on basic route design.
   iii. *Adding Attribute-Based Rules to a Route*. This is optional. It expands on basic route design.
   iv. *Adding Content-Based Rules to a Route*. This is optional. It describes how to create content based routes.
   v. *Linking Routes*. This is optional. It expands on previous chapters.
   vi. *Using Advanced Routing Features*. This is optional. It describes how to create routes for various advanced use cases such as load balancing and service fail-over.
   vii. *Deploying an Artix Router*. This is required reading. It describes how to deploy the router in an Artix runtime environment.

7. Artix Java Router Deployment Guide. Read the following information about how to deploy an Artix Java Router:
   i. *Deploying a Standalone Router*. Explains how to deploy the Java router in standalone mode. This means you can deploy the router independent of any container, but some extra programming steps are required.
   ii. *Deploying into a Spring Container*. Explains how to deploy the Java router into a Spring container. This enables you to specify routing rules in an XML configuration file.
   iii. *Components*. Provides a reference of components available with the Artix Java router. These are plug-ins that can be used to enable integration with different kinds of protocol, containers, databases, and so on.

Advanced integration topics

In addition, you may wish to read the following:

- Artix for CORBA contains detailed information about using Artix to integrate with CORBA applications.
- Artix for J2EE contains detailed information about using Artix with J2EE applications.
- Developing Artix Database Services contains detailed information on how to integrate databases with Artix Web services. You can use Artix command-line or GUI tools to generate a database Web service.

New Development Use Cases

Overview

This section describes reading paths for the following new development use cases:

- "Service consumer"
- "C++ development"
- "JAX-RPC development"
- "JAX-WS development"
- "JavaScript development"

Service consumer

Read the following if you are developing a new service consumer:

1. Artix Technical Use Cases. Read the following chapter:
   i. *Building a Client for a Web Service*. Provides a walk through of the service consumer use case.
2. Artix Bindings and Transports, C++ Runtime includes information about creating bindings and endpoints for Artix services written in C++ or JAX-RPC:
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, Fixed, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, Tuxedo, or JMS).
3. Artix Bindings and Transports, Java Runtime includes information about creating bindings and endpoints for Artix services written in JAX-WS or JavaScript.
   i. Read the relevant binding chapter that applies to your system (for example, SOAP, CORBA, or XML).
   ii. Read the relevant transport chapter that applies to your system (for example, HTTP, MQ, or JMS).

C++ development

For detailed information on developing a new C++ service provider or consumer, read the following:

1. Developing Artix Applications in C++:
   i. *Getting Started with Artix Programming*. An overview of how a developer works in the Artix C++ development environment.
   ii. *Artix Programming Considerations*: *Operations and Parameters* section. An overview of how WSDL is mapped into C++.
   iii. *Server Programming*. The basics of developing an Artix C++ service.
   iv. *Client Programming*. The basics of developing an Artix C++ consumer.
   v. *Artix Programming Considerations*: *Compiling and Linking an Artix Application* section. What is needed to build Artix C++ applications.
   vi. *Artix Programming Considerations*: *Building Artix Stub Libraries on Windows* section. How to build Artix stub code into a Windows DLL.
   vii. *Artix Data Types*. Overview of how WSDL types are mapped into C++.
   viii. *Artix Programming Considerations*: *Exceptions* section. Overview of how to create and handle exceptions in Artix C++.
   ix. *Artix Programming Considerations: Locating Services with UDDI* section. How to use the UDDI interface as an alternate method of finding services.
   x. *Endpoint References and Callbacks*. Overview of EPRs and how to use them in implementing callbacks.
   xi. *Artix Contexts*. How to get information from the binding and transport layers of the runtime.
   xii. *Working with Transport Attributes*. How to extract a default set of transport information from the runtime.
   xiii. *Persistent Maps*. How to use persistent data for high availability.
   xiv. *Reflection*. How to determine the structure of an Artix data type without advance knowledge.

xv. *Default Servants*. How to write a scalable factory pattern.

xvi. *Artix Programming Considerations: Multi-Threading* section. Describes issues for multi-threaded Artix clients and servers.

Advanced C++ development

For detailed information on developing new advanced C++ plug-ins, read the following:

2. Developing Advanced Artix Plug-Ins with C++. How to write custom interceptors and transport plug-ins.

JAX-RPC development

For detailed information on developing a new JAX-RPC service provider or consumer, read the following:

1. Developing Artix Applications in Java:
   i. *The Artix Java Development Model*. An overview of the Artix Java development process. This includes a section on WSDL to Java mapping.
   ii. *Developing Artix Services*. The basics for developing and building a service in Artix.
   iii. *Developing Artix Consumers*. The basics for developing and building a consumer in Artix.
   iv. *Finding Contracts and References at Runtime*. How to use the Java APIs to locate contracts.
   v. *Things to Consider when Developing Artix Applications: Getting a Bus* section. How to get access to a bus reference from the runtime.
   vi. *Working with Artix Data Types*. Overview of XML schema to Java type mapping for most data types.
   vii. *Creating User-Defined Exceptions*. How to create and handle exceptions in Artix.
   viii. *Using Endpoint References*. Details of working with EPRs.
   ix. *Using Native XML*. How to develop Java applications that work with pure XML data.
   x. *Working with Artix Type Factories*. How to create type factories. Type factories are needed to work with features that follow.

xi. *Using Message Contexts*. Describes the Artix implementation and extension of the JAX-RPC `MessageContext` interface. This enables you to pass/receive information from the lower-levels of the Java runtime including the binding, transport, and handler layers.

xii. *Working with Transport Attributes*. Details about the Artix provided transport attribute types.

xiii. *Sending Message Headers*. Details about using the `MessageContext` to send message headers over SOAP and CORBA.

xiv. *Writing Handlers*. Overview of writing JAX-RPC handler objects. Handler objects are like interceptors in that work on a message as it passes through the Artix runtime.

xv. *Manipulating Messages in a Handler*. Details about altering the contents of requests or responses as they pass through the Handler objects in a message chain.

xvi. *Instrumenting a Service*. Details about adding JMX instrumentation to a service implementation.

xvii. *Using Persistent Datastores*. Details how to use persistent data in Artix Java applications.

xviii. *Using the Call Interface for Dynamic Invocations*. Details about writing clients that can invoke operations on a service for which it only has WSDL.

xix. *Using Substitution Groups*. Substitution groups are an advanced XML Schema construct.

xx. *Working with XML Schema anyTypes*. An `anyType` is the XML equivalent of a CORBA `Any`.

Advanced JAX-RPC topics

For detailed information on developing Java advanced plug-ins (for example, transport plug-ins), read the following:

2. Developing Artix Applications in Java:
   i. Using Artix Classloader Environments
   ii. Developing Plug-Ins
   iii. Developing Custom Artix Transports
   iv. Configuring Artix Plug-Ins

JAX-WS development

For detailed information on developing a new JAX-WS service provider or consumer, read the following:

1. Developing Artix Applications with JAX-WS:
   i. *Starting from Java Code.* Describes how to write a JAX-WS application without WSDL.
   ii. *Service Enabling a Java Class*. Describes how to annotate a Java class for use as a service provider. This includes creating the SEI, annotating the code, and generating WSDL.
   iii. *Developing a Consumer without a WSDL Contract*. This includes creating a service object, adding a port to a service, getting a proxy for an endpoint, and implementing the consumer's business logic.
   iv. *Starting from a WSDL Contract*. Describes how to write a JAX-WS service starting from WSDL. This includes developing a service and a consumer starting from WSDL.
   v. *Publishing a Service*. Describes how to publish a service provider as a standalone Java application.
   vi. *Developing RESTful Services*. Describes what it means for a service to be RESTful, and how to build RESTful services using Java classes and annotations.
   vii. *Developing Asynchronous Applications*. Describes how to use the JAX-WS asynchronous APIs to develop asynchronous service consumers.
   viii. *Generating the Stub Code*. Describes how to create a customization file to alter the code generated to use the asynchronous APIs.
   ix. *Implementing an Asynchronous Client with the Polling Approach*. Describes how to use the APIs that provide a mechanism for using the generated response object to pole for an asynchronous response.

x. *Implementing an Asynchronous Client with the Callback Approach*. Describes how to use a callback object to process asynchronous responses.

xi. *Using Raw XML Messages*. Describes how to obtain direct access to raw XML message data on the wire. The JAX-WS client-side interface is `Dispatch`, and the server-side interface is `Provider`.

xii. *Working with Contexts*. Describes how to access the metadata passed by contexts along the messaging chain. This metadata can be accessed by implementation code and by JAX-WS handlers that operate on the message below the implementation level.

JavaScript development

For information on developing a new JavaScript service provider or consumer, read the following:

1. Developing Artix Applications with JavaScript:
   i. *Implementing a service provider using JavaScript*. This includes defining the metadata, which describes how to provide the information provided by the JAX-WS annotations. It also includes implementing the application logic, which is a brief overview of implementing the service using the invoke property
   ii. *Implementing a service provider using E4X*. Describes the minor differences between the JavaScript interface and the ECMAScript for XML (E4X) interface.
   iii. *Deploying a JavaScript service*. Describes how to deploy scripted services using the `ServerApp` Java application provided.

Background reading

For background information on Web services, XML, and WSDL, see "Background reading" on page 17.