Artix 5.6.3

Release Notes: Java

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

http://www.microfocus.com

2015-03-12

Contents

Artix 5.6.3 for Java Release Notes

These release notes contain information about the Artix 5.6.3 release from Micro Focus. They contain information that might not appear elsewhere in the documentation. Read them in their entirety before you install the product.

Artix 5.6.3 has been rebranded as a Micro Focus product.

New and Changed Features

The following features are new in Artix 5.6.3:

- Apache components upgrade
- JAX-WS 2.2 support
- New annotations for "Java first" use cases
- API changes
- Runtime changes

Apache components upgrade

Artix 5.6.3 for Java has been upgraded from using Apache CXF version 2.3.5 to version 3.0.3, and from Apache Camel 2.6.0 to 2.14.1. See Migrating from Artix Java 5.6.2 to Artix Java 5.6.3 for information on any potential migration issues.

JAX-WS 2.2 support

Artix Java 5.6.3 provides support for JAX-WS 2.2 and JAXB 2.2.

When considering JAX-WS/JAXB support, note that the relevant JDKs have different specification and implementations of these standards built into the JDK by default. For example, JDK 6 contains JAX-WS 2.1 and JAXB 2.1, whereas JDK 7 contains JAX-WS 2.2 and JAXB 2.2.

In order to work with JAX-WS 2.2 and JAXB 2.2 with JDK 6, the Java endorsement overriding mechanism must be used. In Artix, `<`*`ARTIX_INSTALL`*`>/lib/cxf/endorsed` already contains versions 2.2 of the necessary jars. So for JDK6, you must set the following system property:

```
-Djava.endorsed.dirs=<ARTIX_INSTALL>/lib/cxf/endorsed.
```

Alternatively, you can simply copy the `geronimo-jaxws_2.2_spec-1.0` and `jaxb-api-2.2.1` jars to the `jre/lib/endorsed` directory of your JDK 1.6 installation.

New annotations for "Java first" use cases

Artix 5.6.3 for Java adds the following annotations:

- `@WSDLDocumentation` annotation to add documentation nodes to the generated WSDL

- `@SchemaValidation` annotation to turn on schema validation
- `@DataBinding` to set the databinding used (if other than JAXB)
- `@GZIP` to turn on GZIP compression
- `@FastInfoset` to turn on FastInfoset support
- `@Logging` to turn on and control various logging functionality
- `@EndpointProperty` to configure endpoint properties
- `@Policy` to associate WS-Policy documents with the service

API changes

As part of cleaning up the APIs and use of generics in the APIs, the `InterceptorProvider` API changed its methods from:

```
    List<Interceptor> getOutInterceptors();
```

to:

```
    List<Interceptor<? extends Message>> getOutInterceptors();
```

In addition, in Artix 5.6.3 for Java the runtime's CXF component was upgraded. See Migrating from Artix Java 5.6.2 to Artix Java 5.6.3 for possible API changes.

Runtime changes

Artix 5.6.3 has the following runtime changes:

- mustUnderstand headers
- SOAP/JMS specification support
- Provider<source> and Dispatch<source>
- WSDLPublish changes

While binary compatible with previous versions (type erasure makes the raw signatures the same), Artix 5.6.3 is not *source* compatible as you may need to update the types of variables used to hold the lists. Generally, this just means making the same change: add `<? extends Message>` to the declaration of the Interceptor.

mustUnderstand headers

In Artix 5.5, SOAP headers with `mustUnderstand` set to `true` were checked after the endpoint finished processing the message. The `mustUnderstand` fault was raised if the endpoint did not process the headers.

Artix 5.6.3 checks for headers with `mustUnderstand` set to true prior to dispatching the message to the endpoint's application logic. If an endpoint expects SOAP headers with `mustUnderstand` set to true, it must be configured so the runtime allows them through. To do so, configure the endpoint's `endpoint-processes-headers` property. It can take either:

- a string with the QName of a SOAP header the endpoint can process, or
- a collection of strings representing the QNames of SOAP headers the endpoint can process.

You can use the `@EndpointProperty` annotation listed in New annotations for "Java first" use cases to control this as well.

SOAP/JMS specification support

Artix now supports the W3C SOAP/JMS specification. Existing SOAP/JMS endpoints will need to be migrated. This was a result of the CXF component in the runtime dropping support for the older SOAP/JMS WSDL configuration. The ***Artix Bindings and Transports, Java Runtime*** document describes how to configure endpoints using the new SOAP/JMS configuration.

Provider<source> and Dispatch<source>

The behavior of the Provider and Dispatch interfaces has changed when they are created with a generic Source type as shown below:

```
Dispatch<Source> dispatch = s.createDispatch(portName,
                Source.class,
                Service.Mode.PAYLOAD);
```

Artix 5.5 returned a `DOMSource` object. Artix 5.6.3 returns a streaming `SAXSource` object. You can either update your applications to accept the `SAXSource` object, or set the endpoint's source-preferred-format property to dom. There is a configuration property for the Endpoint of "`source-preferred-format`" which can be set as follows:

- `dom` — DOMSource
- `sax` — SAXSource (cxf StaxSource)
- `stream` — StreamSource
- `cxf.stax` — StaxSource
- `stax` — javax.xml.transform.stax.StAXSource

WSDLPublish changes

WSDLPublish runtime is no longer dependent on QueryHandler, which has been removed in CXF. A new WSDLPublishMessageObserver, WSDLPublishInInterceptor and WSDLPublishOutInterceptor are introduced to implement the WSDL publishing functionality.

Supported Standards

Artix 5.6.3 for Java supports the following XML and Web services specifications:

SR Specifications

- JAX-WS - Java API for XML-Based WebServices 2.2 - JSR-224. See JAX-WS 2.2 support for more information.
- JAXB Java^{TM} Architecture for XML Binding 2.2 -JSR 222
- Web Services Metadata for the Java Platform - JSR-181

- JAX-RS - The Java API for RESTful Web Services - version 2.0 (JSR-339)
- SAAJ - SOAP with Attachments API for Java (SAAJ) - JSR-67

XML

- XML Namespaces 1.0
- XML Schema 1.0
- XPath 1.0
- XQuery 1.0
- XML Information Set

Messaging

- SOAP 1.1/1.2
- WS-Addressing 08-2004
- WS-Addressing 08-2005
- Message Transmission Optimization Mechanism (MTOM) for SOAP 1.1/1.2
- SOAP over JMS 1.0

Metadata

- WS-Policy 1.5, 07-2006
- WSDL 1.1 - Web Service Definition Language

Security

- WS-Security 1.1 with SAML, Kerberos, X.509 profiles
- WS-SecurityPolicy
- WS-SecureConversation
- XML Signature 02-2002
- XML Encryption 12-2002
- WS-Trust 1.4 (partial support)
- SAML 1.1, 2.0 Token Profiles

Reliable Messaging

- WS-ReliableMessaging 1.1 and 1.2

Web Services Interoperability

- WS-I Basic Profile 1.1

CORBA Compliance

Artix 5.6.3 complies with the following specifications:

- CORBA 2.6.
- GIOP 1.2 (default), 1.1, and 1.0
- IDL-to-Java Language Mapping (formal/99-07-53)

JDK Support

Artix 5.6.3 for Java supports JDK 1.6 and JDK 1.7.

(If you use JDK 1.7, update 6 or later version is required by Artix Security because of the following JDK bug: http://bugs.java.com/bugdatabase/view_bug.do?bug_id=2225327)

Micro Focus recommend moving to JDK 1.7 in the following circumstances:

- AIX users should make sure to use the latest JDK 1.7 where available.
- Certain versions of third-party Java applications with which Artix 5.6.3 for Java can work, such as WebSphere MQ 8.0, have been built with JDK 7, and as such will not work against Java 6. In situations like this Micro Focus recommend moving to use Java 7.

Platforms and Compilers

For the latest information on supported platforms and compilers, see the Artix Supported Platforms page.

Migrating from Artix Java 5.6.2 to Artix Java 5.6.3

In Artix for Java 5.6.2, the runtime was based on the following two Apache components:

- Apache CXF 2.3.5 (Fuse Services Framework)
- Apache Camel 2.6.0 (Fuse Services Framework)

In Artix for Java 5.6.3, these components have been upgraded to:

- Apache CXF 3.0.3
- Apache Camel 2.14.1

The upgrade of these components has brought with it a possibility that previous deployed applications may result in upgrade issues, because of certain API changes in these Apache components.

The following section describes the likely upgrade issues that Artix Java 5.6.2 applications may encounter in upgrading to use Artix Java 5.6.3.

Security / WSS4J

The WSS4J component was upgraded from version 1.5.12 to version 2.0.x. This will bring a number of upgrade issues:

- The Apache WSS4J project has provided a number of migration guides for migrating older WSS4J applications. In most cases the WSS4J 2.0 migration document should provide enough information, however the older WSS4J 1.5 to 1.6 migration guide may help you make sense of the property and API changes between 1.5 and 1.6, prior to upgrading to 2.0. You can see these documents here:
    - Migrating to WSS4J 1.6.x: http://ws.apache.org/wss4j/wss4j16.html
    - Migrating from WSS4J 1.6.x to 2.0.0: http://ws.apache.org/wss4j/migration.html

- If you have implemented a CallbackHandler to set and retrieve passwords for UsernameTokens, Signatures, Decryption, etc., then the namespace of the WSPasswordCallback Object has changed from "`org.apache.ws.security`" to "`org.apache.wss4j.common.ext`".
- If you have implemented a CallbackHandler to create SAML Assertions, then the namespace of the SAML bean objects has changed from "`org.apache.ws.security.saml.ext`" to "`org.apache.wss4j.common.saml`".
- A small number of configuration tags have been removed in WSS4J 2.0.0. See the section entitled *Removed Configuration Tags in WSS4J 2.0.0* in the WSS4J 2.0.0 Migration Guide for more information on this.
- The default namespace for derived keys and secure conversation is now "http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512". The older namespace can be used instead via a new configuration tag.
- The RSA v1.5 Key Transport algorithm is no longer allowed by default. This can be changed via a configuration tag.
- Turning off BSP (Basic Security Profile) Compliance (Basic Security Profile) on the outbound side no longer has the effect of disabling the addition of an InclusiveNamespaces PrefixList when signing a portion of the message. This is now controlled by a separate configuration tag in WSS4J 2.0.0.

WS-ReliableMessaging

- The WS-RM subsystem has been updated to more completely implement the 1.1 specification.
- WS-RM components are fully JMX instrumented. Various status information about the WS-RM processing such as sequences, retries, acknowledgments can be monitored over JMX.
- Closing a client proxy via `((Closable)proxy).close()` will now terminate open sequences.

Transports

- Support for the older JMS 1.0.2 APIs has been removed. Your JMS provider must support the JMS 1.1 APIs. In addition, the JMS configuration now uses the newer standardized W3C Soap over JMS specification, with the older JMS configuration now removed.
- A new WebSocket based transport has been added.
- Support for Netty based HTTP servers and clients has been added.

JAX-RS

- JAX-RS 2.0 has been completely implemented, support for JAX-RS 1.1 is provided via JAX-RS 2.0, which is backward compatible.

- JAX-RS WADL auto-generation code has been moved to a new `cxf-rt-rs-service-description` module.
- JAX-RS 2.0 Client API and CXF specific WebClient and Proxy client code is now available in a new `cxf-rt-rs-client` module.

  Important: the namespace for jaxrs:client elements has changed from "`http://cxf.apache.org/jaxrs`" to "`http://cxf.apache.org/jaxrs-client`".
- The `RequestHandler` and `ResponseHandler` filters have been removed. Instead, use JAX-RS 2.0 `ContainerRequestFilter` and `ContainerResponseFilter` and also `WriterInterceptor` and `ReaderInterceptor` when needed.
- The JAX-RS Form extension has been dropped. Use JAX-RS 2.0 Form.
- The JAX-RS `ParameterHandler` has been dropped. Use JAX-RS 2.0 `ParamConverterProvider`.
- The `javax.annotation.Resource` annotation can no longer be used to annotate JAX-RS context properties. Only `javax.ws.rs.core.Context` annotation is now supported.
- The JAX-RS "HTTP Binding" approach of annotating JAX-WS SEI's with JAX-RS annotations is deprecated, and has been removed from the runtime.

JAX-WS/SOAP

- There is a new code generator frontend that adds Artix Java-specific constructors and methods. To use this code generator pass "`-fe cxf`" into the wsdl2java tool.
- The runtime now uses JAXB 2.2 and JAXWS 2.2, matching the versions in Java 7.
- CXF has added the `AbstractFeature` subclass `WebServiceFeature` and updated the JAX-WS frontend to look for them.
- "`jaxb-validation-event-handler`"s now apply for both Reading and Writing (previously only applied to Reading). There are separate `jaxb-(reader|writer)-validation-event-handler` properties if you need it set for only one direction.
- If the WSDL location that is passed in is not valid, the runtime will now throw an exception.
- `ClientProxy.getClient(proxy)` is no longer needed for most use cases. The client proxy instances now implement the Client API directly. A direct cast to Client should work.

Removed and Changed APIs

- CXFBusImpl has been removed. The only subclass was the ExtensionManagerBus (SpringBus and Blueprint/osgi material subclassed that) so the functionality was pushed up into ExtensionManagerBus. Some of the "common" methods were put directly on the Bus interface to make using the Bus cleaner (no casts to the impl).
- The unused "`run()`" method on Bus was removed.

- Merge BaseDataReader/DataReader and the same for the writer, getting rid of the "Base" versions that are unreferenced.
- The two unused parameters on `Destination.getBackChannel` were removed. They were unused and normally passed in as null.
- Removed QueryHandlers. These were originally used for the `?wsdl` processing (and are still used for `?js`). However, that is better done directly on the interceptor chains as interceptors to allow user supplied interceptors to also handle them.
- Removed all the `/META-INF/cxf/cxf-extension-XYZ.xml` files.
- Updated ConduitInitiator and DestinationFactory to pass Bus as parameter to the various methods.
- Removed support for the old bus-extensions.xml file (in favor of the current and much faster bus-extensions.txt)
- All XML parsing and writing has been moved to StaxUtils and DOM based utilities to DOMUtils. The XMLUtils class that used SAX based parsing and Transformer based writing has been eliminated. This simplifies the code as well as increasing security as the StAX based IO provides better limits and has more control.
- `AddressingProperties` has been turned from an interface to a concrete class that can be created directly with "new". `AddressingPropertiesImpl` has been removed.
- Many of our interfaces/classes that held onto constants were either removed or moved. In particular `XmlSchemaConstants` was removed (use the Constants from the XmlSchema library directly), `WSDLConstants` was moved from API to rt-wsdl, `SOAPConstants` was removed (most are available in `WSDLConstants`). The goal was to reduce some memory usage and help startup time and reduce a lot of duplication.
- `AlternativeSelector` and the `PolicyEngine` and other `PolicyRelated` classes have been updated to pass the current Message in (when appropriate) to allow using message contextual information to select the alternative. HOWEVER, keep in mind that the selected alternative is likely cached and thus if contextual information changes, the alternative may not be recalculated.
- FailoverTargetSelector will not activate the fail-over in cases when HTTP client errors are returned, only HTTP 404 and 503 statuses will be recognized. Set `FailoverTargetSelector supportNotAvailableErrorsOnly` property to `false` if the support for all HTTP errors is required.
- ServletController will not override the endpoint addresses by default as it has side-effects when a given endpoint is accessed via multiple paths. Set CXFServlet "`disable-address-updates`" parameter to '`false`' if required.
- The deprecated `org.apache.cxf.frontend.MethodDispatcher` has been removed.

- The deprecated `JAXBToStringBuilder` and `JAXBToStringStyle` classes that were in `cxf-rt-databinding-jaxb` have been removed. The functionality is now provided by `cxf-xjc-runtime`.
- The deprecated URIMappingInterceptor has been removed. This has not been on the default chain for some time because of security related issues.
- SchemaValidation annotation has had its deprecated "`enabled`" property removed. Please use its "`type`" property to control the validation.
- The "`Spring`" type was removed from the FactoryType annotation. Instead, use `factoryClass=SpringBeanFactory.class`.
- GZIP related interceptors/features have been moved out of the http module so they are usable with other transports such as JMS. As such, their package has changed from `org.apache.cxf.transport.http.gzip` to `org.apache.cxf.transport.common.gzip`.
- The WS-Addressing related VersionTransformer and MAPCodec classes have been changed to not encode the WS-Addressing headers to DOM elements and instead just use the Header list on the SoapMessage directly. This did change the parameters on the encode methods to take the JAXBContext instead of the Marshaller. Any custom VersionTransformers will need to be updated.
- Three JAX-RS classes have been removed from the `org.apache.cxf.jaxrs.client` package: `ResponseReader`, `ServerWebApplicationException`, and `ClientWebApplicationException`. The first class in the list is not needed with JAX-RS 2.0 Response class, the latter two are replaced by `javax.ws.rs.WebApplicationException` and `javax.ws.rs.client.ClientException`.
- All APIs that take or return "generic" classes have been updated to properly define the generic part. For example, methods like "`Class getServiceClass()`" have been updated to be "`Class<?> getServiceClass()`".
- To resolve some of the "split-package" issues between jars, a very few classes have had their packages changed.
    - `org.apache.cxf.jaxb.JAXBUtils` has c hanged to `org.apache.cxf.common.jaxb.JAXBUtils` (and a few other classes in that jaxb package)
    - Many of the internal "Impl" classes and "Managers" (such as `BindingFactoryManagerImpl` and `CXFBusLifeCycleManager`) have moved into `org.apache.cxf.bus.managers`. You should always rely on the interfaces they implement anyway.
- The `selectedConduit` field of `AbstractConduitSelector` has been removed as a `ConduitSelector` may be used to select multiple conduits depending on scenarios and the selectedConduit field may not accurately reflect the conduit that had been selected depending on the state of the threads, clients, and so on.

- The classes in `org.apache.cxf.tools.common.extensions.soap` have been moved to `org.apache.cxf.binding.soap.wsdl.extensions` to remove a runtime dependency on the tooling modules.
- `org.apache.cxf.tools.common.DataTypeAdapter` has been deprecated and moved to `org.apache.cxf.xjc.runtime.DataTypeAdapter` in a new runtime jar that is part of the `cxf-xjc` package. This allows using the runtime without bringing in all the CXF tooling dependencies.

  The DataTypeAdapter has no other runtime dependencies.
- The XJC "`ToString`" plugin also had its runtime dependencies moved out of `org.apache.cxf.tools` and into the `cxf-xjc-runtime` jar. If you use the newer version of the ToString plugin, you will need to add the `cxf-xjc-runtime` dependency to your application. However, the `cxf-xjc-runtime` jar does not depend on other jars or classes and can thus be easily used in other applications without as many dependencies pulled in.
- XMLSchema has been upgraded from 1.4.x to 2.0.x. As such, any use of XmlSchema classes may have changed. In particular, XmlSchema 2.0 uses Java 5 collections which changes how it is used. Also, many static utility methods that existed in `org.apache.cxf.common.xmlschema.XmlSchemaUtils` have now been merged directly into the XmlSchema APIs and are no longer needed or available.
- `org.apache.cxf.ws.addressing.VersionTransformer.convertToInternal` has been replaced by `org.apache.cxf.jaxws.spi.ProviderImpl.convertToInternal`.
- `org.apache.cxf.wsdl.EndpointReferenceUtils` has moved to `org.apache.cxf.ws.addressing.EndpointReferenceUtils`.
- `org.apache.cxf.binding.corba.CorbaBindingFactory.getDestination` now takes the signature `getDestination(EndpointInfo, Bus)` instead of `getDestination(Endpoint)`.
- `org.apache.cxf.wsdl11.WSDLServiceFactory constructor` now takes `WSDLServiceFactory(Bus, String, QName)` instead of `WSDLServiceFactory(Bus, URL, QName)`.
- `org.apache.cxf.binding.corba.CorbaDestination` method `getBackChannel` now takes the signature `getBackChannel(Message)` instead of `getBackChannel(Message, Message, EndpointReferenceType)`.
- When using the `SpringBusFactory` the former

  ```
  SpringBusFactory bf = (SpringBusFactory)
  SpringBusFactory.newInstance()
  ```

  must be replaced by

  ```
  SpringBusFactory bf = new SpringBusFactory();
  ```

New Features in the runtime

- New UDP Transport.
- New Local/In-jvm Transport.
- New optional HTTP transport based on Apache HTTP Components HttpAsyncClient.
- Support for the SOAP over UDP.
- Ability to only apply schema validation to incoming or outgoing messages by setting the "schema-`validation`-enabled" property to "IN", "OUT", "BOTH", or "NONE". @`SchemaValidationEnabled` annotation updated to have a t`ype=IN|OUT|BOTH|NONE` parameter.
- Support for WS-Discovery:
    - Services can send Hello or Bye when started or stopped as well as respond to Probe requests
    - API for sending probes and resolving to EndpointReferences
- New Security Token Service.
- JIBX databinding.
- Faster startup and reduced Spring configuration. The Spring support has been redone based on the `ExtensionManagerBus`. This results in much faster startup. It also means that all of the imports of `META-INF/cxf/cxf-<extension>*.xml` are no longer needed and are deprecated.
- WS-SecurityPolicy support for SAML tokens.
- Transformation feature provides for a fast and effective way to transform inbound and/or outbound XML messages.

Module and jar changes

- Combined API and core into just a cxf-core module. All WSDL-related functionality has been moved to a new `cxf-wsdl` bundle to remove the wsdl4j requirement for JAX-RS applications.
- The direct dependency on a javax.mail implementation has been removed. If your application uses MTOM or Soap with Attachments or similar that requires some of the DataContentHandlers that are part of the mail implementations, you may need to re-add this to your classpath.
- `DynamicClientFactory` was moved from the JAXB databinding to the Simple frontend. However, users are strongly encouraged to use the `JaxWsDynamicClientFactory` subclass.
- The single cxf-<*version*> jar is no longer provided, with the runtime split into more modular jars, which allow users to choose the parts of the runtime that they actually need. The cxf-manifest jar is still provided, that will pull in all the runtime jars, this approach is recommended if in doubt.
- The `org.apache.cxf.tools.*` classes that were in cxf-api have been moved into `cxf-tools-common` or `cxf-tools-validator`.

- The `org.apache.cxf.ws.policy` classes that were in cxf-api have been moved into `cxf-rt-ws-policy`.
- `cxf-common-utilities` is no longer available. All the classes in there were moved into cxf-core to represent a more complete API.
- Various classes in `cxf-rt-core` and `cxf-rt-ws-addr` have been moved up to cxf-core to resolve split-package issues.
- JAX-RS Search extension code has been moved to a new `cxf-rt-rs-extension-search` module.
- All the JBI-related modules have been removed.
- The Jetty-based HTTP transport has been updated to Jetty 8.1.

Runtime changes

- The syntax of WS-SecurityPolicy policies is enforced more strictly.
- In JMS Transport, when using TextMessage, the runtime now leaves the contents as a String and uses `java.io.Reader` and `java.io.Writer` to boost performance.
- The Logging interceptors now log using service specific categories and loggers instead of just `LoggingInInterceptor` and `LoggingOutInterceptor`. The name of the logger used is *`org.apache.cxf.services.ServiceName.PortName.PortTypeName`*. This allows you to configure specific per-service filters and formatters in your logging configuration.
- The ExtensionManagerBus (mostly used when Spring is not available) has been updated to completely support all the features including the WS-SecurityPolicy, WS-RM, and other features. Previous WSDL documents that contained policy fragments may now behave differently as the policies will be enforced.
- The default CA certificates that ship with the JDK are now not loaded by default by the WS-Security Crypto implementation, which is used for encryption and decryption and for signature creation and verification.
- The way that UsernameTokens are processed by WSS4J has been changed. The callbackhandler identifier for plaintext passwords is now `WSPasswordCallback.USERNAME_TOKEN`, the same as for the digest case. The CallbackHandler implementation only sets the password on the callback, and never does any validation of the password.
- The syntax of WS-SecurityPolicy policies is enforced more strictly.

Online migration guides

Below are some links to migration information for the CXF major releases, between 2.3.x and 3.0.x. Migration issues not found in the previous sections may be resolved by consulting these links:

- Migration to CXF 2.4
- Migration to CXF 2.5

- Migration to CXF 2.6
- Migration to CXF 2.7
- Migration to CXF 3.0

Deprecated Features

The following features are deprecated in Artix 5.6.3 for Java and will be removed in a future release:

- In Java routing, the MESSAGE data format is deprecated and replaced with the RAW format. See the ***Artix Java Router, Deployment Guide*** for details.

Dropped Features

The following features are no longer supported in Artix 5.6.3 for Java:

- The Artix Locator component no longer ships with Artix 5.6.3 for Java. However, Locator support for Java clients and servers is still available as part of this release providing high-availability and fail-over support. To use these capabilities the Artix Locator from an Artix C++ release must be used.
- Aurea Actional® interceptors for Artix ESB are not shipped as part of Artix ESB 5.6.3. These interceptors are available as part of the Actional product line.
- OSGi: Artix components can no longer be automatically deployed to an OSGi compliant container.
- FTP Transport: For FTP transport capabilities, users should use the Camel router or the CXF Camel transport.
- DB Service
- Eclipse Designer: Users must use the Artix Command Line tools for code generation, instead of using the Artix Designer. For more information, see the Artix ESB Command Reference.
- Security Service: The Artix Security Service responsible for authentication and authorization support no longer ships with Artix. Standard HTTPS security as well as message encryption using the WSS4j library is still available in Artix.

Documentation Updates

The Artix 5.6.3 user documentation is available from https://supportline.microfocus.com/productdoc.aspx

All books have been rebranded as Micro Focus products, and updated.

In particular:

- ***Developing Artix Applications with JAX-WS*** has been retitled as ***Developing Artix Applications with JAX-WS and JAX-RS***. It now covers the use of Artix to develop RESTful applications using the JAX-RS APIs.

- In the ***Artix Java Router, Deployment Guide*** the coverage of CXF options has been updated.

Resolved Issues

The resolved issues that customers have reported are listed in this section.

Artix for Java 5.6.3 contains fixes to the runtime components Apache CXF version 3.0.3 and Apache Camel 2.14.1, which were made available (or "pushed upstream") to the Apache community for each of the respective projects.

Fix	Description
CXF fixes	
CXF-6187	Adding jndiURL to Client and Server-side, for *java_first_jms* demo'
CXF-6235	wsdl2java behaves differently from cxf-codegen-plugin.
CXF-6178	Make everything about netty completely optional.
CXF-6147	Part of demo wsdl_first_soap12 does not work.
CXF-6170	Fix a jaxrs spring_security demo.
CXF-6145	WS-RM demo server throws exception.
Camel fixes	
CAMEL-8042	CxfClientCallBack handleException does not honor exceptions.

Other Resources

The following additional resources are available:

- For the latest information on supported platforms and compilers, see the Artix Supported Platforms page.
- The most up-to-date versions of Artix technical documentation are available at:

  https://supportline.microfocus.com/productdoc.aspx
- The Artix Community enables developers, support specialists, and customers to exchange information. This is available at: http://community.microfocus.com/microfocus/corba/artix/
- Contact Micro Focus technical support at:

  http://www.microfocus.com