

PROGRESS<sup>®</sup>  
ARTIX<sup>™</sup>

Management Guide, C++  
Runtime

Version 5.6, December 2011

**© 2011 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.**

These materials and all Progress<sup>®</sup> software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Actional, Apama, Artix, Business Empowerment, DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect Technologies, DataDirect XML Converters, DataDirect XQuery, DataXtend, Dynamic Routing Architecture, EdgeXtend, Empowerment Center, Fathom, Fuse Mediation Router, Fuse Message Broker, Fuse Services Framework, IntelliStream, IONA, Making Software Work Together, Mindreef, ObjectStore, OpenEdge, Orbix, PeerDirect, POSSENET, Powered by Progress, PowerTier, Progress, Progress DataXtend, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress Software Developers Network, Progress Sonic, ProVision, PS Select, Savvion, SequeLink, Shadow, SOAPscope, SOAPStation, Sonic, Sonic ESB, SonicMQ, Sonic Orchestration Server, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software, Our Technology—Experience the Connection are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Business Making Progress, Cache-Forward, CloudEdge, DataDirect Spy, DataDirect SupportLink, Fuse, FuseSource, Future Proof, GVAC, High Performance Integration, ObjectStore Inspector, ObjectStore Performance Expert, OpenAccess, Orbacus, Pantero, POSSE, ProDataSet, Progress Arcade, Progress CloudEdge, Progress Control Tower, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress RPM, Progress Software Business Making Progress, PSE Pro, SectorAlliance, SeeThinkAct, Shadow z/Services, Shadow z/Direct, Shadow z/Events, Shadow z/Presentation, Shadow Studio, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Sonic Business Integration Suite, Sonic Process Manager, Sonic Collaboration Server, Sonic Continuous Availability Architecture, Sonic Database Service, Sonic Workbench, Sonic XML Server, The Brains Behind BAM, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

### **Third Party Acknowledgments:**

Progress Artix ESB for C++ v5.6 incorporates Xalan v2.3.1 technologies from the Apache Software Foundation (<http://www.apache.org>). Such Apache technologies are subject to the following terms and conditions: The Apache Software License, Version 1.1. Copyright (C) 1999-2002 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Ant", "Xerces," "Xalan," "Log 4J," and "Apache Software Foundation" must not be used to: endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org). 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>. Xalan was originally based on software copyright (c) 1999, Lotus Development Corporation., <http://www.lotus.com>. Xerces was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com>.

Progress Artix ESB for C++ v5.6 incorporates Xerces C++ v2.4 technology from the Apache Software Foundation (<http://www.apache.org>). Such Apache technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 incorporates Apache Xerces v2.5.0 technology from the Apache Software Foundation (<http://www.apache.org>). Such Apache technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 1999-2002 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Xerces" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, International Business Machines, Inc., <http://www.ibm.com>. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Artix ESB for C++ v5.6 incorporates Xerces C++ v1.7 technology from the Apache Software Foundation (<http://www.apache.org>). Such Apache technology is subject to the following terms and conditions: The Apache Software License, Version 1.1. - Copyright (c) 1999-2004 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Xalan" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING

ING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation and was originally based on software copyright (c) 1999, Lotus Development Corporation., <http://www.lotus.com>. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Artix ESB for C++ v5.6 incorporates Apache Velocity v1.3 technology from the Apache Software Foundation (<http://www.apache.org>). Such Apache technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 2000-2003 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear.
4. The names "The Jakarta Project", "Velocity", and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).
5. Products derived from this software may not be called "Apache", "Velocity" nor may "Apache" appear in their names without prior written permission of the Apache Group.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 incorporates Log4J v1.2.6 technology from the Apache Software Foundation (<http://www.apache.org>). Such Apache technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (C) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in

source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "log4j" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [apache@apache.org](mailto:apache@apache.org).

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

(a) Progress Artix ESB for C++ v5.6 incorporates JDOM Beta 9 technology from JDOM. Such technology is subject to the following terms and conditions: Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [request\\_AT\\_jdom\\_DOT\\_org](mailto:request_AT_jdom_DOT_org). 4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management [request\\_AT\\_jdom\\_DOT\\_org](mailto:request_AT_jdom_DOT_org). In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This

product includes software developed by the JDOM Project (<http://www.jdom.org/>).<sup>1</sup> Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos>. THIS SOFTWARE IS PROVIDED AS IS AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter\_AT\_jdom\_DOT\_org> and Brett McLaughlin <brett\_AT\_jdom\_DOT\_org>. For more information on the JDOM Project, please see <<http://www.jdom.org/>>

Progress Artix ESB for C++ v5.6 incorporates IBM-ICU v2.6 and IBM-ICU v2.6.1 technologies from IBM. Such technologies are subject to the following terms and conditions: Copyright (c) 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder. All trademarks and registered trademarks mentioned herein are the property of their respective owners.

Progress Artix ESB for C++ v5.6 incorporates John Wilson MinML v1.7 technology from John Wilson. Such technology is subject to the following terms and conditions: Copyright (c) 1999, John Wilson ([tug@wilson.co.uk](mailto:tug@wilson.co.uk)). All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright

notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by John Wilson. The name of John Wilson may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY JOHN WILSON ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JOHN WILSON BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 incorporates SourceForge - NET-SNMP v5.0.7 technology from SourceForge and Networks Associates Technology, Inc. Such technology is subject to the following terms and conditions: Various copyrights apply to this package, listed in various separate parts below. Please make sure that you read all the parts. Up until 2001, the project was based at UC Davis, and the first part covers all code written during this time. From 2001 onwards, the project has been based at SourceForge, and Networks Associates Technology, Inc hold the copyright on behalf of the wider Net-SNMP community, covering all derivative work done since then. An additional copyright section has been added as Part 3 below also under a BSD license for the work contributed by Cambridge Broadband Ltd. to the project since 2001. An additional copyright section has been added as Part 4 below also under a BSD license for the work contributed by Sun Microsystems, Inc. to the project since 2003. Code has been contributed to this project by many people over the years it has been in development, and a full list of contributors can be found in the README file under the THANKS section. ---- Part 1: CMU/UCD copyright notice: (BSD like) ---- Copyright 1989, 1991, 1992 by Carnegie Mellon University. Derivative Work - 1996, 1998-2000. Copyright 1996, 1998-2000 The Regents of the University of California. All Rights Reserved. Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU and The Regents of the University of California not be used in advertising or publicity pertaining to distribution of the software without specific written permission. CMU AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL CMU OR THE REGENTS OF THE UNIVERSITY OF CALIFORNIA BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM THE LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR

IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. ---- Part 2: Networks Associates Technology, Inc copyright notice (BSD) ----- Copyright (c) 2001-2003, Networks Associates Technology, Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: \*Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.\* Neither the name of the Networks Associates Technology, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ---- Part 3: Cambridge Broadband Ltd. copyright notice (BSD) ----- Portions of this code are copyright (c) 2001-2003, Cambridge Broadband Ltd. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:\*Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.\* The name of Cambridge Broadband Ltd. may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE

OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ---- Part 4: Sun Microsystems, Inc. copyright notice (BSD) -----Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved. Use is subject to license terms below. This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the fol-

lowing conditions are met:\* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.\* Neither the name of the Sun Microsystems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ---- Part 5: Sparta, Inc copyright notice (BSD) -----Copyright (c) 2003-2005, Sparta, Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:\* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.\* Neither the name of Sparta, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ---- Part 6: Cisco/BUPTNIC copyright notice (BSD) ----- Copyright (c) 2004, Cisco, Inc and Information Network Center of Beijing University of Posts and Telecommunications. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:\* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. \* Neither the name of Cisco, Inc, Beijing University of Posts and Telecommunications, nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS

PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. ---- Part 7: Fabasoft R&D Software GmbH & Co KG copyright notice (BSD) ----- Copyright (c) Fabasoft R&D Software GmbH & Co KG, 2003 oss@fabasoft.com Author: Bernhard Penz. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:\* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. \* The name of Fabasoft R&D Software GmbH & Co KG or any of its subsidiaries, brand or product names may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 incorporates OpenSSL/SSLey v0.9.8i technology from OpenSSL.org. Such Technology is subject to the following terms and conditions: LICENSE ISSUES =====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLey license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License -----

/\*

=====  
=====

Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [openssl-core@openssl.org](mailto:openssl-core@openssl.org).
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====  
=====

This product includes cryptographic software written by Eric Young ([ey@cryptsoft.com](mailto:ey@cryptsoft.com)).  
This product includes software written by Tim Hudson ([tjh@cryptsoft.com](mailto:tjh@cryptsoft.com)).

Original SSLey License -----

Copyright (C) 1995-1998 Eric Young ([ey@cryptsoft.com](mailto:ey@cryptsoft.com)) All rights reserved.

This package is an SSL implementation written by Eric Young ([ey@cryptsoft.com](mailto:ey@cryptsoft.com)). The implementation was written so as to conform with Netscapes SSL. This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson ([tjh@cryptsoft.com](mailto:tjh@cryptsoft.com)). Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be

given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Progress Artix ESB for C++ v5.6 incorporates Bouncycastle v1.3.3 cryptographic technology from the Legion Of The Bouncy Castle (<http://www.bouncycastle.org>). Such Bouncycastle 1.3.3 cryptographic technology is subject to the following terms and conditions: Copyright (c) 2000 - 2006 The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>). Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING

FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Progress Artix ESB for C++ v5.6 incorporates PCRE 7.8 from PCRE for the purpose of providing a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5. Such technology is subject to the following terms and conditions: PCRE LICENCE. PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language. Release 7 of PCRE is distributed under the terms of the "BSD" licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself. The basic library functions are written in C and are freestanding. Also included in the distribution is a set of C++ wrapper functions. THE BASIC LIBRARY FUNCTIONS. Written by: Philip Hazel. Email local part: ph10. Email domain: cam.ac.uk. University of Cambridge Computing Service, Cambridge, England. Copyright (c) 1997-2008 University of Cambridge All rights reserved. THE C++ WRAPPER FUNCTIONS. Contributed by: Google Inc. Copyright (c) 2007-2008, Google Inc. All rights reserved. THE "BSD" LICENCE. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. \* Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 incorporates mcpp v2.6.4 from Kiyoshi Matsui. Such technology is subject to the following terms and conditions: Copyright (c) 1998, 2002-2007 Kiyoshi Matsui kmatsui@t3.rim.or.jp All rights reserved. This software including the files in this directory is provided under the following license. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Artix ESB for C++ v5.6 contains IBM Licensed Materials Copyright IBM Corporation 2010 (IBM 32-bit Runtime Environment for AIX, Java Technology Edition v 1.6.0 SR9 FP2).

Updated: December 5, 2011

# Contents

<b>List of Figures</b>	<b>23</b>
<b>Preface</b>	<b>25</b>
<b>Part I Introduction</b>	
<b>Chapter 1 Artix C++ Runtime Management</b>	<b>31</b>
<b>Introduction to Artix C++ Management</b>	<b>32</b>
<b>Artix C++ Management Integrations</b>	<b>37</b>
<b>Part II Java Management Extensions</b>	
<b>Chapter 2 Monitoring and Managing with JMX</b>	<b>43</b>
<b>Introduction</b>	<b>44</b>
<b>Managed Bus Components</b>	<b>49</b>
<b>Managed Service Components</b>	<b>55</b>
<b>Managed Port Components</b>	<b>63</b>
<b>Chapter 3 Configuring JMX in Artix C++</b>	<b>67</b>
<b>Artix JMX Configuration</b>	<b>68</b>
<b>Chapter 4 Managing Artix Services with JMX Consoles</b>	<b>71</b>
<b>Managing Artix Services with JConsole</b>	<b>72</b>
<b>Managing Artix Services with the JMX HTTP adaptor</b>	<b>76</b>
<b>Chapter 5 Managing WS-RM Persistence with JMX</b>	<b>79</b>
<b>WS-RM Persistence Management</b>	<b>80</b>
<b>Viewing Messages in the WS-RM Persistence Database</b>	<b>82</b>

## Part III Progress Actional

<b>Chapter 6 Artix–Actional Integration</b>	<b>91</b>
Artix–Actional Interaction Architecture	92
<b>Chapter 7 Configuring Artix–Actional Integration</b>	<b>99</b>
Prerequisites	100
Configuring Actional for Artix Integration	101
Configuring Artix for Actional Integration	104
Viewing Artix Endpoints in Actional	106

## Part IV AmberPoint

<b>Chapter 8 Integrating with AmberPoint</b>	<b>111</b>
AmberPoint Proxy Agent	112
Artix AmberPoint Agent	115
<b>Chapter 9 Configuring the Artix AmberPoint Agent</b>	<b>123</b>
Installing AmberPoint	124
Configuring AmberPoint for Artix Integration	125
Configuring Artix C++ Services for AmberPoint Integration	128

## Part V BMC Patrol

<b>Chapter 11 Integrating with BMC Patrol™</b>	<b>135</b>
Introduction	136
The Artix BMC Patrol Integration	140
<b>Chapter 12 Configuring Artix for BMC Patrol</b>	<b>143</b>
Setting up your Artix Environment	144
<b>Chapter 13 Using the Artix BMC Patrol Integration</b>	<b>147</b>

<b>Setting up your BMC Patrol Environment</b>	<b>148</b>
<b>Using the Artix Knowledge Module</b>	<b>150</b>
<b>Chapter 14 Extending to a BMC Production Environment</b>	<b>157</b>
<b>Configuring an Artix Production Environment</b>	<b>158</b>
<b>Index</b>	<b>161</b>

## CONTENTS

# List of Figures

Figure 1: Artix C++ Runtime Management Architecture	33
Figure 2: Artix Service Endpoint in JConsole	38
Figure 3: Actional Server Administration Console	39
Figure 4: Artix JMX Architecture	45
Figure 5: Managed Service in JConsole	73
Figure 6: Managed Port in JConsole	74
Figure 7: Managed Locator in JConsole	75
Figure 8: HTTP Adaptor Main View	77
Figure 9: HTTP Adaptor Bus View	78
Figure 10: Connecting to a JMX Agent	83
Figure 11: WS-RM Persistence Enabled Endpoint	84
Figure 12: WS-RM Sequence Attributes	86
Figure 13: Messages in the WS-RM Persistence database	87
Figure 14: Artix–Actional Integration Architecture	93
Figure 15: Actional Server Administration Console	96
Figure 16: Actional Server Network Overview	107
Figure 17: Actional Server Path Explorer	108
Figure 18: AmberPoint Proxy Agent Integration	112
Figure 19: AmberPoint Proxy Agent Service Network	113
Figure 20: Artix AmberPoint Agent Integration	115
Figure 21: Artix AmberPoint Agent Embedded in Service Endpoint	117
Figure 22: Artix AmberPoint Agent Service Network	119
Figure 23: Overview of the Artix BMC Patrol Integration	138
Figure 24: IONA Server Running in BMC Patrol	141
Figure 25: BMC Patrol Displaying Alarms	142
Figure 26: Graphing for IONAAvgResponseTime	153

Figure 27: Alarms for IONAAvgResponseTime

154

# Preface

## **What is covered in this book**

This guide describes the enterprise management features for Artix applications that use the C++ runtime. It explains how to integrate and manage Artix applications with the following:

- Java Management Extensions (JMX)
- Progress Actional
- AmberPoint
- BMC Patrol

This guide applies to Artix applications written using in C++ only.

For information on Artix applications written in JAX-WS (Java XML-Based APIs for Web Services) or JavaScript, see the [Artix Management Guide, Java Runtime](#).

## **Who should read this book**

This guide is aimed at system administrators managing distributed enterprise environments, and developers writing distributed enterprise applications. Administrators do not require detailed knowledge of the technology that is used to create distributed enterprise applications.ns.

This book assumes that you already have a good working knowledge of at least one of the management technologies mentioned in [“What is covered in this book”](#).

## Organization of this book

This book contains the following parts:

### Part I

- [Chapter 1](#) introduces the Artix C++ management architecture and features.

### Part II

- [Chapter 2](#) introduces the JMX features supported by the Artix C++ runtime, and describes the Artix components that can be managed using JMX.
- [Chapter 3](#) explains how to configure an Artix C++ runtime for JMX.
- [Chapter 4](#) explains how to manage and monitor Artix services using JMX consoles.
- [Chapter 5](#) shows how to manage Web Services Reliable Messaging persistence in Artix using a JMX console

### Part III

- [Chapter 6](#) describes the architecture of the Artix C++ runtime integration with Actional.
- [Chapter 7](#) explains how to configure integration between Artix and Actional SOA management products.
- [Chapter 8](#) shows examples of monitoring Artix services using Actional.

### Part IV

- [Chapter 8](#) describes the architecture of the Artix C++ runtime integration with AmberPoint.
- [Chapter 9](#) explains how to configure integration with the Artix AmberPoint Agent, and shows examples from the Artix AmberPoint integration demo.

**Part V**

- [Chapter 11](#) introduces Enterprise Management Systems, and the Artix integration with BMC Patrol.
- [Chapter 12](#) describes how to configure your Artix environment for integration with BMC Patrol.
- [Chapter 13](#) describes how to configure your BMC Patrol environment for integration with Artix.
- [Chapter 14](#) describes how to extend an Artix BMC Patrol integration from a test environment to a production environment

**The Artix Documentation Library**

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

## PREFACE

# Part I

## Introduction

---

### **In this part**

This part contains the following chapters:

<a href="#">Artix C++ Runtime Management</a>
--

page 31
---------



# Artix C++ Runtime Management

*Artix provides support for integration with a range of management systems. This chapter introduces the management architecture for the Artix C++ runtime and the supported integrations.*

---

**In this chapter**

This chapter includes the following section:

<a href="#">Introduction to Artix C++ Management</a>	<a href="#">page 32</a>
<a href="#">Artix C++ Management Integrations</a>	<a href="#">page 37</a>

---

# Introduction to Artix C++ Management

---

## Overview

This section introduces the Artix ESB C++ runtime management architecture and explains its various components. This applies to Artix applications written in C++.

---

## Management architecture

The Artix ESB C++ management architecture provides:

- Integration with third-party enterprise management and SOA management systems
- Instrumentation used to monitor system status and potential problems
- Flexible runtime configuration
- Tools for developers without access to management systems.

[Figure 1](#) shows a basic overview of the Artix C++ management architecture. The Artix C++ runtime uses Artix plug-ins and interceptors to send management instrumentation data to third-party management systems.

In addition, the Artix instrumentation data can also be monitored using JMX-compliant consoles.

---

## Integration with third-party management systems

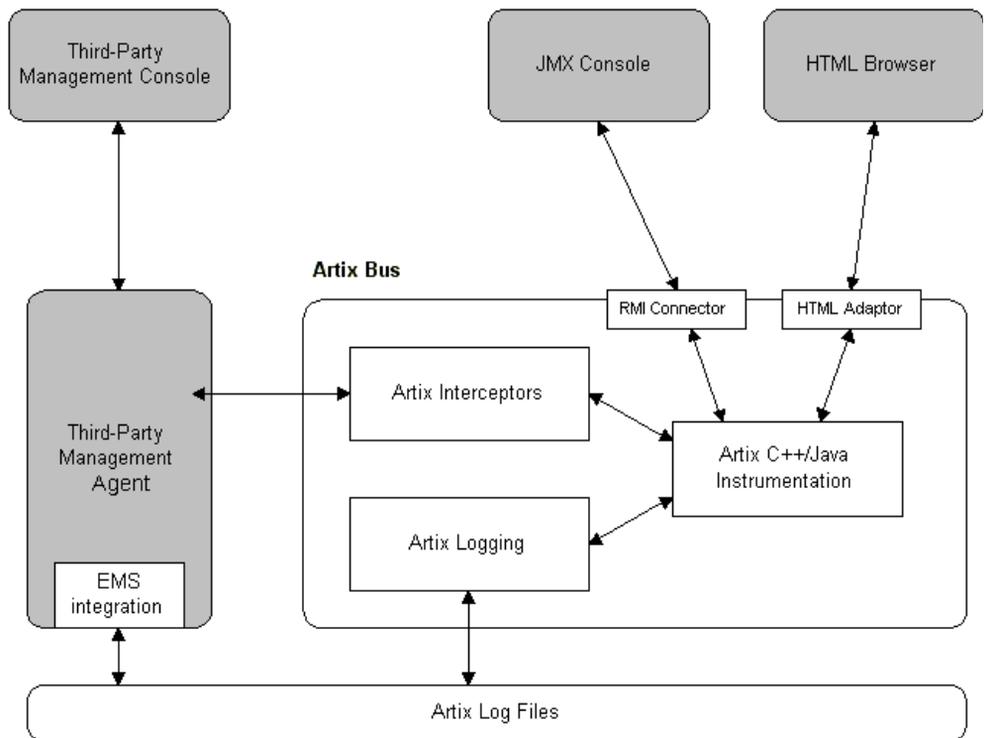
Integrations with third-party enterprise management and SOA management systems are critical to large corporations. Artix provides integration with the Actional and AmberPoint SOA management systems, and the BMC Patrol Enterprise Management System (EMS).

These management systems give a top-to-bottom view of enterprise infrastructure. For example, this means that instead of getting 100 different messages when services are not responding, you get a single message saying your services on these hosts are not working because the following network segment is dead.

If you integrate with an enterprise management or SOA management system, your product can also be hooked into higher-level monitoring tools such as Business Activity Monitoring (BAM), Service Level Agreement monitoring, and impact analysis tools. For example, when something goes wrong, the relevant administrators are automatically notified, trouble tickets are created, and service level impact is analyzed.

For more details on integration with third-party management systems, see [“Artix C++ Management Integrations”](#) on page 37.

**Figure 1:** *Artix C++ Runtime Management Architecture*



## Instrumentation

---

Management *instrumentation* refers to application code used to monitor specific components in a system (for example, code that outputs logging or performance data to a management console). Instrumentation is used to reflect the state of a system and view potential problems with the normal operation of the system, while imposing minimal overhead. If you are using instrumentation to view problems, it is important that the act of observing the system causes minimal disturbance.

The main types of instrumentation supported by Artix include:

- Object-based instrumentation (for example, JMX)
- Logging

### Object-based instrumentation

Artix supports object-based instrumentation using Java Management Extensions (JMX). The main purpose of this object-based instrumentation is to enable monitoring and management of Artix applications by JMX-aware third-party management consoles such as JConsole (see [Figure 1](#)).

Artix has been instrumented to allow Java runtime components to be exposed as JMX Managed Beans (MBeans). This enables an Artix Java runtime to be monitored and managed either in process or remotely using the JMX Remote API. Managed components are exposed using an `Object` interface with attributes and methods.

Artix Java runtime components can be exposed as JMX MBeans out-of-the-box (for example, Artix C++ service endpoints and Artix C++ bus). In addition, the Artix C++ runtime supports the registration of custom MBeans. Java developers can create their own MBeans and register them either with their JMX MBean server of choice, or with a default MBean server created by Artix

For more details on JMX object-based instrumentation, see [Part II “Java Management Extensions”](#).

## Logging

Logging in the Artix C++ runtime is controlled by the `event_log:filters` configuration variable, and by the log stream plug-ins. For example, the `local_log_stream` sends logging to a text file, and the `xmlfile_log_stream` directs logging to an XML file.

The `event_log:filters` configuration variable is used to specify logging severity levels—for information, warning, error, and fatal error messages. You can also use the `event_log:filters` variable to set fine-grained logging for specific Artix subsystems. For example, you can set logging for the Artix core, specific transports, bindings, or services. You can set logging for Artix services, such as the locator, and for services that you have implemented.

For more details on Artix C++ runtime logging, see [Configuring and Deploying Artix Solutions, C++ Runtime](#).

---

## Flexible configuration

The Artix C++ runtime is based on the highly flexible and scalable Adaptive Runtime (ART). This is a plug-in based architecture in which runtime behavior is configured using common and application-specific settings that are applied during application start up. This means that the same application code can be run, and can exhibit different capabilities, in different configuration environments.

You can change default behavior, enable specific functionality, or fine-tune behavior using a number of different configuration mechanisms. These include configuration file, command line, or programmatic configuration.

Artix configuration files are typically organized into a hierarchy of scopes, whose fully-qualified names map directly to Artix bus names. By organizing configuration variables into various scopes, you can provide different settings for individual services, or common settings for groups of services.

For more details on ART-based configuration, see [Configuring and Deploying Artix Solutions, C++ Runtime](#).

---

## Developer-based tools

Large corporations use third-party enterprise management and SOA management systems to monitor Artix applications in production environments. However, the following users need to use more lightweight management tools:

- Application developers who need to test the effects of their changes in a running test environment.
- Application developers who do not have access to an enterprise management or SOA management system.
- Support engineers who need to diagnose or correct problems raised by customers or management systems.

For facilitate such users, Artix provides out-of-the-box integration with JConsole. For more details, see [“JMX” on page 37](#).

---

# Artix C++ Management Integrations

---

## Overview

Artix has been designed to integrate with a range of third-party management systems. These include enterprise management systems, SOA management systems, and developer-focused tools. This section introduces Artix integrations with the following systems:

- “JMX”
  - “Progress Actional”
  - “AmberPoint”
  - “BMC Patrol”
- 

## JMX

The JMX instrumentation provided in Artix enables Artix service endpoints and the Artix bus to be monitored by any JMX-compliant management console (for example, JConsole or MC4J).

You can use JMX consoles to monitor and manage key Artix Java runtime components both locally and remotely. For example, using any JMX-compliant client, you can perform tasks such as:

- View service status
- View a service endpoint’s address
- Stop or start a service
- Shutdown an Artix Java bus

Artix provides out-of-the-box integration with JConsole, which is the JMX-based management console provided with the JDK.



### Actional agents

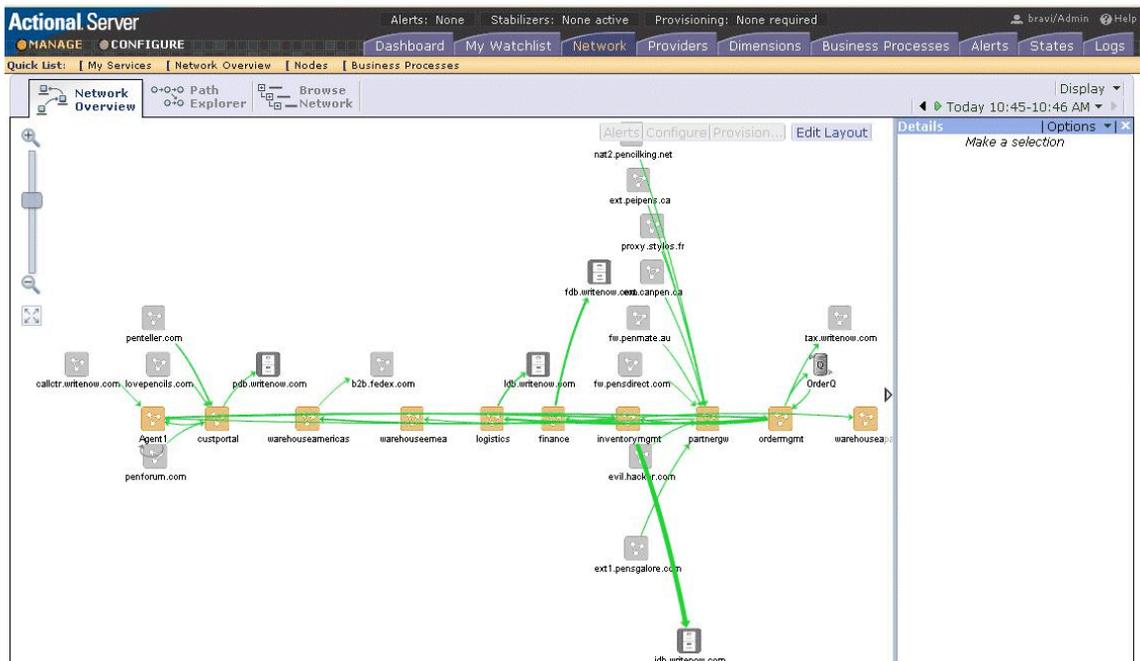
An Actional agent is run on each Artix node that you wish to manage. Actional agents are used to provide instrumentation data back to the Actional server. Actional agents are provisioned from the Actional server to establish initial contact and send configuration to the Actional agent.

### Artix interceptors

Artix interceptors are added to an endpoint's messaging chain that send the instrumentation data to the Actional agent using an Actional-specific API. These interceptors essentially push events to the Actional agent. The data is analyzed and stored in the Actional agent for retrieval by the Actional server.

Figure 3 shows an example system monitored in the **Actional Server Administration Console**.

**Figure 3:** *Actional Server Administration Console*



For more details on Artix integration with Progress Actional, see [Part III](#).

---

**AmberPoint**

Integration between Artix and AmberPoint enables the AmberPoint SOA management system to monitor Artix services. An Artix AmberPoint Agent can be deployed in Artix endpoints that use SOAP over HTTP to enable reporting of performance metrics back to AmberPoint.

The Artix AmberPoint Agent enables the use of the following AmberPoint features:

- Dynamic discovery of Artix clients and services using SOAP over HTTP.
- Monitoring of Artix client and service invocations, and reporting them back to AmberPoint.
- Mapping Qualities of Service to customer Service Level Agreements (SLAs).
- Monitoring of Artix invocation flow dependencies, which enables AmberPoint to draw Web service dependency diagrams.
- Centralized logging and performance statistics.

For more details on Artix integration with AmberPoint, see [Part IV](#).

---

**BMC Patrol**

Integration between Artix and BMC Patrol enables the BMC Patrol Enterprise Management System (EMS) to monitor Artix services. You can use the Artix integration with BMC Patrol to track key server metrics, such as server response times. You can also set up alarms and post events when a server crashes to enable specific recovery actions to be taken.

The Artix ESB C++ runtime integration with BMC Patrol, key server metrics are logged by the Artix performance logging plug-ins. Log file interpreting utilities are then used to analyze the logged data. Artix provides BMC Knowledge Modules (KM), which conform to standard BMC Patrol KM design and operation. These modules tell the BMC Patrol console how to interpret the data obtained from the Artix interceptors.

The Artix server metrics tracked by the Artix BMC Patrol integration include the number of invocations received, and the average, maximum and minimum response times. The Artix BMC Patrol integration also enables you to track these metrics for individual operations. Events can be generated when any of these parameters go out of bounds.

For more details on Artix integration with BMC Patrol, see [Part V](#).

# Part II

## Java Management Extensions

---

### **In this part**

This part contains the following chapters:

<a href="#">Monitoring and Managing with JMX</a>	<a href="#">page 43</a>
<a href="#">Configuring JMX in Artix C++</a>	<a href="#">page 67</a>
<a href="#">Managing Artix Services with JMX Consoles</a>	<a href="#">page 71</a>



# Monitoring and Managing with JMX

*This chapter explains how to monitor and manage an Artix C++ runtime using Java Management Extensions (JMX).*

---

**In this chapter**

This chapter discusses the following topics:

<a href="#">Introduction</a>	<a href="#">page 44</a>
<a href="#">Managed Bus Components</a>	<a href="#">page 49</a>
<a href="#">Managed Service Components</a>	<a href="#">page 55</a>
<a href="#">Managed Port Components</a>	<a href="#">page 63</a>

# Introduction

---

## Overview

You can use Java Management Extensions (JMX) to monitor and manage key Artix runtime components both locally and remotely. For example, using any JMX-compliant client, you can perform the following tasks:

- View bus status.
  - Stop or start a service.
  - Change bus logging levels dynamically.
  - Monitor service performance details.
  - View the interceptors for a selected port.
- 

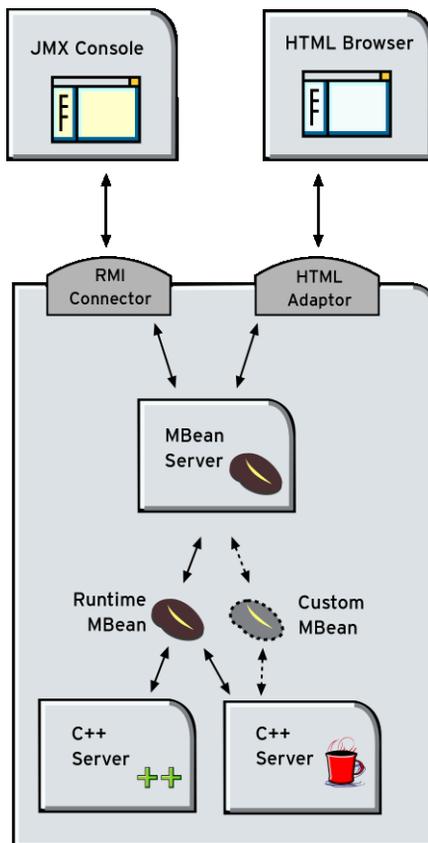
## How it works

Artix has been instrumented to allow runtime components to be exposed as JMX Managed Beans (MBeans). This enables an Artix runtime to be monitored and managed either in process or remotely with the help of the JMX Remote API.

Artix runtime components can be exposed as JMX MBeans, out-of-the-box. In addition, support for registering custom MBeans is also available. Java developers can create their own MBeans and register them either with their MBeanServer of choice, or with a default MBeanServer created by Artix (see [“Relationship between runtime and custom MBeans”](#) on page 46).

Figure 4 shows an overview of how the various components interact. The Java custom MBeans are optional components that can be added as required.

Figure 4: *Artix JMX Architecture*



---

## What can be managed

Artix C++ servers can have their runtime components exposed as JMX MBeans. The following components can be managed:

- Bus
- Service
- Port

All runtime components are registered with an `MBeanServer` as Open Dynamic MBeans. This ensures that they can be viewed by third-party management consoles without any additional client-side support libraries.

All MBeans for Artix runtime components conform with Sun's JMX Best Practices document on how to name MBeans (see <http://java.sun.com/products/JavaManagement/best-practices.html>). Artix runtime MBeans use `com.iona.instrumentation` as their domain name when creating `ObjectNames`.

**Note:** An `MBeanServerConnection`, which is an interface implemented by the `MBeanServer` is used in the examples in this chapter. This ensures that the examples are correct for both local and remote access.

See also “Further information” on page 48 for details of how to access MBean Server hosting runtime MBeans either locally and remotely.

---

## Relationship between runtime and custom MBeans

The Artix runtime instrumentation provides an out-of-the-box JMX view of C++ and JAX-RPC services. Java developers can also create custom JMX MBeans to manage Artix Java components such as services.

You may choose to write custom Java MBeans to manage a service because the Artix runtime is not aware of the current service's application semantics. For example, the Artix runtime can check service status and update performance counters, while a custom MBean can provide details on the status of a business loan request processing.

It is recommended that custom MBeans are created to manage application-specific aspects of a given service. Ideally, such MBeans should not duplicate what the runtime is doing already (for example, calculating service performance counters).

It is also recommended that custom MBeans use the same naming convention as Artix runtime MBeans. Specifically, runtime MBeans are named so that containment relationships can be easily established. For example:

```
// Bus :
com.iona.instrumentation:type=Bus,name=demos.jmx_runtime

Service :
com.iona.instrumentation:type=Bus.Service,name="{http://ws.iona.com}SOAPService",Bus=demos.jmx_runtime

// Port :
com.iona.instrumentation:type=Bus.Service.Port,name=SoapPort,Bus
.Service="{http://ws.iona.com}SOAPService",Bus=demos.jmx_runtime
```

Using these names, you can infer the relationships between ports, services and buses, and display or process a complete tree in the correct order. For example, assuming that you write a custom MBean for a loan approval Java service, you could name this MBean as follows:

```
com.iona.instrumentation:type=Bus.Service.LoanApprovalManager,name=LoanApprovalManager,Bus.Service="{http://ws.iona.com}SOAPService",Bus=demos.jmx_runtime
```

## Accessing the MBeanServer programmatically

Artix runtime support for JMX is enabled using configuration settings only. You do not need to write any additional Artix code. When configured, you can use any third party console that supports JMX Remote to monitor and manage Artix servers.

If you wish to write your own JMX client application, this is also supported. To access Artix runtime MBeans in a JMX client, you must first get a handle to the MBeanServer. The following code extract shows how to access the MBeanServer locally:

```
Bus bus = Bus.init(args);
MBeanServer mbeanServer =
    (MBeanServer)bus.getRegistry().getEntry(ManagementConstants.M
BEAN_SERVER_INTERFACE_NAME);
```

The following shows how to access the MBeanServer remotely:

```
// The address of the connector server
String url = "service:jmx:rmi://host:1099/jndi/artix";
JMXServiceURL address = new JMXServiceURL(url);

// Create the JMXConnectorServer
JMXConnector cntor = JMXConnectorFactory.connect(address, null);

// Obtain a "stub" for the remote MBeanServer
MBeanServerConnection mbsc = cntor.getMBeanServerConnection();
```

Please see the following demo for a complete example on how to access, monitor and manage Artix runtime MBeans remotely:

```
InstallDir\cxx_java\samples\advanced\management\jmx_runtime
```

## Further information

For further information, see the following URLs:

### JMX

<http://java.sun.com/products/JavaManagement/index.jsp>

### JMX Remote

<http://www.jcp.org/aboutJava/communityprocess/final/jsr160/>

### Open Dynamic MBeans

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/openmbean/package-summary.html>

### ObjectName

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html>

### MBeanServerConnection

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/MBeanServerConnection.html>

### MBeanServer

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/MBeanServer.html>

---

# Managed Bus Components

---

## Overview

This section describes the attributes and methods that you can use to manage JMX MBeans representing Artix bus components. For example, you can use any JMX client to perform the following tasks:

- View bus attributes.
- Enable monitoring of bus services.
- Dynamically change logging levels for known subsystems.

If you wish to write your own JMX client, this section describes methods that you can use to access Artix logging levels and subsystems, and provides a JMX code example.

---

## Bus MBean registration

When an Artix bus is initialized, a corresponding JMX MBean is created and registered for that bus with an MBeanServer.

C++

For example, in an Artix C++ application, this occurs after the following call:

```
Bus_var server_bus = Bus.init(argc, argv);
```

When a bus is shutdown, a corresponding MBean is unregistered from the MBeanServer.

---

## Bus naming convention

An Artix bus ObjectName uses the following convention:

```
com.iona.instrumentation:type=Bus,name=busIdentifier
```

---

## Bus attributes

The following bus component attributes can be managed by any JMX client:

**Table 1:** *Managed Bus Attributes*

Name	Description	Type	Read/Write
scope	Bus scope used to initialize a bus.	String	No

**Table 1:** *Managed Bus Attributes*

Name	Description	Type	Read/Write
<code>identifier</code>	Bus identifier, typically the same as its scope.	<code>String</code>	No
<code>arguments</code>	Bus arguments, including the executable name.	<code>String[]</code>	No
<code>servicesMonitoring</code>	Used to enable/disable services performance monitoring.	<code>Boolean</code>	Yes
<code>services</code>	A list of object names representing services on this bus.	<code>ObjectName[]</code>	No

`servicesMonitoring` is a global attribute which applies to all services and can be used to change a performance monitoring status.

**Note:** By default, service performance monitoring is enabled when JMX management is enabled in a standalone server, and disabled in an `it_container` process.

When using a JMX console to manage a `it_container` server, you can enable performance monitoring by setting the `serviceMonitoring` attribute to `true`.

`services` is a list of object names that can be used by JMX clients to build a tree of components. Given this list, you can find all other registered service MBeans that belong to this bus.

For examples of bus attributes displayed in a JMX console, see [“Managing Artix Services with JMX Consoles”](#) on page 71.

**Bus methods**

If you wish to write your own JMX client, you can use the following bus methods to access logging levels and subsystems:

**Table 2:** *Managed Bus Methods*

Name	Description	Parameters	Return Type
getLoggingLevel	Returns a logging level for a subsystem.	subsystem (String)	String
setLoggingLevel	Sets a logging level for a subsystem.	subsystem (String), level (String)	Boolean
setLoggingLevelPropagate	Sets a logging level for a subsystem with propagation.	subsystem (String), level (String), propagate (Boolean)	Boolean

All the attributes and methods described in this section can be determined by introspecting `MBeanInfo` for the Bus component (see <http://java.sun.com/j2se/1.5.0/docs/api/javax/management/MBeanInfo.htm>)

**Example JMX client**

The following code extract from an example JMX client application shows how to access bus attributes and logging levels:

```
MBeanServerConnection mbsc = ...;
String busScope = ...;
ObjectName busName = new ObjectName("com.ionainstrumentation:type=Bus,name=" + busScope);

if (mbsc.isRegistered(busName)) {
    throw new MBeanException("Bus mbean is not registered");
}

// MBeanInfo can be used to check for all known attributes and methods
MBeanInfo info = mbsc.getMBeanInfo(busName);

// bus scope
String scope = (String)mbsc.getAttribute(busName, "scope");
// bus identifier
String identifier = (String)mbsc.getAttribute(busName, "identifier");
// bus arguments
String[] busArgs = (String[])mbsc.getAttribute(busName, "arguments");
```

```

// check servicesMonitoring attribute, then disable and reenale it
Boolean status = (Boolean)mbsc.getAttribute(busName, "servicesMonitoring");
if (!status.equals(Boolean.TRUE)) {
    throw new MBeanException("Service monitoring should be enabled by default");
}

mbsc.setAttribute(busName, new Attribute("servicesMonitoring", Boolean.FALSE));
status = (Boolean)mbsc.getAttribute(busName, "servicesMonitoring");
if (!status.equals(Boolean.FALSE)) {
    throw new MBeanException("Service monitoring should be disabled now");
}

mbsc.setAttribute(busName, new Attribute("servicesMonitoring", Boolean.TRUE));
status = (Boolean)mbsc.getAttribute(busName, "servicesMonitoring");
if (!status.equals(Boolean.TRUE)) {
    throw new MBeanException("Service monitoring should be reenabled now");
}

// list of service MBeans
ObjectName[] serviceNames = (ObjectName[])mbsc.getAttribute(busName, "services");

// logging
String level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS"},
    new String[] {"subsystem"});
if (!level.equals("LOG_ERROR")) {
    throw new MBeanException("Wrong IT_BUS logging level");
}

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.INITIAL_REFERENCE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_ERROR")) {
    throw new MBeanException("Wrong IT_BUS.INITIAL_REFERENCE logging level");
}

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.CORE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_INFO_LOW")) {
    throw new MBeanException("Wrong IT_BUS.CORE logging level");
}

```

```

Boolean result = (Boolean)mbsc.invoke(
    busName,
    "setLoggingLevel",
    new Object[] {"IT_BUS", "LOG_WARN"},
    new String[] {"subsystem", "level"});

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS"},
    new String[] {"subsystem"});
if (!level.equals("LOG_WARN")) {
    throw new MBeanException("IT_BUS logging level has not been set properly");
}

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.INITIAL_REFERENCE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_WARN")) {
    throw new MBeanException("IT_BUS.INITIAL_REFERENCE logging level has not been set
properly");
}

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.CORE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_INFO_LOW")) {
    throw new MBeanException("IT_BUS.CORE logging level should not be changed");
}

// propagate
result = (Boolean)mbsc.invoke(
    busName,
    "setLoggingLevelPropagate",
    new Object[] {"IT_BUS", "LOG_SILENT", Boolean.TRUE},
    new String[] {"subsystem", "level", "propagate"});

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS"},
    new String[] {"subsystem"});

```

```
if (!level.equals("LOG_SILENT")) {
    throw new MBeanException("IT_BUS logging level has not been set properly");
}

level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.INITIAL_REFERENCE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_SILENT")) {
    throw new Exception("IT_BUS.INITIAL_REFERENCE logging level has not been set properly");
}
level = (String)mbsc.invoke(
    busName,
    "getLoggingLevel",
    new Object[] {"IT_BUS.CORE"},
    new String[] {"subsystem"});
if (!level.equals("LOG_SILENT")) {
    throw new MBeanException("IT_BUS.CORE logging level shouldve been set to LOG_SILENT");
}
```

---

**Further information**

For information on Artix logging levels and subsystems, see [Configuring and Deploying Artix Solutions, C++ Runtime](#).

---

# Managed Service Components

---

## Overview

This section describes the attributes and methods that you can use to manage JMX MBeans representing Artix service components. For example, you can use any JMX client to perform the following tasks:

- View managed services.
- Dynamically change a service status.
- Monitor service performance data.
- Manage service ports.

The Artix locator and session manager services, have also been instrumented. These provide an additional set of attributes on top of those common to all services. For information on WS-RM persistence instrumentation, see [Chapter 5](#).

If you wish to write your own JMX client, this section describes methods that you can use and provides a JMX code example.

---

## Service MBean registration

When an Artix servant is registered for a service, a JMX Service MBean is created and registered with an MBeanServer.

### C++

For example, in an Artix C++ application, this happens after the following call:

```
Bus_var server_bus = Bus.init(argc, argv);

BankServiceImpl servant;
bus->register_servant(
    servant,
    wsdl_location,
    QName("http://www.iona.com/bus/tests", "BankService")
);
```

When a service is removed, a corresponding MBean is unregistered from the MBeanServer.

**Service naming convention**

An Artix service `ObjectName` uses the following convention:

```
com.ionainstrumentation:type=Bus.Service,name="{namespace}local
name",Bus=busIdentifier
```

In this format, a name has an expanded service QName as its value. This value includes double quotes to permit for characters that otherwise would not be allowed.

**Service attributes**

The following service component attributes can be managed by any JMX client:

**Table 3:** *Managed Service Attributes*

Name	Description	Type	Read/Write
name	Service QName in expanded form.	String	No
state	Service state.	String	No
serviceCounters	Service performance data.	CompositeData	No
ports	A list of ObjectNames representing ports for this service.	ObjectName[]	No

name is an expanded QName, such as

```
{http://www.ionainstrumentation.com/bus/tests}BankService.
```

state represents a current service state that can be manipulated by stop and start methods.

ports is a list of ObjectNames that can be used by JMX clients to build a tree of components. Given this list, you can find all other registered Port MBeans which happen to belong to this Service.

**serviceCounters attributes**

The following service performance attributes can be retrieved from the `serviceCounters` attribute:

**Table 4:** *serviceCounters Attributes*

Name	Description	Type
<code>averageResponseTime</code>	Average response time in milliseconds.	Float
<code>requestsOneway</code>	Total number of oneway requests to this service.	Long
<code>requestsSinceLastCheck</code>	Number of requests happened since last check.	Long
<code>requestsTotal</code>	Total number of requests (including oneway) to this service.	Long
<code>timeSinceLastCheck</code>	Number of seconds elapsed since last check.	Long
<code>totalErrors</code>	Total number of request-processing errors.	Long

For examples of service attributes displayed in a JMX console, see [“Managing Artix Services with JMX Consoles”](#) on page 71

**Service methods**

If you wish to write your own JMX client, you can use the following service methods to manage a specific service:

**Table 5:** *Managed Service Attributes*

Name	Description	Parameters	Return Type
name	Start (activate) a service.	None	Void
state	Stop (deactivate) a service.	None	Void

All the attributes and methods described in this section can be accessed by introspecting `MBeanInfo` for the Service component.

**Example JMX client**

The following code extract from an example JMX client application shows how to access service attributes and methods:

```
MBeanServerConnection mbsc = ...;

String busScope = ...;
ObjectName serviceName = new ObjectName("com.iona.instrumentation:type=Bus.Service" +
    ",name=\"{http://www.iona.com/hello_world_soap_http}SOAPService\" "
    + ",Bus=" + busScope);

if (!mbsc.isRegistered(serviceName)) {
    throw new MBeanException("Service MBean should be registered");
}

// MBeanInfo can be used to check for all known attributes and methods
MBeanInfo info = mbsc.getMBeanInfo(serviceName);

// service name
String name = (String)mbsc.getAttribute(serviceName, "name");

// check service state attribute then reset it by invoking stop and start methods

String state = (String)mbsc.getAttribute(serviceName, "state");
if (!state.equals("ACTIVATED")) {
    throw new MBeanException("Service should be activated");
}

mbsc.invoke(serviceName, "stop", null, null);
```

```

state = (String)mbsc.getAttribute(serviceName, "state");
if (!state.equals("DEACTIVATED")) {
    throw new MBeanException("Service should be deactivated now");
}

mbsc.invoke(serviceName, "start", null, null);

state = (String)mbsc.getAttribute(serviceName, "state");
if (!state.equals("ACTIVATED")) {
    throw new MBeanException("Service should be activated again");
}

// check service counters

CompositeData counters = (CompositeData)mbsc.getAttribute(serviceName, "serviceCounters");
Long requestsTotal = (Long)counters.get("requestsTotal");
Long requestsOneway = (Long)counters.get("requestsOneway");
Long totalErrors = (Long)counters.get("totalErrors");
Float averageResponseTime = (Float)counters.get("averageResponseTime");
Long requestsSinceLastCheck = (Long)counters.get("requestsSinceLastCheck");
Long timeSinceLastCheck = (Long)counters.get("timeSinceLastCheck");

// ports
ObjectName[] portNames = (ObjectName[])mbsc.getAttribute(serviceName, "ports");

```

**Further information****MBeanInfo**

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/MBeanInfo.html>

**CompositeData**

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/openmbean/CompositeData.html>

---

## Artix Locator Service

---

### Overview

The Artix locator can also be exposed as a JMX MBean. A locator managed component is a service managed component that can be managed like any other bus service with the same set of attributes and methods. The Artix locator also exposes its own specific set of attributes.

---

### Locator attributes

An Artix locator MBean exposes the following locator-specific attributes:

**Table 6:** *Locator MBean Attributes*

Name	Description	Type
registeredEndpoints	Number of registered endpoints.	Integer
registeredServices	Number of registered services, less or equal to number of endpoints.	Integer
serviceLookups	Number of service lookup requests.	Integer
serviceLookupErrors	Number of service lookup failures.	Integer
registeredNodeErrors	Number of node (peer ping) failures.	Integer

**Example JMX client**

The following code extract from an example JMX client application shows how to access locator attributes and methods:

```
MBeanServerConnection mbsc = ...;
String busScope = ...;
ObjectName serviceName = new ObjectName("com.ionainstrumentation:type=Bus.Service" +
    ",name=\"{http://ws.ionainstrumentation.com/2005/11/locator}LocatorService\""
    + ",Bus=" + busScope);

// use common attributes and methods, see an example above

// Locator specific attributes
Integer regServices = (Integer)mbsc.getAttribute(serviceName, "registeredServices");
Integer endpoints = (Integer)mbsc.getAttribute(serviceName, "registeredEndpoints");
Integer nodeErrors = (Integer)mbsc.getAttribute(serviceName, "registeredNodeErrors");
Integer lookupErrors = (Integer)mbsc.getAttribute(serviceName, "serviceLookupErrors");
Integer lookups = (Integer)mbsc.getAttribute(serviceName, "serviceLookups");
```

---

## Artix Session Manager Service

---

### Overview

The Artix session manager can also be exposed as a JMX MBean. A session manager component is a service managed component that can be managed like any other bus service with the same set of attributes and methods. The Artix session manager also exposes its own specific set of attributes.

---

### Session manager attributes

An Artix session manager MBean exposes the following session manager-specific attributes:

**Table 7:** *Session Manager MBean Attributes*

Name	Description	Type
registeredEndpoints	Number of registered endpoints.	Integer
registeredServices	Number of registered services, less or equal to number of endpoints.	Integer
serviceGroups	Number of service groups.	Integer
serviceSessions	Number of service sessions	Integer

---

### Example JMX client

The following code extract from an example JMX client application shows how to access session manager attributes and methods:

```
MBeanServerConnection mbsc = ...;
String busScope = ...;
ObjectName serviceName = new ObjectName("com.iona.instrumentation:type=Bus.Service" +
    ",name=\"{http://ws.ionac.com/sessionmanager}SessionManagerService\" +", Bus=" +
    busScope);
// use common attributes and methods, see an example above

// SessionManager specific attributes
Integer regServices = (Integer)mbsc.getAttribute(serviceName, "registeredServices");
Integer endpoints = (Integer)mbsc.getAttribute(serviceName, "registeredEndpoints");
Integer serviceGroups = (Integer)mbsc.getAttribute(serviceName, "serviceGroups");
Integer serviceSessions = (Integer)mbsc.getAttribute(serviceName, "serviceSessions");
```

---

# Managed Port Components

---

## Overview

This section describes the attributes that you can use to manage JMX MBeans representing Artix port components. For example, you can use any JMX client to perform the following tasks:

- Monitor managed ports.
- View message and request interceptors.

If you wish to write your own JMX client, this section also shows an example of accessing these attributes in JMX code.

---

## Port MBean registration

Port managed components are typically created as part of a service servant registration. When service is activated, all supported ports will also be registered as MBeans.

When a service is removed, a corresponding Service MBean, as well as all its child Port MBeans are unregistered from the MBeanServer.

---

## Naming convention

An Artix port `ObjectName` uses the following convention:

```
com.iona.instrumentation:type=Bus.Service.Port,name=portName,Bus
.Service="{namespace}localname",Bus=busIdentifier
```

---

## Port attributes

The following bus component attributes can be managed by any JMX client:

**Table 8:** *Supported Service Attributes*

Name	Description	Type	Read/Write
name	Port name.	String	No
address	Transport specific address representing an endpoint.	String	No
interceptors	List of interceptors for this port.	String[]	No

**Table 8:** *Supported Service Attributes*

Name	Description	Type	Read/Write
transport	An optional attribute representing a transport for this port.	ObjectName[]	No

**interceptors**

The `interceptors` attribute is a list of interceptors for a given port. Internally, `interceptors` is an instance of `TabularData` that can be considered an array/table of `CompositeData`. However, due to a current limitation of `CompositeData`, (no insertion order is maintained, which makes it impossible to show interceptors in the correct order), the interceptors are currently returned as a list of strings, where each `String` has the following format:

```
[name]: name [type]: type [level]: level [description]: optional
description
```

In this format, `type` can be `CPP` or `Java`; `level` can be `Message` or `Request`.

It is most likely that this limitation will be fixed in a future JDK release, probably JDK 1.7 because the enhancement request has been accepted by Sun. In the meantime, interceptors details can be retrieved by parsing a returned `String` array.

For examples of port attributes displayed in a JMX console, see [“Managing Artix Services with JMX Consoles” on page 71](#)

**Example JMX client**

The following code extract from an example JMX client application shows how to access port attributes and methods:

```
MBeanServerConnection mbsc = ...;

String busScope = ...;
ObjectName portName = new ObjectName("com.ionainstrumentation:type=Bus.Service.Port" +
    ",name=SoapPort" +

    ",Bus.Service=\"{http://www.ionainstrumentation.com/hello_world_soap_http}SOAPService\" " + ",Bus=" +
    busScope);

if (!mbsc.isRegistered(portName)) {
    throw new MBeanException("Port MBean should be registered");
}

// MBeanInfo can be used to check for all known attributes and methods
MBeanInfo info = mbsc.getMBeanInfo(portName);

// port name
String name = (String)mbsc.getAttribute(portName, "name");

// port address
String address = (String)mbsc.getAttribute(portName, "address");

// check interceptors

String[] interceptors = (String[])mbsc.getAttribute(portName, "interceptors");
if (interceptors.length != 6) {
    throw new MBeanException("Number of port interceptors is wrong");
}

handleInterceptor(interceptors[0],
    "MessageSnoop",
    "Message",
    "CPP");
handleInterceptor(interceptors[1],
    "MessagingPort",
    "Request",
    "CPP");
handleInterceptor(interceptors[2],
    "http://schemas.xmlsoap.org/wsdl/soap/binding",
    "Request",
    "CPP");
```

```

handleInterceptor(interceptors[3],
    "TestInterceptor",
    "Request",
    "Java");
handleInterceptor(interceptors[4],
    "bus_response_monitor_interceptor",
    "Request",
    "CPP");
handleInterceptor(interceptors[5],
    "ServantInterceptor",
    "Request",
    "CPP");

```

For example, the `handleInterceptor()` function may be defined as follows:

```

private void handleInterceptor(String interceptor,
    String name,
    String level,
    String type) throws Exception {
    if (interceptor.indexOf("[name]: " + name) == -1 ||
        interceptor.indexOf("[type]: " + type) == -1 ||
        interceptor.indexOf("[level]: " + level) == -1) {

        throw new MBeanException("Wrong interceptor details");
    }
    // analyze this interceptor further
}

```

# Configuring JMX in Artix C++

*This chapter explains how to configure an Artix C++ runtime to be managed with Java Management Extensions (JMX).*

---

**In this chapter**

This chapter discusses the following topic:

<a href="#">Artix JMX Configuration</a>
---

page 68
---------

---

# Artix JMX Configuration

---

## Overview

This section explains the Artix configuration variable settings that you must configure to enable JMX monitoring of the Artix runtime, and access for remote JMX clients.

## Enabling the management plugin

To expose the Artix runtime using JMX MBeans, you must enable a `bus_management` plug-in as follows:

```
jmx_local
{
    plugins:bus_management:enabled="true";
};
```

This setting enables local access to JMX runtime MBeans. The `bus_management` plug-in wraps runtime components into Open Dynamic MBeans and registers them with a local MBeanServer.

## Configuring remote JMX clients

To enable remote JMX clients to access runtime MBeans, use the following configuration settings:

```
jmx_remote
{
    plugins:bus_management:enabled="true";
    plugins:bus_management:connector:enabled="true";
};
```

These settings allow for both local and remote access.

### Specifying a remote access URL

Remote access is performed through JMX Remote, using an RMI Connector on a default port of 1099. Using this configuration, you can use the following JNDI-based JMXServiceURL to connect remotely:

```
service:jmx:rmi:///jndi/rmi://host:1099/artix
```

### Configuring a remote access port

To specify a different port for remote access, use the following configuration variable:

```
plugins:bus_management:connector:port="2000";
```

You can then use the following JMXServiceURL:

```
service:jmx:rmi:///jndi/rmi://host:2000/artix
```

---

### Configuring a stub-based JMXServiceURL

You can also configure the connector to use a stub-based JMXServiceURL as follows:

```
jmx_remote_stub
{
  plugins:bus_management:enabled="true";
  plugins:bus_management:connector:enabled="true";
  plugins:bus_management:connector:registry:required="false";
};
```

See the [javax.management.remote.rmi](#) package for more details on remote JMX.

---

### Publishing the JMXServiceURL to a local file

You can also request that the connector publishes its JMXServiceURL to a local file:

```
plugins:bus_management:connector:url:publish="true";
```

The following entry can be used to override the default file name:

```
plugins:bus_management:connector:url:file="../../service.url";
```

**Further information**

---

For further information, see the following:

**RMI Connector**

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/remote/rmi/RMIConnector.html>

**JMXServiceURL**

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/remote/JMXServiceURL.html>

<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/remote/rmi/package-summary.html>

# Managing Artix Services with JMX Consoles

*You can use third-party management consoles that support JMX Remote to monitor and manage Artix services (for example, JConsole). You can view the status of a bus instance, stop or start a service, change bus logging levels, or view interceptor chains.*

---

**In this chapter**

This chapter discusses the following topics:

<a href="#">Managing Artix Services with JConsole</a>	page 72
<a href="#">Managing Artix Services with the JMX HTTP adaptor</a>	page 76

# Managing Artix Services with JConsole

---

## Overview

You can use JConsole, which is provided with JDK, to monitor and manage Artix applications. JConsole displays Artix runtime managed components in a hierarchical tree, as shown in [Figure 5](#).

---

## Using JConsole

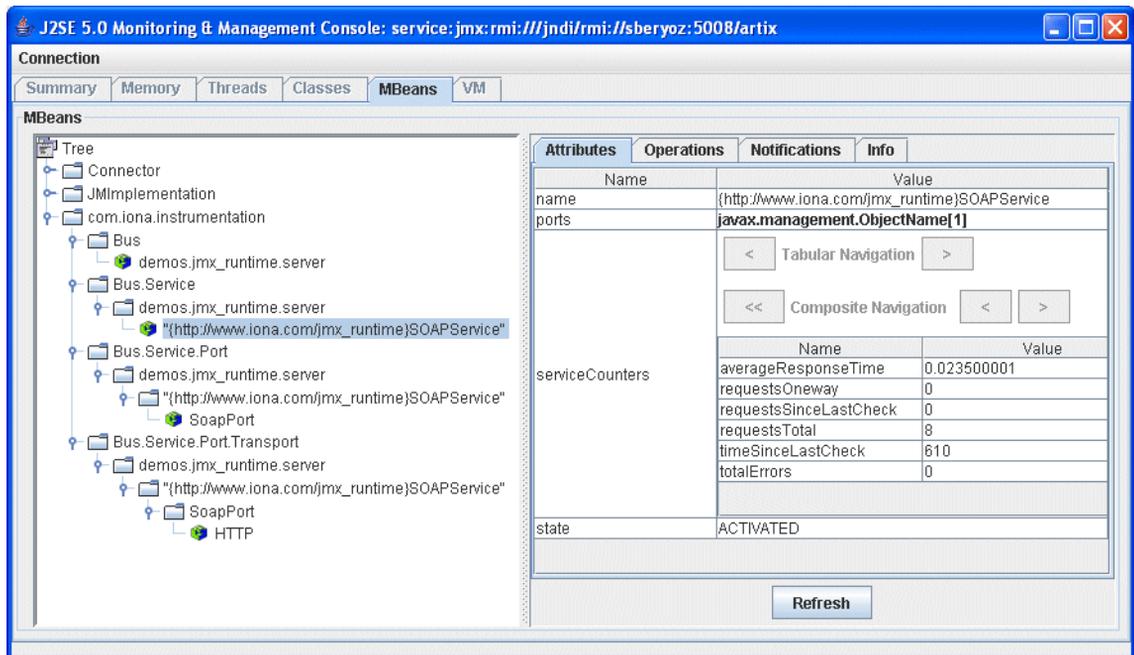
To use JConsole:

1. Start up JConsole using the following command: `JDK_HOME/bin/jconsole`
2. Select the **Advanced** tab.
3. Enter or paste a `JMXServiceURL` (either the default URL, or one copied from a published `connector.url` file).

## Managing services

Figure 5 shows the attributes displayed for a managed service component (for example, the `serviceCounters` performance metrics displayed in the right pane). For detailed information on these attributes, see “Service attributes” on page 56.

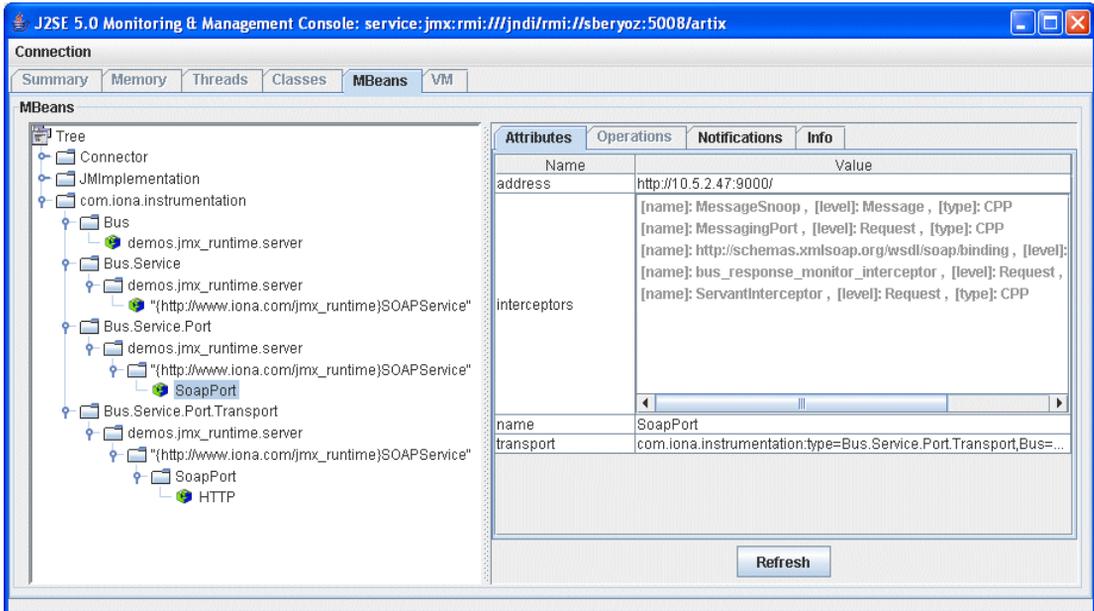
**Figure 5:** *Managed Service in JConsole*



Managing ports

Figure 6 shows the attributes displayed for a managed port component (for example, the `interceptors` list displayed in the right pane). For detailed information on these attributes, see “Port attributes” on page 63.

Figure 6: Managed Port in JConsole



## Managing containers

Figure 7 shows an example of a locator service deployed into an Artix container. For more information, see “Locator attributes” on page 60.

**Figure 7:** *Managed Locator in JConsole*

The screenshot displays the J2SE 5.0 Monitoring & Management Console interface. The title bar indicates the connection to a service at `service:jmx:rmi:///jndi/rmi://sberyoz:5008/artix`. The 'MBeans' tab is selected, showing a tree view of the MBean hierarchy. The selected MBean is `(http://ws.iona.com/2005/11/locator)LocatorService`. The right-hand pane shows the 'Attributes' tab for this MBean, displaying various performance and configuration metrics.

Name	Value														
name	(http://ws.iona.com/2005/11/locator)LocatorService														
ports	javax.management.ObjectName[1]														
registeredEndpoints	1														
registeredNodeErrors	0														
registeredServices	1														
<div style="text-align: center;"> <input type="button" value="Tabular Navigation"/> </div>															
<div style="text-align: center;"> <input type="button" value="Composite Navigation"/> </div>															
<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>averageResponseTime</td> <td>0.0010</td> </tr> <tr> <td>requestsOneway</td> <td>0</td> </tr> <tr> <td>requestsSinceLastCheck</td> <td>1</td> </tr> <tr> <td>requestsTotal</td> <td>1</td> </tr> <tr> <td>timeSinceLastCheck</td> <td>3</td> </tr> <tr> <td>totalErrors</td> <td>0</td> </tr> </tbody> </table>		Name	Value	averageResponseTime	0.0010	requestsOneway	0	requestsSinceLastCheck	1	requestsTotal	1	timeSinceLastCheck	3	totalErrors	0
Name	Value														
averageResponseTime	0.0010														
requestsOneway	0														
requestsSinceLastCheck	1														
requestsTotal	1														
timeSinceLastCheck	3														
totalErrors	0														
serviceLookupErrors	0														
serviceLookups	1														
state	ACTIVATED														

**Note:** When using a JMX console to manage a service running in an Artix container, set the `serviceMonitoring` attribute to `true` to enable service performance monitoring (see “Bus attributes” on page 49).

## Further information

For more information on using JConsole, see the following:

<http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html>

---

# Managing Artix Services with the JMX HTTP adaptor

## Overview

---

You can also manage Artix services using the default HTTP adaptor console that is provided with the JMX reference implementation. This console is browser-based, as shown in [Figure 8](#).

---

## Using the JMX HTTP adaptor

To use the JMX HTTP adaptor:

1. Specify following configuration settings:

```
plugins:bus_management:http_adaptor:enabled="true";  
plugins:bus_management:http_adaptor:port="7659";
```

2. Enter the following URL in your browser:  
`http://localhost:7659`

This displays the main HTTP adaptor management view, as shown in Figure 8.

**Figure 8:** HTTP Adaptor Main View

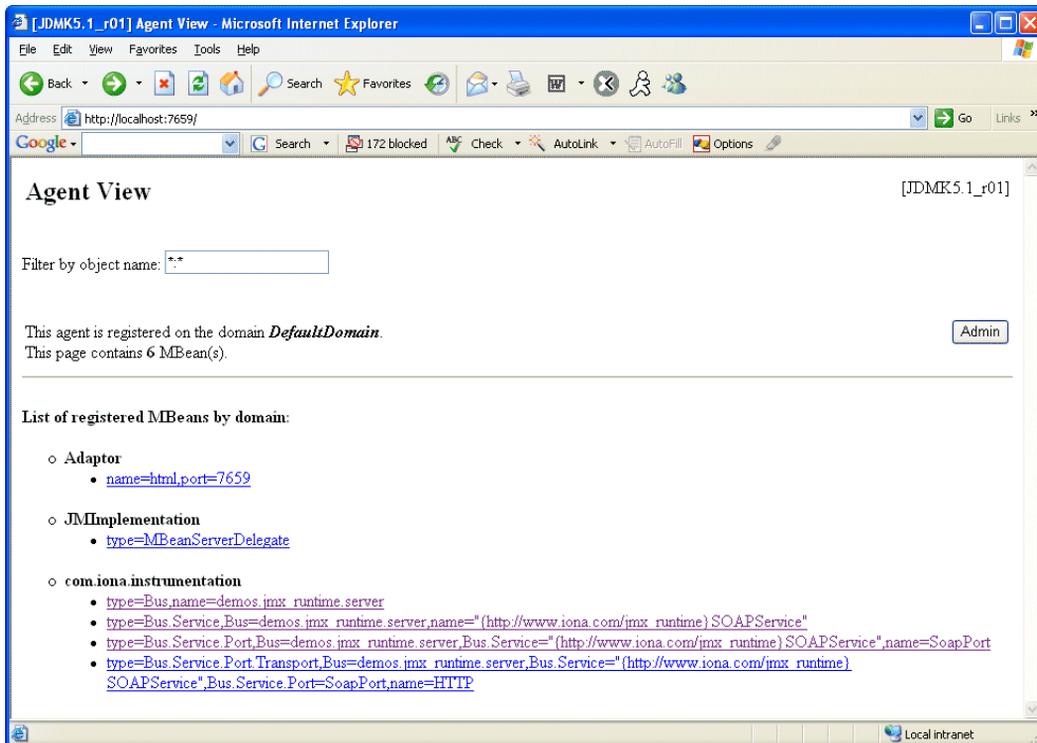


Figure 9 shows the attributes displayed for a managed bus component (for example, the `services` that it includes). For detailed information on these attributes, see “Bus attributes” on page 49.

**Figure 9:** *HTTP Adaptor Bus View*

Back to Agent View

Reload Period in seconds: 0 Reload Unregister

**MBean description:**  
Bus

**List of MBean attributes:**

Name	Type	Access	Value
<a href="#">arguments</a>	java.lang.String[]	R/O	<a href="#">view the values of arguments</a>
<a href="#">identifier</a>	java.lang.String	R/O	art
<a href="#">scope</a>	java.lang.String	R/O	demos.jmx_runtime.server
<a href="#">services</a>	javax.management.ObjectName[]	R/O	<a href="#">view the values of services</a>
<a href="#">servicesMonitoring</a>	java.lang.Boolean	R/W	<input checked="" type="radio"/> True <input type="radio"/> False

Apply

### Further information

For further information on using the HTTP JMX adaptor, see the following:

<http://java.sun.com/developer/technicalArticles/J2SE/jmx.html>

# Managing WS-RM Persistence with JMX

*You can manage Web Services Reliable Messaging persistence in Artix using any JMX console.*

---

**In this chapter**

This chapter discusses the following topics:

<a href="#">WS-RM Persistence Management</a>	<a href="#">page 80</a>
<a href="#">Viewing Messages in the WS-RM Persistence Database</a>	<a href="#">page 82</a>

---

# WS-RM Persistence Management

---

## Overview

You can use any JMX console to view messages in the WS-RM persistence database both locally and remotely. You also can monitor the WS-RM persistence enabled endpoint, the WS-RM acksTo endpoint URI, and the client's RM source endpoint. This section explains the WS-RM persistence information that can be managed in a JMX console.

---

## Managed WS-RM persistence components

The following WS-RM persistence components can be managed in a JMX console:

- Managed WS-RM persistence endpoints (*RMEndpointPersistentStore*)
  - Managed WS-RM persistence sequences (*RMSequencePersistentStore*)
- 

## Managed WS-RM persistence endpoints

WS-RM persistence endpoint managed components are used to represent WS-RM persistence enabled endpoints. When a WS-RM persistence destination endpoint is created, it is registered as an MBean. When an WS-RM persistence destination endpoint is closed, the MBean is unregistered from the MBeanServer.

The MBean naming convention is as follows:

```
com.ionainstrumentation:type=Bus.Service.Port.EndpointPersistent,
name=WSRM_ENDPOINT_PERSISTENCE,
Bus.Service.Port=portName,
Bus.Service="{namespace}localname",
Bus=busIdentifier
```

## WS-RM persistence endpoint attributes

You can view the following attributes for a WS-RM persistence endpoint in a JMX console:

Name	Description	Type
service name	WS-RM persistence enabled service name	String
port name	WS-RM persistence enabled port	String

## Managed WS-RM persistence sequences

WS-RM persistence sequence managed components are used to represent WS-RM sequences. A destination sequence with a unique ID is created for each client. When a WS-RM persistence destination sequence is created, it is registered as an MBean. When a WS-RM persistence destination sequence is recovered from database, it is also registered as an MBean. When a WS-RM persistence destination sequence is terminated, it is unregistered from the MBeanServer.

The MBean naming convention is as follows:

```
com.iona.instrumentation:type=Bus.Service.Port.EndpointPersistent.SequencePersistent,
name=sequenceName,
Bus.Service.Port.EndpointPersistent=WSRM_ENDPOINT_PERSISTENCE,
Bus.Service.Port=portName,
Bus.Service="(namespace)localName",
Bus=busIdentifier
```

In this syntax, *sequenceName* includes the string *sequence\_id* and the sequence ID.

### WS-RM persistence sequence attributes

You can view the following attributes for a WS-RM persistence sequence in a JMX console:

Name	Description	Type
acksto uri	WS-RM acknowledgement URI	String
messages	Messages in the WS-RM persistence database	String[]
sequence id	Sequence unique ID representing a client	String

The messages attribute is a list of messages in the WS-RM persistence database. The messages are returned as a list of strings, where each string has the following format:

```
[message id]: messageId [message]: soapMessage
```

---

# Viewing Messages in the WS-RM Persistence Database

---

## Overview

Before you start viewing in the WS-RM persistence database, you must set your Artix configuration to enable JMX management for WS-RM persistence. This section uses the Artix WS-RM sample application to explain how to view and monitor messages in the WS-RM persistence database.

---

## Enable JMX management for WS-RM persistence

To enable JMX management for WS-RM persistence in your Artix configuration file, perform the following steps:

1. Open the following file:

```
ArtixInstallDir\samples\advanced\wsm\etc\wsm.cfg
```

2. Edit the `demos.wsm_persistence_enabled.server` scope as follows:

```
server {
  plugins:artix:db:home = "./server_db";
  plugins:bus_management:enabled="true";
  plugins:bus_management:connector:enabled="true";
  plugins:bus_management:connector:url:file="../../etc/connector.url";

  # optional port, default is 1099
  plugins:bus_management:connector:port="5008";
};
```

**Note:** Enabling JMX management for WS-RM persistence is similar to enabling JMX management for other Artix components.

**Start the server**

3. To start the server, go to the following directory:

```
InstallDir\samples\advanced\wsm\bin\
```

4. Run the following command:

```
run_cxx_server_persistence.bat
```

This starts the server using following example command:

```
start server.exe -ORBname demos.wsm_persistence_enabled.server
```

When the server runs, a file named `connector.url` is created in the `...samples\advanced\wsm\etc\` directory.

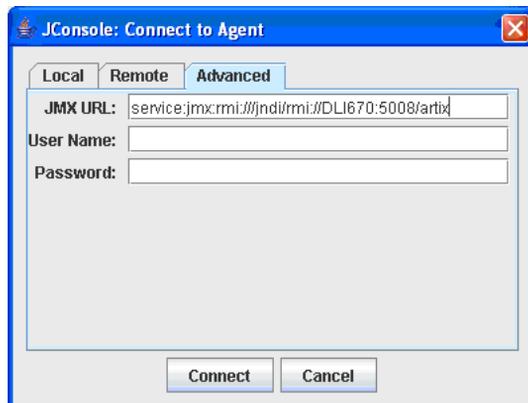
**Start a JMX console**

You can start any JMX console. For example, to start JConsole, execute the following command:

```
%JAVA_HOME%\bin\jconsole.exe
```

This displays the **JConsole: Connect to Agent** dialog, as shown in [Figure 10](#).

**Figure 10:** *Connecting to a JMX Agent*

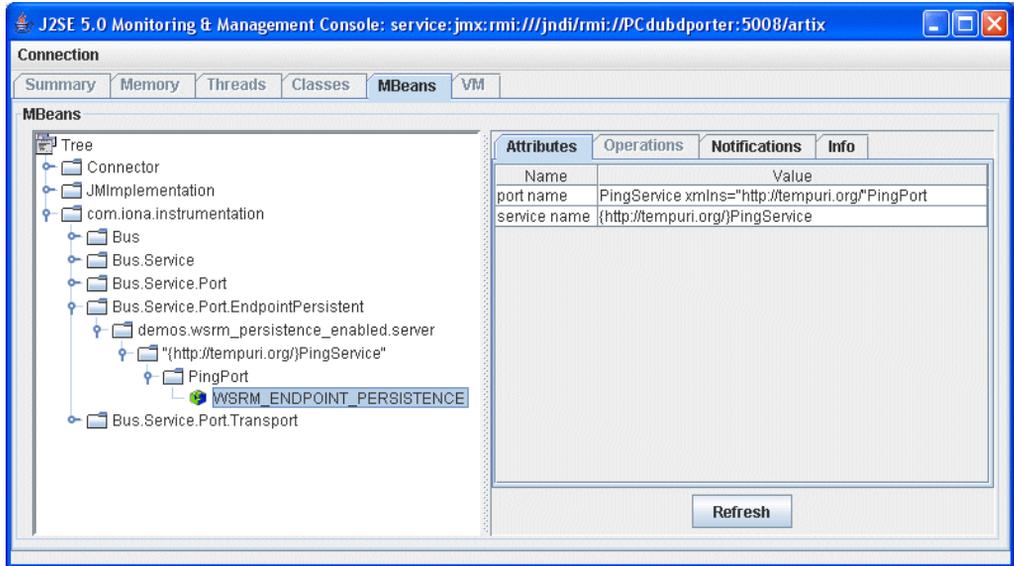


Copy the contents of the `connector.url` file into the JMX URL field, and click **Connect**. This displays the **J2SE 5.0 Monitoring and Management Console**, as shown in [Figure 11](#).

### View WS-RM persistence enabled endpoints

You can view a WS-RM persistence enabled endpoint in the **MBeans** tab of the JMX console, as shown in [Figure 11](#):

**Figure 11:** *WS-RM Persistence Enabled Endpoint*



In this example, PingPort is a WS-RM persistence enabled port. You can view the port and service name in the **Attributes** tab on the right of the console.

## View messages in the WS-RM persistence database

To view messages in the WS-RM persistence database, perform the following steps:

1. Edit the client code in

... \samples\advanced\wsmr\cxx\client\PingClientSample.cxx as follows:

```
int
run_persistence_client(
    int argc,
    char* argv[]
)
...
for (int i=0; i < 10; i++)
{
    cout << "Invoking PingOneway " << i << endl;
    PingType param1;
    param1.setText("PingOneway message from client");
    client1.PingOneway(param1);
    cout << i << " PingOneway invoked" << endl;
}
...
```

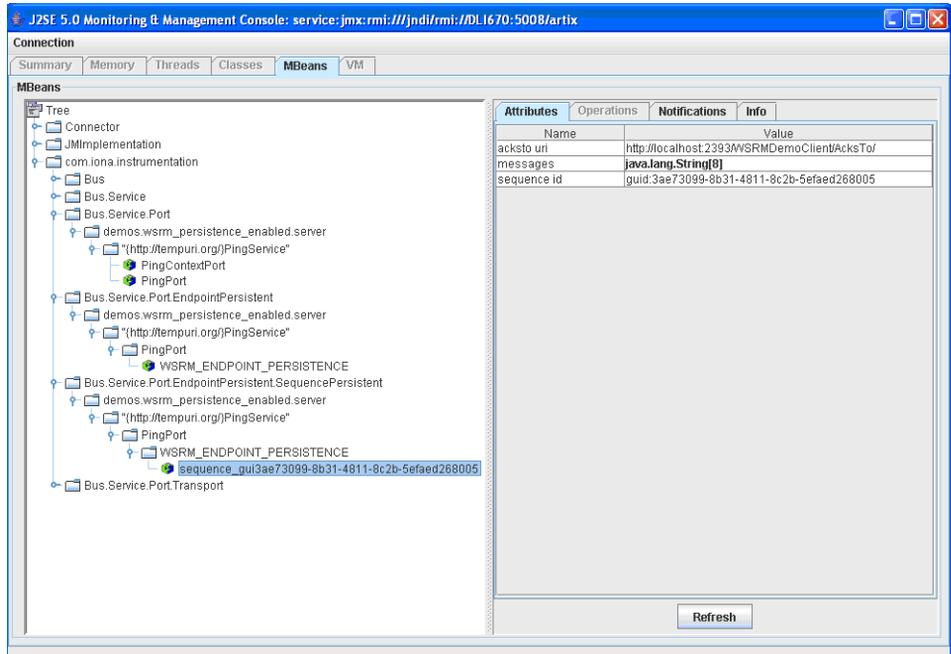
This adds a loop to the client that invokes the server 10 times in order to easily view messages in WS-RM persistence database.

2. Start the client. For example, go to the ... \samples\advanced\wsmr\bin directory, and run the following command:

```
run_cxx_client_persistence.bat
```

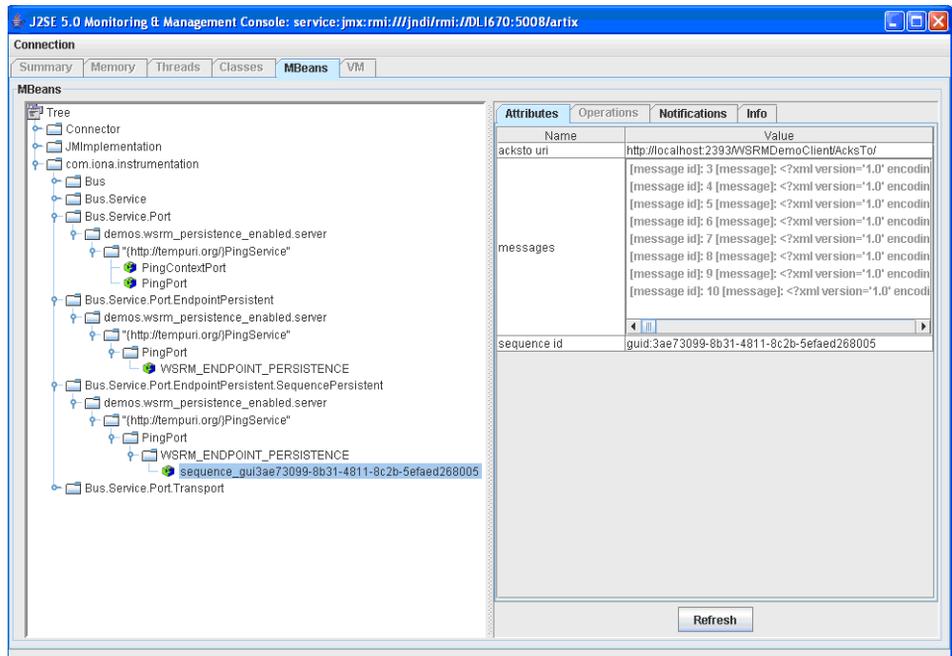
3. You can view the attributes for the WS-RM sequence in the JMX console, as shown in Figure 12. The WS-RM sequence name consists of the `sequence_guid` string and a sequence ID.

**Figure 12:** *WS-RM Sequence Attributes*



- You can view all the messages in WS-RM persistence database by clicking in the **Attributes** tab on the right of the console, as shown in [Figure 13](#). Each message consists of a message ID and a SOAP message.

**Figure 13:** Messages in the WS-RM Persistence database



You can click the **Refresh** button to view the current messages in WS-RM persistence database.



# Part III

## Progress Actional

---

### **In this part**

This part contains the following chapters:

<a href="#">Artix–Actional Integration</a>	<a href="#">page 91</a>
<a href="#">Configuring Artix–Actional Integration</a>	<a href="#">page 99</a>



# Artix–Actional Integration

*Artix provides support for integration with Progress Actional SOA management products.*

---

**In this chapter**

This chapter includes the following section:

<a href="#">Artix–Actional Interaction Architecture</a>
---

page 92
---------

---

# Artix–Actional Interaction Architecture

---

## Overview

Integration between Artix and Actional enables Artix services to be monitored by Actional SOA management products. For example, you can use Actional SOA management tools to perform discovery, monitoring, auditing, and reporting on Artix services and consumers. You can also correlate and track all messages through your SOA network to perform dependency mapping and root cause analysis.

The Artix–Actional integration is deployed on Artix systems to enable reporting of management data back to the Actional server. The data reported back to Actional includes system administration metrics such as response time, fault location, auditing, and alerts based on policies and rules.

This guide explains how to integrate Artix applications written in C++.

---

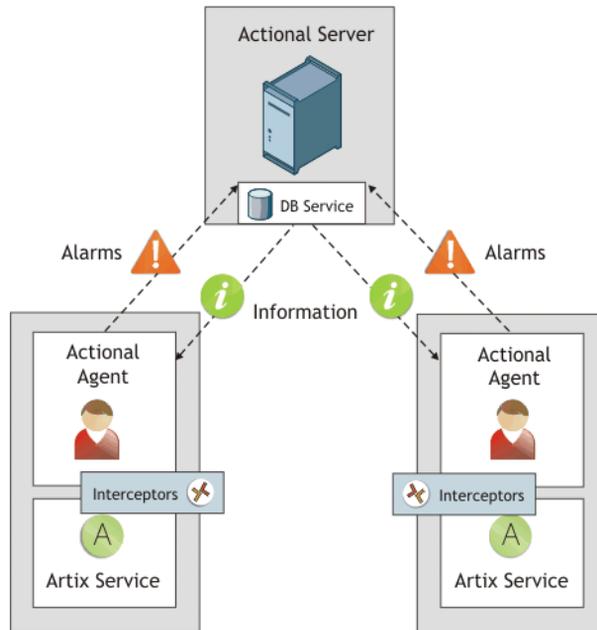
## Artix–Actional integration architecture

The Actional SOA management system includes an Actional server and an Actional agent. The Actional agent is run on each node that you wish to manage. A node is defined as a system on the current network. A node with an Actional agent installed is referred to as an *instrumented node* or a *managed node*.

The managed node uses Actional’s interceptor API to send monitoring data to the Actional agent. The Actional server pings the Actional agent periodically to retrieve the monitoring data. It analyzes this data and represents it in the Actional SOA management GUI tools. In addition, any alerts triggered at the Actional agent are sent immediately to the Actional server.

[Figure 14](#) shows how Artix Web service applications are integrated with Actional using this architecture.

**Figure 14:** *Artix-Actional Integration Architecture*



The main components in this architecture are:

- “Actional server”
- “Actional agent”
- “Artix interceptors”
- “Actional agent interceptor API”
- “Artix service endpoints”
- “Service consumers”

---

**Actional server**

The Actional server is a central management server that manages nodes containing an Actional agent.

The Actional server hosts a database and pings Actional agents to obtain management data at configured time intervals. It analyzes the management data and displays it in an Actional console—for example, the **Actional Server Administration Console**. This is a Web application deployed on Apache Tomcat, runtime management and agent configuration modes.

By default, the Actional server uses port 4040. The default Actional server database is Apache Derby.

---

**Actional agent**

An Actional agent is run on each Artix node that you wish to manage. Actional agents are used to provide instrumentation data back to the Actional server.

Actional agents are provisioned from the Actional server to establish initial contact and send configuration to the Actional agent. There is one Actional agent per managed node. By default, the Actional agent uses port 4041.

---

**Artix interceptors**

At the level of a managed node, Artix interceptors send the instrumentation data to the Actional agent using an Actional-specific API. These interceptors essentially push events to the Actional agent.

The data is analyzed and stored in the Actional agent for retrieval later by the Actional server. However, any alerts triggered at the Actional agent are sent immediately to the Actional server.

---

**Actional agent interceptor API**

The Actional Agent Interceptor C++ SDK is an Actional-specific API used to send the management instrumentation data from the service endpoint to the Actional agent.

The Artix service application to be managed by Actional must use the Actional Agent Interceptor C++ SDK to send monitoring data to the Actional agent. For detailed information on how to use this API, see the Actional product documentation.

---

**Artix service endpoints**

An Artix service endpoint is a service built using Artix, and described using WSDL. The endpoint can be implemented in C++. However, the main characteristic of an Artix service endpoint is that it can be described in WSDL, and classified as a service, which can be consumed.

---

**Service consumers**

Service consumers are clients that consume service endpoints by exchanging messages based on the service interface. Consumers can be built using Artix, or any product that supports the technology used by the endpoint. For example, a pure CORBA client could be a consumer for a CORBA endpoint. A .NET client could be a consumer for an Artix SOAP endpoint.

---

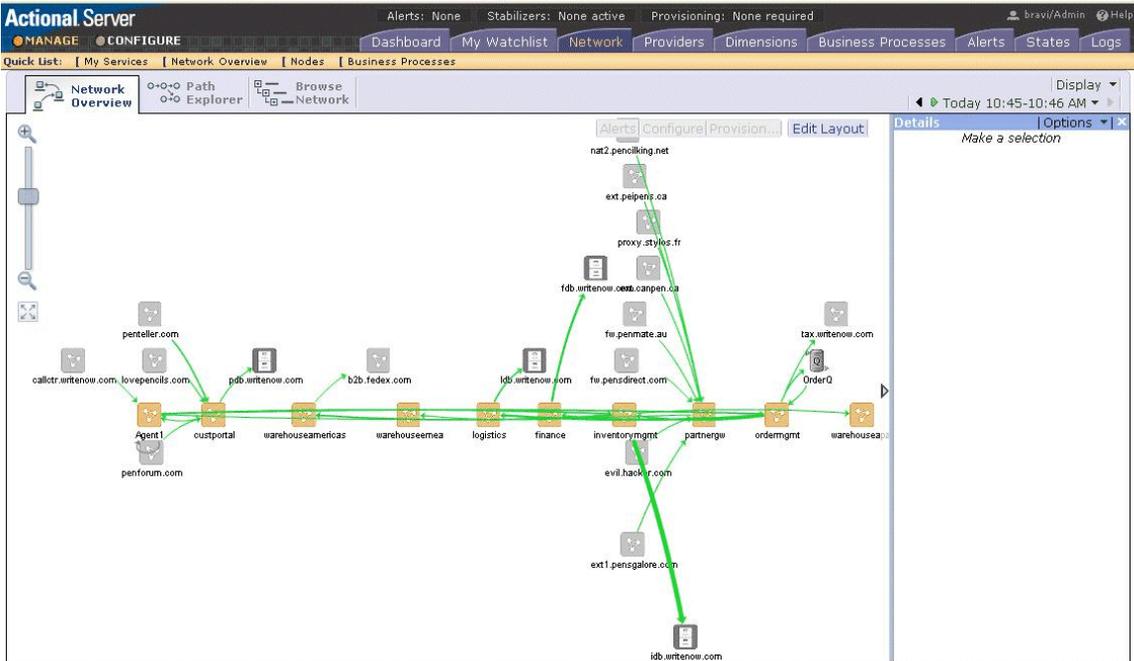
**Actional SOA management system**

In this document, Actional is the general term used to describe the Actional SOA management system in which all data is stored and viewed. This simplifies the architecture of Actional for the sake of this discussion.

[Figure 15](#) shows an example of the **Actional Server Administration Console**. Managed nodes are displayed as orange boxes, and unmanaged nodes are displayed as grey boxes. The green arrow indicates the message flow through various nodes.

Clicking on each of the nodes shows more in-depth information regarding the response time, alerts and warnings, and so on. The organization of the information in this web console is in the form of *Node–Group–Service–Operation*. In Artix, this translates to *Node–Service–Port–Operation*.

Figure 15: Actional Server Administration Console



**Further information**

For detailed information on using Actional features, see the Actional product documentation.



# Configuring Artix– Actional Integration

*This chapter explains how to configure integration between Artix and Actional SOA management products, and shows examples from Artix-Actional integration demos.*

---

**In this chapter**

This chapter includes the following sections:

<a href="#">Prerequisites</a>	<a href="#">page 100</a>
<a href="#">Configuring Actional for Artix Integration</a>	<a href="#">page 101</a>
<a href="#">Configuring Artix for Actional Integration</a>	<a href="#">page 104</a>
<a href="#">Viewing Artix Endpoints in Actional</a>	<a href="#">page 106</a>

# Prerequisites

---

## Overview

This section describes prerequisites for integration between Artix and Actional SOA management products.

The Actional for SOA Operations product is aimed at a technical audience (for example, system administrators managing services on the network). While the Actional for Continuous Service Optimization (Actional CSO) product is aimed at a business audience.

---

## Supported product versions

Artix supports integration with the following Actional product versions:

- Actional Point of Operational Visibility 8.1 and above.
  - Actional Management Server 8.1 and above.
- 

## Supported protocols and transports

The following protocols and transports are supported:

- SOAP over HTTP
  - SOAP over JMS
- 

## Actional agents

The Actional agent component is also known as the Actional Point of Operational Visibility.

You must ensure that Actional agents have been set up on each Artix node that you wish to manage. The provisioning of Actional agents is performed using the Actional server. For some basic details, see [“Configuring Actional for Artix Integration” on page 101](#).

For information on how to set up Actional agents on managed nodes, see the Actional product documentation.

---

## Further information

For information on the full range of platform versions and database versions supported by Actional, see the Actional product documentation.

This Artix integration with Actional supports the full range of operating system platforms supported by Artix. For more details, see the [Artix Installation Guide](#).

---

# Configuring Actional for Artix Integration

---

## Overview

This section provides some basic configuration guidelines for Actional agent and server configuration. For full details, see the Actional product documentation.

This basic configuration will help to set up the Artix–Actional integration demos. For information on how to run these demos, see the `readme.txt` files in the following directories:

```
ArtixInstallDir/samples/advanced/management/actional/soap_over_http  
ArtixInstallDir/samples/advanced/management/actional/soap_over_jms
```

---

## Actional agent configuration

No specific Actional agent configuration settings are required for integration with Artix. For example, for the purposes of the Actional-Artix integration demos, the Actional agent can be started with the default configuration settings.

---

## Actional server configuration

The following sample configuration steps describe how to set up the Actional server to run a simple Artix-Actional demo:

1. Install the Actional server with typical installation options, and select the Apache Derby database.
2. Specify the following URL in your browser:  
`http://localhost:4040/lgservlet`
3. If this is a new installation click **Start**, and follow the new Actional server setup steps.

Otherwise, if the Actional server is already installed, perform the following steps:

- i. In the Actional console Web interface, select the **Configure** radio button in the top left of the screen.
- ii. Select **Platform** tab. This displays the general configuration settings.

---

**Creating a managed node**

To create a managed node for a simple Artix demo, perform the following steps:

1. In the Actional **Configure** view menu bar, open the **Network** tab. This displays the **Network Nodes**.
  2. Select **Add**. This displays **Node Creation / Managing Agents**.
  3. Click **Managed Node**.
- 

**Configuring a new node**

To configure a managed node for the demo, perform the following steps in the wizard:

**Step 1: New Node - Identification**

1. Specify the **Name** as `agent1`.
2. Specify the **Display icon** as `auto-discover` (you can select `IONA Artix` from the drop down list, if desired).
3. Click **Next**.

**Step 2: New Node - Management**

1. Specify the **Transport** as `HTTP/S`.
2. Supply the Actional agent user name and password.
3. Ensure that **Override Agent Database** is checked.
4. Click **Next**.

**Step 3: New Node - Agents**

1. Specify the following URL:  
`http://HostName:4041/lagent`  
You can specify a host name or an `IP_ADDRESS`.
2. Click **Add**. The agent URL is added.
3. Click **Next**.

**Step 4: New Node - Endpoints**

1. For **Endpoints**, add the hostname, fully qualified hostname, and IP address.
2. Click **Next**.

**Step 5: New Node - Filters**

1. Do not specify any filters for the demo.
2. Click **Next**.

### Step 6: New Node - Trust Zone

1. Do not specify a trust zone the demo.
2. Click **Finish**

The node is created, and needs to be provisioned.

---

### Provisioning a new node

To provision the new node, perform the following steps:

1. Select the **Deployment** tab from the **Configure** menu bar.
2. The **Provisioning** page is displayed, and `agent1` is listed as not provisioned.
3. Select the `agent1` check box.
4. Click **Provision**. This displays a message when complete: `Successfully provisioned`.
5. Click the **Manage** radio button on the Actional Web interface. You should see `agent1` added to the **Network Overview** screen.

---

# Configuring Artix for Actional Integration

---

## Overview

This section explains how to configure Artix Java (JAX-RPC) services for integration with Actional. It shows some examples from the Artix–Actional integration demos:

```
ArtixInstallDir/cxx_java/samples/advanced/management/monitoring/actional_http_handler
ArtixInstallDir/cxx_java/samples/advanced/management/monitoring/actional_jms_handler
```

---

## Configuring the Artix actional plug-in

Configuring basic Artix actional plug-in includes the following steps:

### Specifying the plug-in name:

```
orb_plugins = ["actional"];
```

### Adding actional to the interceptor chain

You must specify actional interceptor to the request-level interceptor lists. If payload needs to be captured, actional interceptor is also required on message interceptor list. Actional inteceptor can be added to client and server interceptor list depends on which one or both needs to be monitored.

```
# Add actional to the client interceptors chain.
binding:artix:client_request_interceptor_list= "actional";
binding:artix:client_message_interceptor_list= "actional";

# Add actional to server interceptors chain.
binding:artix:server_request_interceptor_list= "actional";
binding:artix:server_message_interceptor_list= "actional";
```

For more details on configuring binding lists and interceptors, see [Artix Configuration Reference](#).

---

## Optimizing your Actional integration

Artix provides the following configuration options to enable you to fine-tune the behavior of the monitoring plug-in.

### Capturing the message payload

You can choose to enable capturing of the message payload (for example, a SOAP message over HTTP). If this option is set to `false`, only the payload size is reported. The default values are:

```
plugins:actional:client:capture_request_payload = "false";  
plugins:actional:client:capture_response_payload = "false";  
plugins:actional:server:capture_request_payload = "false";  
plugins:actional:server:capture_response_payload = "false";
```

---

# Viewing Artix Endpoints in Actional

---

## Overview

When your Artix service endpoints and consumers have been configured for integration with Actional, they can be monitored using the Actional SOA management tools.

For example, when you run the Artix–Actional SOAP over HTTP demo, the **Actional Server Administration Console** displays the agent nodes. Invocations are displayed as arrows flowing to and from the node. For details on how to run this demo, see the `readme.txt` file in the following directory:

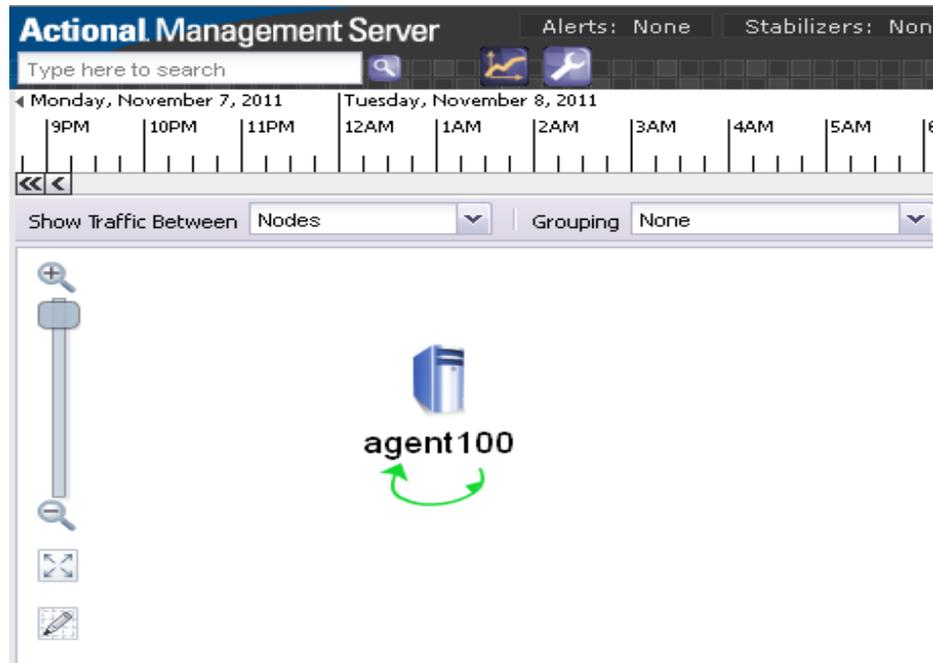
```
ArtixInstallDir/samples/advanced/management/actional/soap_over_http
```

## Network overview

[Figure 16](#) shows a running SOAP over HTTP demo displayed in the **Network Overview** screen of the **Actional Server Administration Console**.

In [Figure 16](#), interactions between the client and server applications are recorded by agent100, which is installed on the machine that runs the demo. This agent reports monitoring data back to the Actional server.

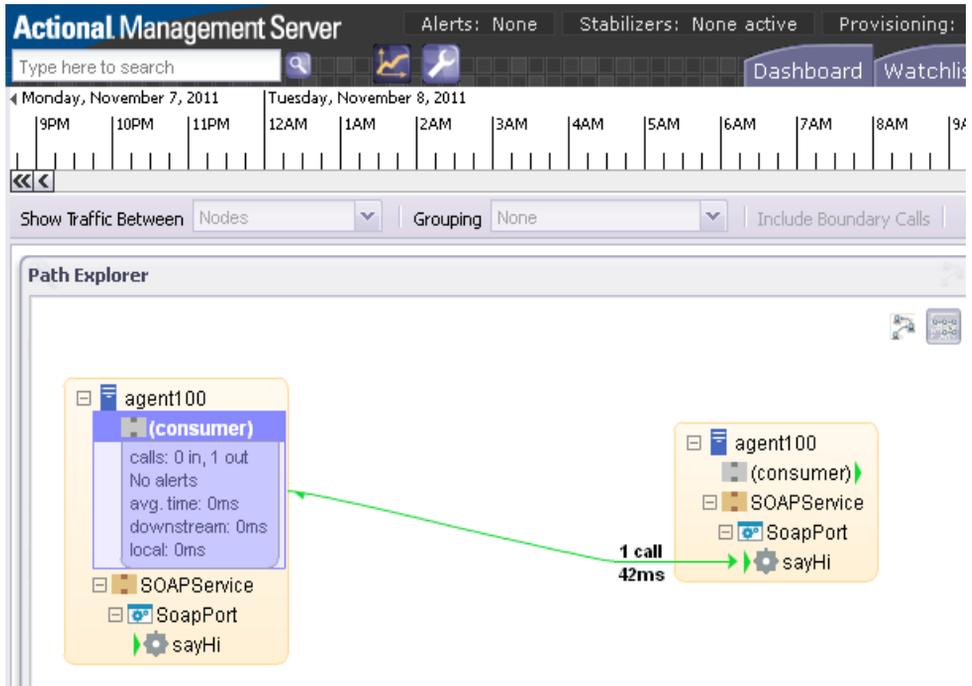
**Figure 16:** *Actional Server Network Overview*



**Path Explorer**

Figure 17 shows the example invocation displayed in the **Path Explorer** screen of the **Actional Server Administration Console**.

**Figure 17:** *Actional Server Path Explorer*

**Further information****Actional**

For information on how to set up and run the Actional server, Actional agent, and Actional Server Administration Console, see the Actional product documentation.

**Artix**

For more information on Artix configuration, see the following:

- [Configuring and Deploying Artix Solutions, C++ Runtime](#)
- [Artix Configuration Reference, C++ Runtime](#)

# Part IV

## AmberPoint

---

### **In this part**

This part contains the following chapters:

<a href="#">Integrating with AmberPoint</a>	page 111
<a href="#">Configuring the Artix AmberPoint Agent</a>	page 123



# Integrating with AmberPoint

*Artix provides support for integration with the AmberPoint SOA management system. This chapter describes two approaches to integrating Artix services with AmberPoint.*

---

**In this chapter**

This chapter includes the following sections:

<a href="#">AmberPoint Proxy Agent</a>	<a href="#">page 112</a>
<a href="#">Artix AmberPoint Agent</a>	<a href="#">page 115</a>

---

# AmberPoint Proxy Agent

---

## Overview

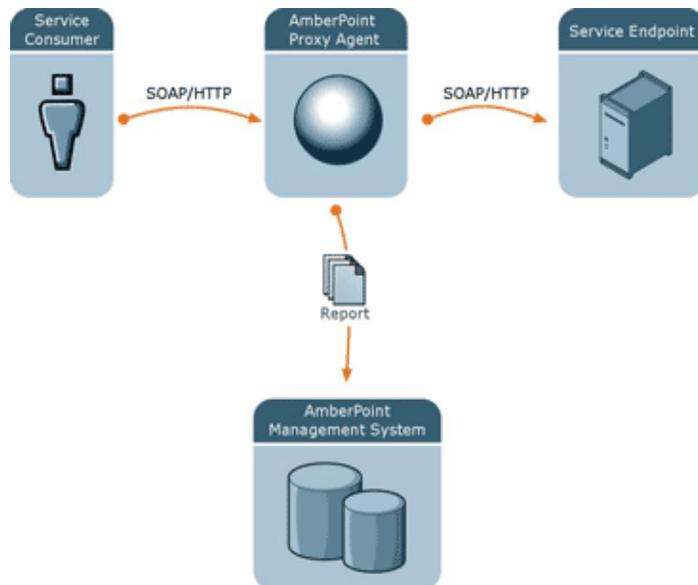
There are two possible approaches to integrating Artix with the AmberPoint SOA management system:

- AmberPoint Proxy Agent
  - Artix AmberPoint Agent
- 

## AmberPoint Proxy Agent architecture

AmberPoint provides the AmberPoint Proxy Agent, which acts as a proxy for Web service endpoints by making the service endpoint WSDL available to the service consumer (client). [Figure 18](#) shows a simple AmberPoint Proxy Agent architecture:

**Figure 18:** *AmberPoint Proxy Agent Integration*



In this architecture, the following restrictions apply:

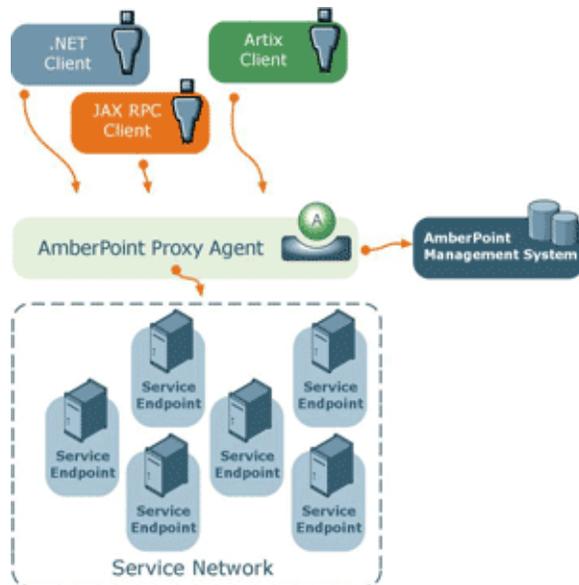
- All messages between the service consumer and service endpoint must be routed through the AmberPoint Proxy Agent.
- All messages must use SOAP over HTTP.
- The service consumer is unaware of the back-end service endpoint, and views its relationship as being with the proxy only.

If you can work within these limits, the AmberPoint monitoring and management features can be used out-of-the box with Artix. However, if you require a more flexible integration (for example, with increased performance and scalability), you should use the Artix AmberPoint Agent.

### AmberPoint Proxy Agent in a service network

Figure 19 shows the AmberPoint Proxy Agent deployed in a service network with multiple service consumers and service endpoints.

**Figure 19:** *AmberPoint Proxy Agent Service Network*



Because all messages are routed through the AmberPoint Proxy Agent, the additional network hops may impact on performance. In addition, the proxy involves the risk of a single point of failure.

If these are important issues for your system, you should use the Artix AmberPoint Agent instead.

---

**Further information**

For information on using the AmberPoint Proxy Agent, see the AmberPoint product documentation.

---

# Artix AmberPoint Agent

---

## Overview

The Artix AmberPoint Agent enables Artix endpoints to be discovered and monitored by AmberPoint. This is the recommended approach to integrating Artix services with AmberPoint, and can be used with Artix services.

The Artix AmberPoint Agent can be deployed with Artix endpoints that use SOAP over HTTP to enable reporting of performance metrics back to AmberPoint. The Artix AmberPoint Agent offers significant benefits over the AmberPoint Proxy Agent. For example, these include increased performance and scalability, dynamic discovery, and the use of callbacks. This section describes the Artix AmberPoint Agent in detail.

---

## Artix AmberPoint Agent architecture

Figure 20 shows how Artix can be integrated with AmberPoint using the Artix AmberPoint Agent.

**Figure 20:** *Artix AmberPoint Agent Integration*



The main components in this architecture are:

- “Artix AmberPoint Agent”
- “Artix interceptor”
- “Artix service endpoints”
- “Service consumers”
- “AmberPoint SOA Management System”
- “AmberPoint Nano Agent API”

**Note:** Integration with the Artix AmberPoint Agent currently applies to SOAP over HTTP, and services that have one endpoint only.

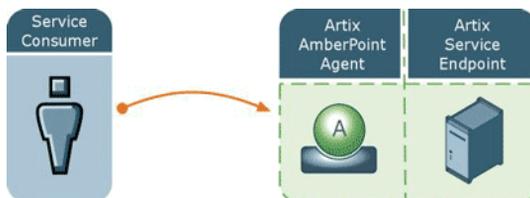
## Artix AmberPoint Agent

An Artix AmberPoint Agent consists of components developed by Progress and AmberPoint (the Artix interceptor, and the AmberPoint Nano Agent API). You can deploy multiple agents into your SOA network to capture data for the AmberPoint management system. Artix AmberPoint Agents gather performance data for all Artix endpoint types, as well as normal Web service endpoints.

### Deployment modes

Artix AmberPoint Agents can be deployed in different ways in your system, for example:

- *Embedded in Artix consumers intercepting traffic.* This is suitable if Artix is deployed on the client side only, and the service endpoints do not support AmberPoint. This requires configuration for the consumer only.
- *Embedded in Artix service endpoints intercepting traffic.* This is suitable if Artix is used to implement the service endpoint. This works even when the consumers are third party products. This requires configuration for the service endpoint only. This is the most common and recommended approach, as shown in [Figure 21](#).
- *Deployed as standalone Artix intermediaries (proxies) on your service network.* This option is suitable if you do not want touch your existing system and you do not want to update your endpoints or consumers. This approach is also necessary if Artix is not deployed at either the consumer or service endpoints.

**Figure 21:** *Artix AmberPoint Agent Embedded in Service Endpoint*

---

**Artix interceptor**

An Artix interceptor is deployed on the dispatch path of all messages exchanged between Artix service endpoints and consumers. It may be deployed in the same process as the consumer and/or the endpoint, or as an intermediary between the consumer and service.

The Artix interceptor captures all data in the dispatch path. The Artix interceptor then reports performance metrics using the AmberPoint nano agent API.

---

**Artix service endpoints**

An Artix service endpoint is a service built using Artix, and described using WSDL. The endpoint can be implemented using C++. However, its main characteristic is that it can be described in WSDL, and classified as a service, which can therefore be consumed. The Artix AmberPoint Agent provides a WSDL contract describing the endpoint that is being monitored.

---

**Service consumers**

Service consumers are clients that consume service endpoints by exchanging messages based on the service interface. Consumers can be built using Artix, or any product that supports the technology used by the endpoint. For example, a pure CORBA client could be a consumer for a CORBA endpoint. A .NET client could be a consumer for an Artix SOAP endpoint.

---

**AmberPoint SOA Management System**

In this document, AmberPoint is the general term used to describe the system in which all performance metrics are stored and viewed. For the purposes of this document, all interactions are made using the AmberPoint Nano Agent API, and the AmberPoint graphical tools are used to view the Artix data. This simplifies the architecture of AmberPoint for the sake of this discussion.

---

**AmberPoint Nano Agent API**

The AmberPoint Nano Agent API is a Java public API provided by AmberPoint that enables customers to monitor their endpoints. This is the API that Artix uses to notify AmberPoint of the existence of the service endpoint. Artix also uses the AmberPoint nano agent API at runtime to report performance metrics about a previously registered endpoint.

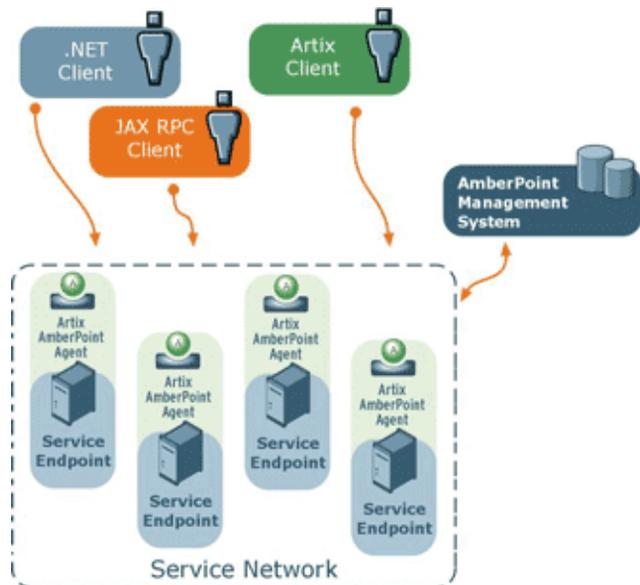
The AmberPoint Nano Agent API enables the Artix interceptor to do the following:

- Allow dynamic discovery of new Artix endpoints without manual registration of the endpoints by the user. This registration process assumes that the Artix interceptor has the required configuration for the nano agent to contact AmberPoint. When the Artix AmberPoint Agent becomes active, it uses the Nano Agent API to register a new endpoint.
- Allow periodic reporting of messages using the Artix interceptor. These reports contain performance data about the endpoint and the messages being exchanged.

## Artix AmberPoint Agent in a service network

Figure 22 shows the Artix AmberPoint Agent deployed in a service network with multiple service consumers and service endpoints.

**Figure 22:** *Artix AmberPoint Agent Service Network*



This loosely-coupled architecture has the following benefits:

- Because the Artix AmberPoint Agent is collocated and embedded in the service endpoint, there are no additional network hops, so performance is maximized.
- Unlike with the AmberPoint Proxy Agent, there is no risk of a single point of failure, so reliability and scalability are also improved.
- An Artix AmberPoint Agent can be embedded into an Artix router. This enables it to dynamically discover and monitor the Artix service endpoints and consumers that the router creates and manages.
- Because the client is aware of the back-end service endpoint, the use of callbacks is supported.

---

**Supported AmberPoint features**

The Artix AmberPoint Agent enables the use of the following AmberPoint features:

- Dynamic discovery of Artix clients and services using SOAP over HTTP.
- Monitoring of Artix client and service invocations, and reporting them back to AmberPoint.
- Mapping Qualities of Service (QoS) to customer Service Level Agreements (SLAs).
- Monitoring of Artix invocation flow dependencies, which enables AmberPoint to draw Web service dependency diagrams.
- Centralized logging and performance statistics.

---

**Further information**

For detailed information on using AmberPoint features, see the AmberPoint product documentation.



# Configuring the Artix AmberPoint Agent

*This chapter explains how to set up integration with the Artix AmberPoint Agent, and shows examples from the Artix AmberPoint integration demos.*

---

**In this chapter**

This chapter includes the following sections:

<a href="#">Installing AmberPoint</a>	page 124
<a href="#">Configuring AmberPoint for Artix Integration</a>	page 125
<a href="#">Configuring Artix C++ Services for AmberPoint Integration</a>	page 128

---

# Installing AmberPoint

---

## Overview

The Artix ESB C++ runtime supports integration with version 5.1 of the AmberPoint SOA management system. This section explains how to install AmberPoint to enable integration with the Artix AmberPoint Agent.

---

## Installation steps

When installing the AmberPoint runtime, perform the following steps:

1. In the AmberPoint installation wizard, choose a suitable HTTP port number for the J2EE application server in which the AmberPoint server will be deployed (for example, 9090).
2. AmberPoint comes bundled with Tomcat application server, so for the demo purposes, choose to install Tomcat.
3. Select **Deploy AmberPoint into the container**.
4. Select **Install a Java VM specifically for this application**.
5. Select **Deploy a new sphere with the SOA Management System**. This deploys the persistence runtime into the J2EE application server, and configures it to use the embedded Tomcat HSQL relational database management system.
6. You can also install AmberPoint sample Web services, but these are not required.
7. Provide a user name and password with administrative privileges (for example, `admin/admin`).
8. When installation is complete, copy the AmberPoint Nano Agent Server into the deployment directory of the application server. For example, for Tomcat, use the following command:

```
copy AP_InstallDir/add_ons/socket_converter/apssocketconverter.war  
AP_InstallDir/server/webapps
```

If you are not using Tomcat, use the vendor's visual tools to deploy `apssocketconverter.war` into the application server.

---

# Configuring AmberPoint for Artix Integration

---

## Overview

This section explains how to configure the AmberPoint SOA management system for integration with Artix.

---

## Starting the AmberPoint Server

When you have completed the AmberPoint installation steps, run the AmberPoint server using Window's **Start** menu.

Alternatively, execute the following script:

**Windows**      `AP_InstallDir\server\bin\startup.bat`

**UNIX**            `AP_InstallDir/server/bin/startup.sh`

You can see how your application server starts up and deploys the AmberPoint server in the log files in the `AP_InstallDir/server/logs` directory.

---

## Configuring the AmberPoint Nano Agent Sever

When the application server has started and deployed all the AmberPoint `.war` files, perform the following steps:

1. Open a web browser and specify the following URL:  
`http://hostname:port/apasc/`
2. Login using the admin user name and password that you provided when installing AmberPoint.
3. When logged in, click **Network|Infrastructure** in the tabbed menu. This displays a list of registered **Deployments** with this application server's container.
4. Ensure that one of the deployed items is named `apsocketconverter` and has a green button beside it This indicates that the AmberPoint Nano Agent Server has been successfully deployed and is ready to be configured.

5. In the left pane, click the **Register** button.  
From the drop-down menu, select **Message Source****Simple Message Source**: This displays the **Register Message Source** form.

6. In the **Register Message Source** form, enter the following:

<b>Name</b>	Artix Message Source
<b>Type of Message Source</b>	File
<b>Start At</b>	At present
<b>Location</b>	<i>AmberPointInstallDir</i> \server\amberpoint\ apsocketconverter\logdir

The source **Name** can be any string value. The **Location** specifies the location of the log file for incoming messages. The default **Criteria for this policy** applies this message source to all active services that this AmberPoint system is aware of.

7. Without modifying the **Criteria for this policy**, click **Preview Services** to see which services this message Source applies to. If you have no services currently registered, only one service named **MonitorEnabler** is displayed.
8. Click the **Go** button at the top left of the screen, and wait until the **Policy Status** is *Applied*.
9. Return to a command window to build an Artix AmberPoint demo (see [“Configuring Artix C++ Services for AmberPoint Integration”](#) on page 128).

**Configuring the AmberPoint port**

If the default AmberPoint Nano Agent Server port (33333) does not suit your setup, change the following attributes to the new port number:

- `messageLogWriter logLocation` in your Artix `apobserver.configuration` file
- `messageLogReader logLocation` in:

```
AP_InstallDir/server/webapps/apsocketconverter.war@/WEB-INF/
application/resources/readerConfig.xml
```

Whenever you update values in the Artix `apobserver.configuration` file, you must restart the services already being monitored by the Artix AmberPoint Agent for the changes to take effect.

If you update the Nano Agent Server port, you may need to restart the application server for changes to take effect (except for those servers that support hot deployment).

For example, these settings appear as follows in the Artix `apobserver.configuration` file:

```
...
<ap:messageLogWriter
  logWriterImplClass="com.amberpoint.msglog.socketimpl.SocketLogWriter"
  logName="{hostname}" <!-- default = localhost -->
  logLocation="{port}" <!-- default = 33333 -->
  syncEverySoManyEntries="50">
</ap:messageLogWriter>
...
<ap:hostMapper algorithm="asSent" urlProperty="ap:requestURL"/>
...
<ap:hostMapper algorithm="asSent" urlProperty="ap:wSDLUrl"/>
...
```

---

# Configuring Artix C++ Services for AmberPoint Integration

---

## Overview

This section explains how to configure Artix C++ and JAX-RPC services to support the Artix AmberPoint Agent. It describes Artix AmberPoint demo configuration settings in detail. However, if your AmberPoint installation and demo run on the same host, you do not need to make any configuration changes to run the demo. If you wish to run the demo now, skip this section, and see the `readme.txt` in the following directory:

```
ArtixInstallDir/samples/integration/amberpoint
```

This `amberpoint` demo is based on the `.../samples/routing/content_based` demo, with some modifications to enable Artix and AmberPoint integration.

---

## Configuring the AmberPoint Nano Agent plug-in

You must enable the AmberPoint Nano Agent plug-in for the Artix runtime. For example, the configuration scope in which the demo servers run includes an Artix plug-in named `ap_nano_agent`. This is loaded into the Artix runtime, and enables discovery and monitoring by AmberPoint of services and consumers running inside Artix processes.

```
demos {
  content_based {
    orb_plugins = ["xmlfile_log_stream", "soap", "at_http", "ap_nano_agent"];
    ...
  }
  ...
}
```

In this demo, there are three server instances, each exposing the same interface but running under different service and endpoint name pairs. These are as follows:

```
{TargetService1, TargetPort1}
{TargetService2, TargetPort2}
{TargetService3, TargetPort3}
```

## Configuring the Artix router

To enable router support, you must also add the AmberPoint Nano Agent plug-in to the router's configuration. For example, the demo configuration scope in which the Artix router runs includes additional configuration for the Artix routing plug-in. Its `orb_plugins` list includes the `ap_nano_agent` plug-in, which enables the router's endpoints and consumers to be discovered and monitored by AmberPoint.

```

demos {
  content_based {
    ...
    router {
      orb_plugins = [ "xmlfile_log_stream", "ap_nano_agent", "routing" ];
      plugins:routing:use_pass_through="false";
      ...
    }
  }
}

```

The `ap_nano_agent` plug-in must precede the `routing` plug-in. This is because the Artix AmberPoint Agent must register itself in the interceptor chain before the routing plug-in instantiates and activates the services that it manages.

Setting `plugins:routing:use_pass_through` to `false` disables passing data through the router without parsing. The `ap_nano_agent` plug-in requires that the underlying payload is parsed in the Artix type format.

## Configuring the consumer hostname

`plugins:ap_nano_agent:hostname_address:publish_hostname` specifies the form in which the Artix AmberPoint Agent resolves the host address that an Artix service consumer (proxy) runs on. This variable takes the following values:

<code>unqualified</code>	The host name in short form, without the domain name ( <i>hostname</i> ).
<code>ipaddress</code>	The host name in the form of an IP address (for example, 123.4.56.789). This is the default.
<code>canonical</code>	The host name takes a fully qualified form ( <i>hostname.domainname</i> ).
<code>true</code>	same as <code>unqualified</code>
<code>false</code>	same as <code>ipaddress</code>

`plugins:ap_nano_agent:hostname_address:local_hostname` is an arbitrary string used as the client hostname instead of trying to resolve it using the underlying IP runtime. This is undefined by default.

To report the correct service consumer address invoking to an Artix service monitored by this agent, specify the following setting in the client and server configuration scope:

```
plugins:bus:register_client_context="true";
```

### Configuring the service hostname

The server-side host name resolution is driven by the specific transport. Because the HTTP transport is the only one currently supported the following variables must be configured:

- `policies:soap:server_address_mode_policy:publish_hostname`
- `policies:at_http:server_address_mode_policy:publish_hostname`

Possible values are the same as those for

`plugins:ap_nano_agent:hostname_address:publish_hostname`.

These variables specify the format that a service endpoint address is published to service consumers. AmberPoint discovers Artix services by consuming a published WSDL contract. It correlates the address in the WSDL with the inflow of log messages that describe operations invoked on an endpoint. This means that you must synchronize these configuration values with the configuration values of the AmberPoint Client Nano Agent.

### Configuring the AmberPoint hostname

The default Artix hostname resolution setting is `ipaddress`, which is the same as that for the configuration of AmberPoint Client Nano Agent. However, if you change the Artix hostname resolution, you must also update the AmberPoint Client Nano Agent configuration file. For example:

```
ArtixInstallDir/etc/amberpoint/5.1/nanoagent/conf/apobserver.configuration
```

To update the hostname resolutions setting, open the file in a text editor and find the two occurrences of the `hostMapper algorithm` attribute.

You must update the value of `hostMapper algorithm` attribute if you change the value of

`policies:soap:server_address_mode_policy:publish_hostname` and `policies:at_http:server_address_mode_policy:publish_hostname` configuration variables.

The equivalent AmberPoint values are as follows:

Artix <code>publish_hostname</code> variable	AmberPoint <code>hostMapper</code> algorithm
<code>ipaddress</code>	<code>useIpAddress</code> or <code>asSent</code>
<code>canonical</code>	<code>useFQN</code> or <code>asSent</code>
<code>unqualified</code>	<code>asSent</code>

To avoid updating the AmberPoint Nano Agent Client configuration each time you change the Artix configuration, simply use `hostMapper algorithm="asSent"`.

If you are running your Artix services and the AmberPoint Nano Agent Server on different machines, you must also update the `messageLogWriter logName` attribute to point the host name or IP address where the Nano Agent Server is running.

### Configuring the AmberPoint port

If the default AmberPoint Nano Agent Server port (33333) does not suit your setup, you can update your AmberPoint configuration file to the new port number. For more details, see [“Configuring the AmberPoint port” on page 127](#).

### Viewing Artix services in AmberPoint

When you run the demo, and start the Artix router and servers, and make client invocations to the router, these calls are in turn forwarded on to the servers.

#### AmberPoint dependency diagrams

While the demo is running, in the AmberPoint GUI, select the **Network|Services|Dependencies** screen. AmberPoint tracks the call flow, as it happens, between Artix services with the Artix AmberPoint Agent in their runtime. The dependency flow diagram is a directed graph, and can be of any complexity. For example, a client makes three calls to the source service implemented by the router. Each call is routed to the intended destination service, defined by the routing rules. Each `TargetService` receives a single call out of the three made. And each dependency tracking is shown in relation to the service selected in the **Selector** list, which is referred as a primary service. You can manually create dependencies between services using the AmberPoint tools if so desired. See the AmberPoint user documentation for details on what you can do with dependency diagrams (for example, using the **Network|Services|Dependencies** screen).

**AmberPoint performance diagrams**

You can use the AmberPoint **Performance|Activity** screen to view performance statistics. See the AmberPoint user documentation for details on what you can do with performance statistics.

**AmberPoint logging policies**

You can collect call logs by adding an AmberPoint logging policy using the **Exceptions|Services** screen. To add an AmberPoint logging policy, click the **Add Logging Policy** button at the top of the screen. This displays the **Add Policy** form. Use this form to specify a meaningful name, and tune its parameters to your needs. If you wish to log messages for all available services, edit the policy rules at the bottom of this form.

When the log policy is created, you must wait until it is applied, like when you created a **Message Source** (see “[Configuring the AmberPoint Nano Agent Sever](#)” on page 125). After the log policy has been applied and turns green, send some more traffic using the demo. You can then watch the **Message Log** using the **Exceptions|Services|Message Log** tab.

**Further information**

There are many other AmberPoint features that you can use with Artix. For example, when AmberPoint has captured the Artix traffic, you can use its runtime to define customers and their SLAs, and map these SLAs to the services in the network. You can also create reactions (alerts) if an SLA violation has occurred and so on. See the AmberPoint user documentation for more details.

**Artix AmberPoint demo**

For more details on the Artix AmberPoint integration demo, see:

```
ArtixInstallDir\samples\integration\amberpoint\README.txt
```

**Artix C++ configuration**

- [Configuring and Deploying Artix Solutions, C++ Runtime](#)
- [Artix Configuration Reference, C++ Runtime](#)

# Part V

## BMC Patrol

---

### **In this part**

This part contains the following chapters:

<a href="#">Integrating with BMC Patrol™</a>	<a href="#">page 135</a>
<a href="#">Configuring Artix for BMC Patrol</a>	<a href="#">page 143</a>
<a href="#">Using the Artix BMC Patrol Integration</a>	<a href="#">page 147</a>
<a href="#">Extending to a BMC Production Environment</a>	<a href="#">page 157</a>



# Integrating with BMC Patrol™

*This chapter introduces Artix ESB's integration with the BMC Patrol™ Enterprise Management System. It describes the requirements and main components of this integration.*

---

**In this chapter**

This chapter contains the following sections:

<a href="#">Introduction</a>	<a href="#">page 136</a>
<a href="#">The Artix BMC Patrol Integration</a>	<a href="#">page 140</a>

---

# Introduction

---

## Overview

Artix ESB supports integration with Enterprise Management Systems such as BMC Patrol. This section includes the following topics:

- [“The application life cycle”](#)
  - [“Enterprise Management Systems”](#)
  - [“Artix BMC Patrol features”](#)
  - [“How it works”](#)
- 

## The application life cycle

Most enterprise applications go through a rigorous development and testing process before they are put into production. When applications are in production, developers rarely expect to manage those applications. They usually move on to new projects, while the day-to-day running of the applications is managed by a production team. In some cases, the applications are deployed in a data center that is owned by a third party, and the team that monitors the applications belongs to a different organization.

---

## Enterprise Management Systems

Different organizations have different approaches to managing their production environment, but most will have at least one *Enterprise Management System* (EMS).

For example, the main Enterprise Management Systems include BMC Patrol™ and IBM Tivoli™. These systems are popular because they give a top-to-bottom view of every part of the IT infrastructure.

This means that if an application fails because the `/tmp` directory fills up on a particular host, for example, the disk space is reported as the fundamental reason for the failure. The various application errors that arise are interpreted as symptoms of the underlying problem with disk space. This is much better than being swamped by an event storm of higher-level failures that all originate from the same underlying problem. This is the fundamental strength of integrated management.

---

## Artix BMC Patrol features

The Artix ESB BMC Patrol integration performs the following key enterprise management tasks:

- Posting an event when a server crashes. This enables programmed recovery actions to be taken.
- Tracking key server metrics (for example, server response times). Alarms are triggered when these go out of bounds.

The server metrics tracked by the Artix BMC Patrol integration include the number of invocations received, and the average, maximum and minimum response times. The Artix BMC Patrol integration also enables you to track these metrics for individual operations. Events can be generated when any of these parameters go out of bounds. You can also perform a number of actions on servers including stopping, starting and restarting.

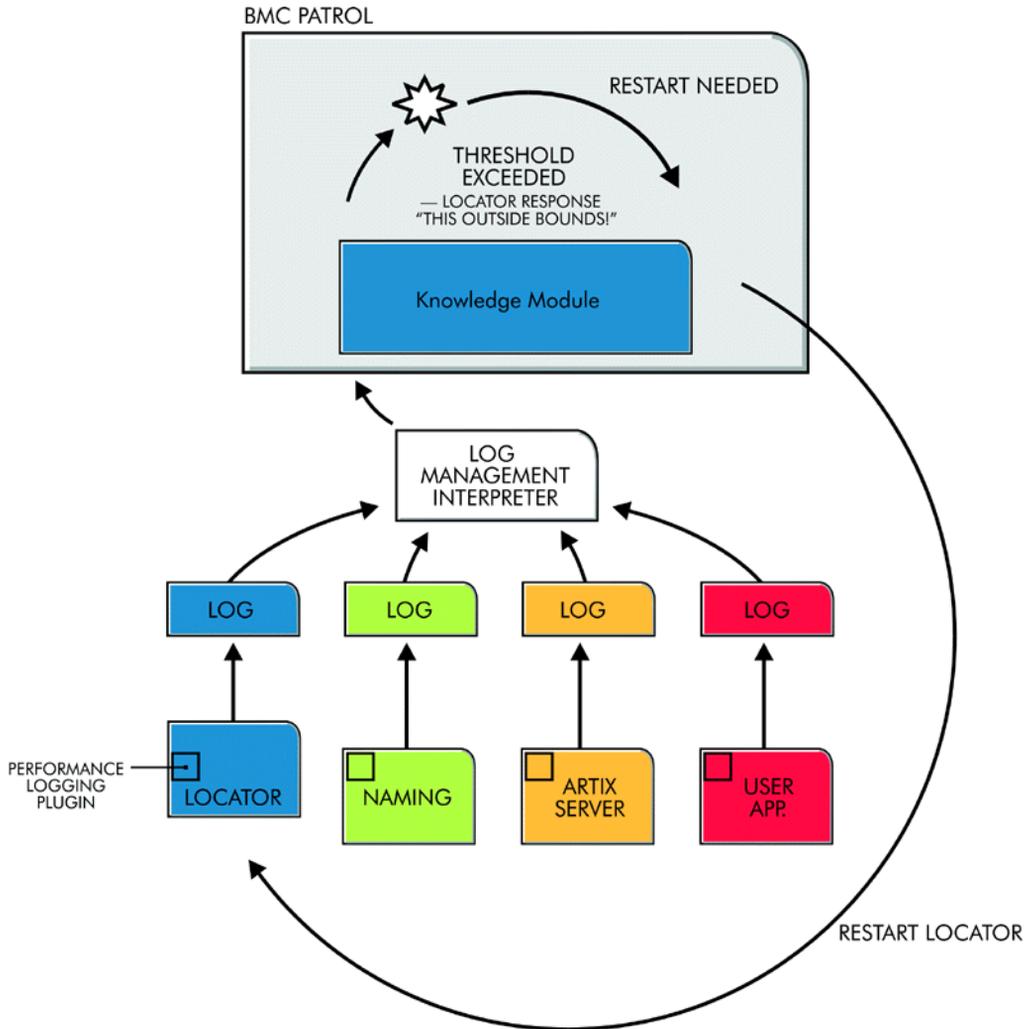
---

## How it works

In the BMC Patrol integration, key server metrics are logged by the Artix performance logging plug-ins. Log file interpreting utilities are then used to analyze the logged data.

Artix also provides Knowledge Modules, which conform to standard BMC Knowledge Module design and operation. These modules tell the BMC Patrol console how to interpret the logging data received from the Artix services. [Figure 23 on page 138](#) shows a simplified view of how the Knowledge Modules work. In this example, an alarm is triggered in the BMC Patrol console when a locator becomes unresponsive, and this results in an action to restart the locator.

**Figure 23:** Overview of the Artix BMC Patrol Integration



The performance logging plug-ins collect data relating to server response times and log it periodically in the performance logs. The Knowledge Module executes parameter collection periodically on each host, using the log file interpreter running on each host to collect and summarize the logged data.

The Knowledge Module compares the response times and other values against the defined alarm ranges, and issues an alarm event if a threshold has been breached. These events can be analyzed and appropriate action taken automatically (for example, restart a server). Alternatively, the user can intervene manually and execute a BMC Patrol menu command to stop, start or restart the offending server.

---

# The Artix BMC Patrol Integration

---

## Overview

This section describes the requirements and main components of the Artix BMC Patrol integration. It includes the following topics:

- [“BMC Patrol requirements”](#)
  - [“Main components”](#)
  - [“Example metrics”](#)
  - [“Further information”](#)
- 

## BMC Patrol requirements

To use the Artix BMC Patrol integration, you must have BMC Patrol 3.4 or higher. The BMC Patrol integration is compatible with the BMC Patrol 7 Central Console.

---

## Main components

The BMC Patrol integration consists of the following Knowledge Modules (KM):

- `IONA_SERVERPROVIDER`
- `IONA_OPERATIONPROVIDER`

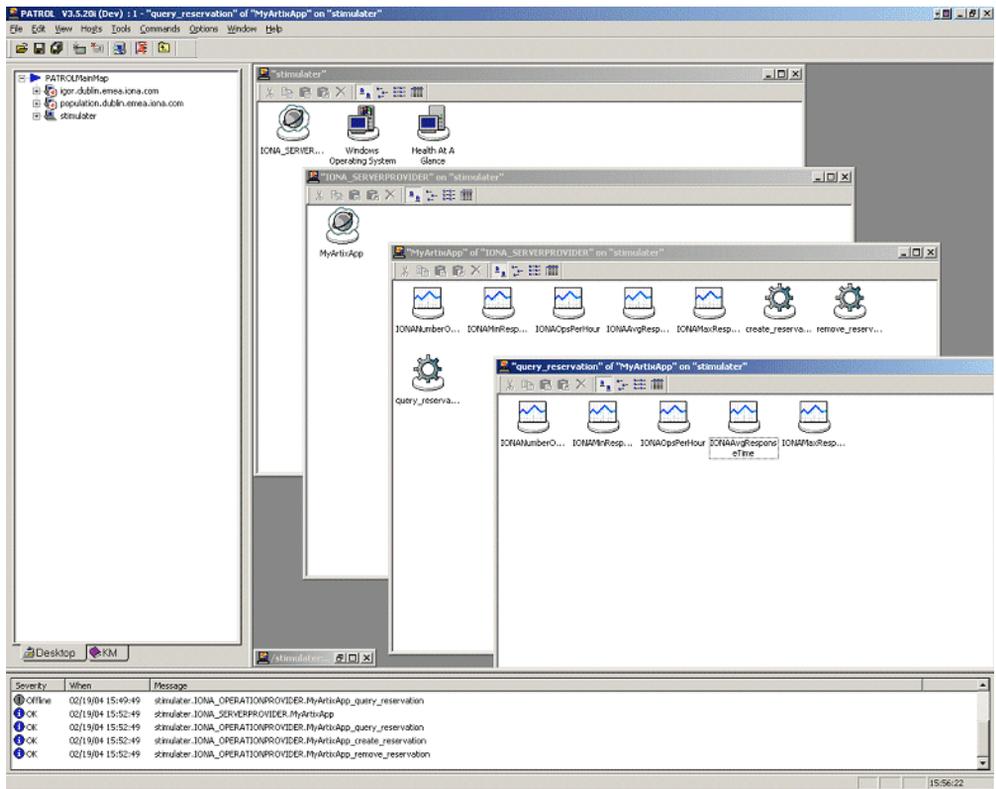
`IONA_SERVERPROVIDER.km` tracks key metrics associated with your Artix servers on a particular host. It also enables servers to be started, stopped, or restarted, if suitably configured.

`IONA_OPERATIONPROVIDER.km` tracks key metrics associated with individual operations on each server.

Example metrics

Figure 24 shows an example of the IONA\_SERVERPROVIDER Knowledge Module displayed in BMC Patrol. The window in focus shows the IONA performance metrics that are available for an operation named `query_reservation`, running on a machine named `stimulator`.

Figure 24: IONA Server Running in BMC Patrol



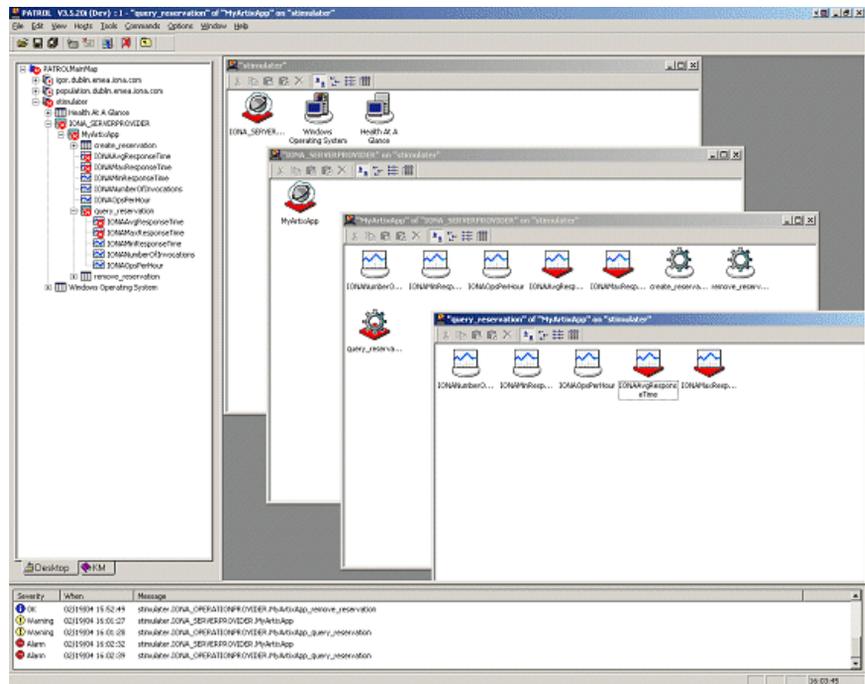
The IONA server performance metrics include the following:

- IONAAvgResponseTime
- IONAMaxResponseTime
- IONAMinResponseTime
- IONANumInvocations
- IONAOpsPerHour

For more details, see [“Using the Artix Knowledge Module” on page 150](#).

Figure 25 shows alarms for server metrics, for example, IONAAvgResponseTime. This measures the average response time of all operations on this server during the last collection cycle.

**Figure 25:** BMC Patrol Displaying Alarms



### Further information

For a detailed description of Knowledge Modules, see your BMC Patrol documentation.

# Configuring Artix for BMC Patrol

*This chapter explains the steps that you need to perform in your IONA product to configure integration with BMC Patrol.*

---

**In this chapter**

This chapter contains the following sections:

<a href="#">Setting up your Artix Environment</a>
---

page 144
----------

---

# Setting up your Artix Environment

---

## Overview

The best way to learn how to use the BMC Patrol integration is to start with a host that has both BMC Patrol and Artix installed. This section explains how to make your Artix servers visible to BMC Patrol. It includes the following topics:

- “EMS configuration files”
  - “Creating a servers.conf file”
  - “Creating a server\_commands.txt file”
  - “Further information”
- 

## EMS configuration files

You need to create two text files that are used to configure the BMC Patrol integration:

- `servers.conf`
- `server_commands.txt`

These files are used to track your Artix applications in BMC Patrol. You will find starting point files in the `IONA_km.zip` located in the following directory of your Artix installation:

```
ArtixInstallDir\management\BMC\IONA_km.zip
```

When you unzip, the starting point files are located in the `lib/iona/conf` directory.

---

## Creating a servers.conf file

The `servers.conf` file is used to instruct BMC Patrol to track your Artix servers. It contains the locations of performance log files for specified applications. Each entry must take the following format:

```
my_application, 1, /path/to/myproject/log/myapplication_perf.log
```

This example entry instructs BMC Patrol to track the `myapplication` server, and reads performance data from the following log file:

```
/path/to/myproject/log/myapplication_perf.log
```

You must add entries for the performance log file of each Artix server on this host that you wish BMC Patrol to track. BMC Patrol uses the `servers.conf` file to locate these log files, and then scans the logs for information about the server's key performance indicators.

The following example is taken from the Artix ESB for Java product sample application for BMC Patrol integration:

```
management-bmc-patrol-demo-server,1,%ARTIX_HOME%\java\samples\advanced
\management\bmc-patrol\BMCCounterServer.log
management-bmc-patrol-demo-client,1,%ARTIX_HOME%\java\samples\advanced
\management\bmc-patrol\BMCCounterClient.log
```

### Creating a `server_commands.txt` file

The `server_commands.txt` file is used to instruct BMC Patrol how to start, stop, and restart your Artix servers. It contains the locations of the relevant scripts for specified servers. Each entry must take the following format:

```
myapplication,start=/path/to/myproject/bin/start_myapplication.sh
myapplication,stop=/path/to/myproject/bin/stop_myapplication.sh
myapplication,restart=/path/to/myproject/bin/restart_myapplication.sh
```

In this example, each entry specifies a script that can be used to stop, start, or restart the `myapplication` server. When BMC Patrol receives an instruction to start `myapplication`, it looks up the `server_commands.txt` file, and executes the script specified in the appropriate entry.

You must add entries that specify the relevant scripts for each server on this host that you wish BMC Patrol to track.

---

**Copy the EMS files to your BMC installation**

When you have added content to your `servers.conf` and `server_commands.txt` files, copy these files into your BMC installation, for example:

```
$PATROL_HOME/lib/iona/conf
```

This enables tracking of your Artix server applications in BMC Patrol.

---

**Further information**

For details of how to configure your Artix servers to use performance logging, see [“Configuring an Artix Production Environment” on page 158](#).

For a complete explanation of configuring performance logging, see [Configuring and Deploying Artix Solutions, C++ Runtime](#).

# Using the Artix BMC Patrol Integration

*This chapter explains the steps that you must perform in your BMC Patrol environment to monitor Artix applications. It also describes the Artix Knowledge Module and how to use it to monitor servers and operations. It assumes that you already have a good working knowledge of BMC Patrol.*

---

**In this chapter**

This chapter contains the following sections:

<a href="#">Setting up your BMC Patrol Environment</a>	<a href="#">page 148</a>
<a href="#">Using the Artix Knowledge Module</a>	<a href="#">page 150</a>

---

# Setting up your BMC Patrol Environment

---

## Overview

To enable monitoring of the Artix servers on your host, you must first perform the following steps in your BMC Patrol environment:

1. [“Install the Knowledge Module”](#)
  2. [“Set up your Java environment”](#)
  3. [“Set up your EMS configuration files”](#)
  4. [“View your servers in the BMC Console”](#)
- 

## Install the Knowledge Module

The Artix BMC Patrol Knowledge Module is shipped in two formats:

**Windows**     *ArtixInstallDir\management\BMC\IONA\_km.zip*

**UNIX**         *ArtixInstallDir/management/BMC/IONA\_km.tgz*

To install the Artix Knowledge Module:

### Windows

Use WinZip to unzip `IONA_km.zip`. Extract this file into your `%PATROL_HOME%` directory.

If this is successful, the following directory is created:

```
%PATROL_HOME%\lib\iona
```

### UNIX

Copy the `IONA_km.tgz` file into `$PATROL_HOME`, and enter the following commands:

```
$ cd $PATROL_HOME
$ gunzip IONA_km.tgz
$ tar xvf IONA_km.tar
```

---

## Set up your Java environment

The Artix Knowledge Module requires a Java Runtime Environment (JRE). If your BMC Patrol installation already has a `$PATROL_HOME/lib/jre` directory, it should work straightaway. If not, you must setup a JRE (version 1.3.1 or later) on your machine as follows:

1. Copy the `jre` directory from your Java installation into `$PATROL_HOME/lib`. You should now have a directory structure that includes `$PATROL_HOME/lib/jre`.
2. Confirm that you can run `$PATROL_HOME/lib/jre/bin/java`.

---

## Set up your EMS configuration files

In [Chapter 12](#), you generated the following EMS configuration files:

- `servers.conf`
- `server_commands.txt`

Copy these generated files to `$PATROL_HOME/lib/iona/conf`.

---

## View your servers in the BMC Console

To view your servers in the **BMC Console**, and check that your setup is correct:

1. Start your **BMC Console** and connect to the **BMC Patrol Agent** on the host where you have installed the IONA Knowledge Module.
2. In the **Load KMs** dialog, open the `$PATROL_HOME/lib/knowledge` directory, and select the `IONA_SERVER.km1` file. This will load the `IONA_SERVERPROVIDER.km` and `IONA_OPERATIONPROVIDER.km` Knowledge Modules.
3. In your **Main Map**, the list of servers that were configured in the `servers.conf` file should be displayed. If they are not currently running, they are shown as offline.

You are now ready to manage these servers using BMC Patrol.

# Using the Artix Knowledge Module

## Overview

This section describes the Artix Knowledge Module and explains how to use it to monitor servers and operations. It includes the following topics:

- “Server Provider parameters”
- “Monitoring servers”
- “Monitoring operations”
- “Operation parameters”
- “Starting, stopping and restarting servers”
- “Troubleshooting”

## Server Provider parameters

The `IONA_SERVERPROVIDER` class represents instances of Artix server or client applications. The parameters exposed in the Knowledge Module are shown in [Table 9](#).

**Table 9:** *Artix Server Provider Parameters*

Parameter Name	Default Warning	Default Alarm	Description
IONAAvgResponseTime	1000–5000	> 5000	The average response time (in milliseconds) of all operations on this server during the last collection cycle.
IONAMaxResponseTime	1000–5000	> 5000	The slowest operation response time (in milliseconds) during the last collection cycle.
IONAMinResponseTime	1000–5000	> 5000	The quickest operation response time (in milliseconds) during the last collection cycle.
IONANumInvocations	10000–100000	> 100000	The number of invocations received during the last collection period.
IONAOpsPerHour	1000000–10000000	> 10000000	The throughput (in Operations Per Hour) based on the rate calculated from the last collection cycle.

---

## Monitoring servers

You can use the parameters shown in [Table 9](#) to monitor the load and response times of your Artix servers.

The Default Alarm ranges can be overridden on any particular instance, or on all instances, using the BMC Patrol 7 Central console. You can do this as follows:

1. In the **PATROL Central** console's **Main Map**, right click on the selected parameter and choose the **Properties** menu item.
2. In the **Properties** pane, select the **Customization** tab.
3. In the **Properties** drop-down list, select ranges.
4. You can customize the alarm ranges for this parameter on this instance. If you want to apply the customization to all instances, select the **Override All Instances** checkbox.

**Note:** The `IONANumInvocations` parameter is a raw, non-normalized metric and can be subject to sampling errors. To minimize this, keep the performance logging period relatively short, compared to the poll time for the parameter collector.

---

## Monitoring operations

In the same way that you can monitor the overall performance of your servers and clients, you can also monitor the performance of individual operations. In Artix, an operation relates to a WSDL operation defined on a port.

In many cases, the most important metrics relate to the execution of particular operations. For example, it could be that the `make_reservation()`, `query_reservation()` calls are the operations that you are particularly interested in measuring. This means updating your `servers.conf` file as follows:

```
mydomain_myserver,1,/var/mydomain/logs/myserver_perf.log,[make_reservation,query_reservation]
```

In this example, the addition of the bold text enables the `make_reservation` and `query_reservation` operations to be tracked by BMC Patrol.

**Operation parameters**

**Table 10** shows the Artix parameters that are tracked for each operation instance:

**Table 10:** *Artix Operation Provider Parameters*

Parameter Name	Default Warning	Default Alarm	Description
IONAAvgResponseTime	1000–5000	> 5000	The average response time (in milliseconds) for this operation on this server during the last collection cycle.
IONAMaxResponseTime	1000–5000	> 5000	The slowest invocation of this operation (in milliseconds) during the last collection cycle.
IONAMinResponseTime	1000–5000	> 5000	The quickest invocation (in milliseconds) during the last collection cycle.
IONANumInvocations	10000–100000	> 100000	The number of invocations of this operation received during the last collection period.
IONAOpsPerHour	1000000–100000000	> 10000000	The number of operations invoked in a one hour period based on the rate calculated from the last collection cycle.

Figure 26 shows BMC Patrol graphing the value of the `IONAAvgResponseTime` parameter on a `query_reservation` operation call.

Figure 26: Graphing for `IONAAvgResponseTime`

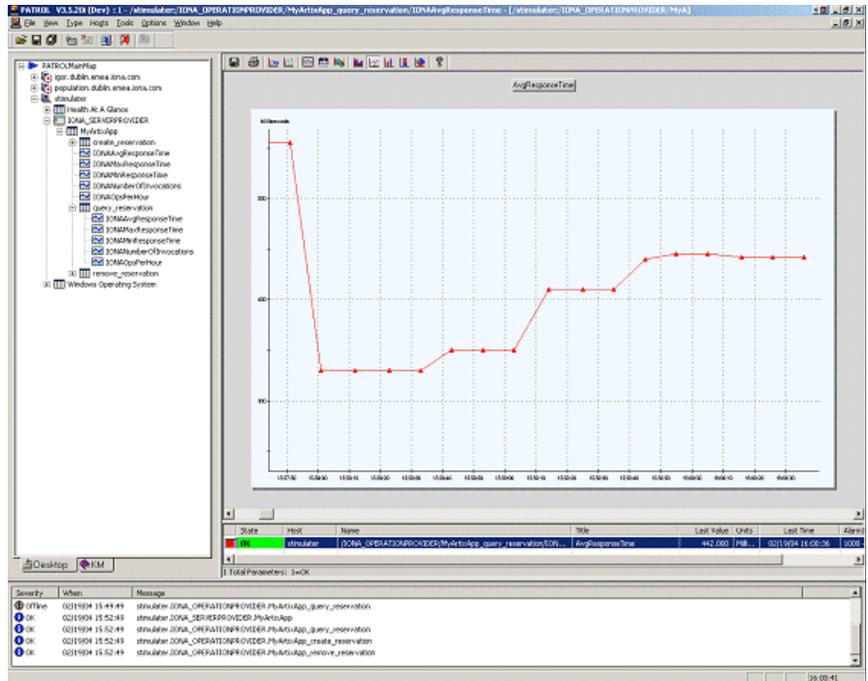
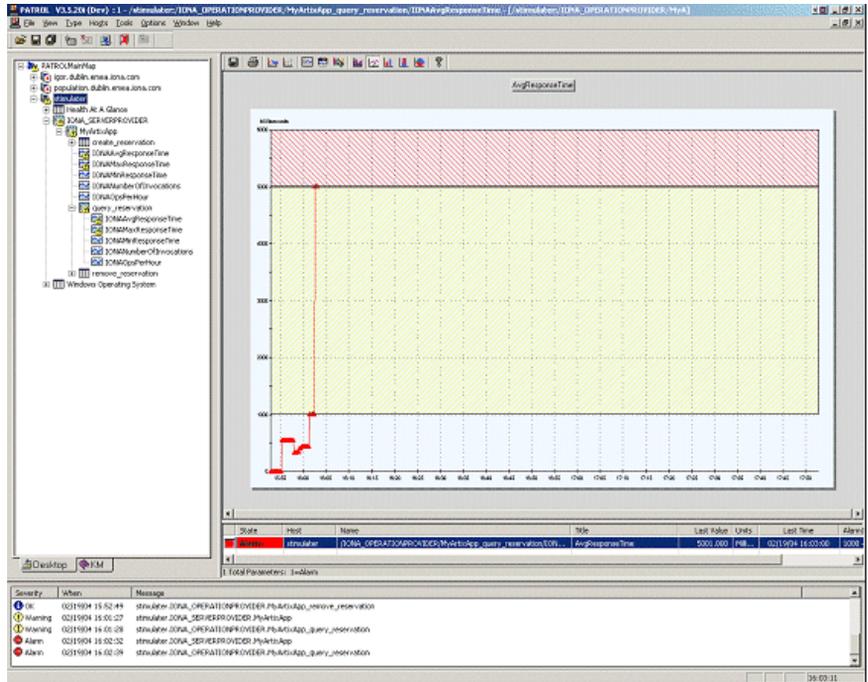


Figure 27 shows warnings and alarms issued for the IONAAvgResponseTime parameter.

**Figure 27:** Alarms for IONAAvgResponseTime



---

## Starting, stopping and restarting servers

The `server_commands.txt` file contains the details about the commands for services that you are deploying on your host (see [Chapter 12](#)). To execute commands in this file, perform the following steps:

1. Right click on an instance in the BMC Patrol Console **Main Map**.
2. Select **Knowledge Module Commands|IONA|Commands**.
3. Select one of the following commands:

<b>Start</b>	Starts a server.
<b>Stop</b>	Stops a server.
<b>Restart</b>	Executes a stop followed by a start.

---

## Troubleshooting

If you have difficulty getting the Artix BMC Patrol integration working, you can use the menu commands to cause debug output to be sent to the system output window.

To view the system output window for a particular host, right click on the icon for your selected host in the BMC Patrol **Main Map**, and choose **System Output Window**.

You can change the level of diagnostics for a particular instance by right clicking on that instance and choosing:

### **Knowledge Module Commands|IONA|Log Levels**

You can choose from the following levels:

- **Set to Error**
- **Set to Info**
- **Set to Debug**

**Set to Debug** provides the highest level of feedback and **Set to Error** provides the lowest.



# Extending to a BMC Production Environment

*This section describes how to extend an Artix BMC Patrol integration from a test environment to a production environment.*

---

**In this chapter**

This chapter contains the following sections:

<a href="#">Configuring an Artix Production Environment</a>	<a href="#">page 158</a>
---	--------------------------

---

# Configuring an Artix Production Environment

---

## Overview

This section describes the steps that you need to take when extending the BMC Patrol integration from an Artix test environment to a production environment. It includes the following sections:

- [“Monitoring your Artix applications”](#)
  - [“Monitoring Artix applications on multiple hosts”](#)
  - [“Monitoring multiple Artix applications on the same host”](#)
- 

## Monitoring your Artix applications

You must add configuration settings to your Artix server configuration files.

For C++ applications, add the following example configuration settings to your Artix application’s `.cfg` file:

```
// my_app.cfg

my_application {

# Ensure that it_response_time_collector is in your orb_plugins list.
orb_plugins = [ ..., "it_response_time_collector" ];

# Enable performance logging.
use_performance_logging = true;

# Collector period (in seconds). How often performance information is logged.
plugins:it_response_time_collector:period = "60";

# Set the name of the file which holds the performance log
plugins:it_response_time_collector:filename =
    "/opt/myapplication/log/myapplication_perf.log"

};
```

**Note:** The specified `plugins:it_response_time_collector:period` should divide evenly into your cycle time (for example, a period of 20 and a cycle time of 60).

## Monitoring Artix applications on multiple hosts

To monitor your Artix applications on multiple hosts, you must distribute the Artix KM to your hosts. The best approach to distributing the Artix Knowledge Module to a large number of machines is to use the Knowledge Module Distribution Service (KMDS).

### Using the KMDS to distribute the IONA KM

To create a deployment set for machines that run Patrol Agents (but not the Patrol Console), perform the following steps:

1. Choose a machine with the Patrol Developer Console installed. Follow the procedure for installing the Artix KM on this machine (see [“Setting up your BMC Patrol Environment”](#) on page 148).
2. Start the Patrol Developer Console and choose **Edit Package** from the list of menu Items.
3. Open the following file:

```
$PATROL_HOME/archives/IONA_Server_KM_Agent_Resources.pkg file
```

You will see a list of all the files that need to be installed on machines that run the Patrol Agent.

4. Now select **Check In Package** from the **File** menu to check the package into the KMDS.
5. You can now use the KMDS Manager to create a deployment set based on this KM package, and distribute it to all the machines that Artix installed and that also have a Patrol Agent.
6. You repeat this process for the  
IONA\_Server\_KM\_Console\_Resources.pkg file.

This creates a deployment set for all machines that have both the Patrol Agent and Patrol Console installed, and which will be used to monitor Artix applications.

For further details about using the KMDS, see your BMC Patrol documentation.

---

**Monitoring multiple Artix applications on the same host**

Sometimes you may need to deploy multiple Artix applications on the same host. The solution is simply to merge the `servers.conf` and `server_commands.txt` files from each of the applications into single `servers.conf` and `server_commands.txt` files.

For example, if the `servers.conf` file from the `UnderwriterCalc` application looks as follows:

```
UnderwriterCalc,1,/opt/myAppUnderwritierCalc/log/UnderwriterCalc_perf.log
```

And the `servers.conf` file for the `ManagePolicy` application looks as follows:

```
ManagePolicy, 1, /opt/ManagePolicyApp/log/ManagePolicy_perf.log
```

The merged `servers.conf` file will then include the following two lines:

```
UnderwriterCalc,1,/opt/myAppUnderwritierCalc/log/UnderwriterCalc_perf.log  
ManagePolicy, 1, /opt/ManagePolicyApp/log/ManagePolicy_perf.log
```

You can now copy this merged file to your `$PATROL_HOME/lib/iona/conf` directory and BMC Patrol will monitor both applications.

Exactly the same procedure applies to the `server_commands.txt` file.

---

**Further information**

For more detailed information on the BMC Patrol consoles, see your BMC Patrol documentation.

# Index

## A

- acksto uri 81
- Actional agent 94, 100
- Actional Agent Interceptor SDK 94
- Actional CSO 100
- Actional for Continuous Service Optimization 100
- Actional for SOA Operations 100
- Actional Point of Operational Visibility 100
- Actional server 94
- Actional server, configuration 101
- Actional Server Administration Console 39, 94, 95, 106
- Add Logging Policy 132
- Add Policy 132
- address 63
- alarms 137, 139, 154
- alerts 92, 94
- AmberPoint Nano Agent API 118
- AmberPoint Nano Agent Client 131
- AmberPoint Nano Agent Server 124, 131
- AmberPoint Proxy Agent 112
- AmberPoint server 125
- Apache Derby 94, 101
- Apache Tomcat 94
- ap\_nano\_agent 128, 129
- apobserver.configuration 127, 130
- application server 124
- apsocketconverter 125
- apsocketconverter.war 124
- arguments 50
- Artix AmberPoint agent 115, 116
- Artix interceptor 117
- Artix interceptors 94
- Artix Java handlers 94
- Artix router 119, 129
- Artix service endpoint 95, 117
- asSent 131
- Attributes tab 84
- averageResponseTime 57

## B

- binding:artix:client\_message\_interceptor\_list 104
- binding:artix:client\_request\_interceptor\_list 104

- binding:artix:server\_message\_interceptor\_list 104
- binding:artix:server\_request\_interceptor\_list 104
- BMC Console 149
- BMC Patrol Agent 149
- bus
  - attributes 49
  - ObjectName 49
- bus\_management 68

## C

- callbacks 119
- canonical 129
- Check In Package 159
- collector 151
- commands 155
- CompositeData 64
- connector.url 72
- consumer 95, 117
- Criteria for this policy 126
- Customization tab 151
- custom JMX MBeans 46
- cycle time 158

## D

- database 94, 101
- Dependencies 131
- dependency diagrams 120
- dependency mapping 92
- deployment modes 116
- Deployments 125
- diagnostics 155
- dynamic discovery 118, 120

## E

- Edit Package 159
- EMS 136
- endpoint 95, 117
- Enterprise Management System 136
- event\_log:filters 35

## F

- File menu 159

**G**

getLoggingLevel 51  
Go 126

**H**

hostMapper algorithm 130  
HSQL 124  
HTTP adaptor 76  
HTTP port 124

**I**

identifier 50  
Infrastructure 125  
instrumented node 92  
interceptor 117  
interceptors 63, 74  
IONAAvgResponseTime 142, 150, 152, 153, 154  
IONA\_km.tgz 148  
IONA\_km.zip 144, 148  
IONAMaxResponseTime 142, 150, 152  
IONAMinResponseTime 142, 150, 152  
IONANumInvocations 142, 150, 151, 152  
IONA\_OPERATIONPROVIDER 140, 149  
IONAOpsPerHour 142, 150, 152  
IONA\_SERVER.kml 149  
IONA\_Server\_KM\_Agent\_Resources.pkg 159  
IONA\_Server\_KM\_Console\_Resources.pkg 159  
IONA\_SERVERPROVIDER 140, 149, 150  
ipaddress 129  
it\_response\_time\_collector 158

**J**

J2SE 5.0 Monitoring and Management Console 83  
Java, requirements 149  
Java Management Extensions 43, 67  
JConsole 72, 83  
    Connect to Agent 83  
JMX 43, 67  
JMX HTTP adaptor 76  
JMX Remote 47  
JMXServiceURL 68

**K**

KMDS 159  
Knowledge Module Distribution Service 159  
Knowledge Modules 142

**L**

Load KMs dialog 149  
local\_log\_stream 35  
locator  
    managed attributes 60  
log file interpreter 139  
logging  
    levels 51  
    subsystems 51  
logging period 151  
logging policies 132  
Log Levels 155

**M**

Main Map 149, 155  
Managed Beans 34, 44  
managed node 102  
managed node, configuration 102  
management consoles 71  
MBeans 34, 44  
MBeanServer 44, 80  
MBeanServerConnection 46  
MBeans tab 84  
menu commands 139, 155  
Message Log 132  
messageLogReader logLocation 127  
messageLogWriter logLocation 127  
messages 81  
MonitorEnabler 126  
monitoring 120

**N**

Nano Agent API 118  
Network Overview 103, 106

**O**

operation  
    parameters 152  
    WSDL 151  
orb\_plugins 104, 158  
Override All Instances checkbox 151

**P**

parameter collector 151  
parameters 150, 152  
Path Explorer 108  
Patrol Agents 159

PATROL Central 151  
 Patrol Developer Console 159  
 Performance 119  
 performance log files 144  
 performance logging  
   period 151  
   plugins 139  
 persistence endpoint 80  
 plugins:ap\_nano\_agent:hostname\_address:local\_hostname 130  
 plugins:ap\_nano\_agent:hostname\_address:publish\_hostname 129  
 plugins:artix:db  
   home 82  
 plugins:bus:register\_client\_context 130  
 plugins:bus\_management:connector:enabled 68, 82  
 plugins:bus\_management:connector:port 82  
 plugins:bus\_management:connector:registry:required 69  
 plugins:bus\_management:connector:url:file 69, 82  
 plugins:bus\_management:connector:url:publish 69  
 plugins:bus\_management:enabled 68, 82  
 plugins:bus\_management:http\_adaptor:enabled 76  
 plugins:bus\_management:http\_adaptor:port 76  
 plugins:it\_response\_time\_collector:filename 158  
 plugins:it\_response\_time\_collector:period 158  
 plugins:routing:use\_pass\_through 129  
 policies:at\_http\_server\_address\_mode\_policy:publish\_hostname 130  
 policies:soap:server\_address\_mode\_policy:publish\_hostname 130  
 Policy Status 126  
 port 124  
   name 63  
   ObjectName 63  
 port, WSDL 151  
 port name 80  
 ports 56  
 Preview Services 126  
 Properties 151  
 Properties menu 151  
 provisioning 103  
 proxy agent 112

## R

Register 126  
 registeredEndpoints 60, 62  
 registeredNodeErrors 60  
 registeredServices 60, 62

Register Message Source 126  
 relational database 124  
 remote access port 69  
 remote JMX clients 68  
 reporting 118  
 requestsOneway 57  
 requestsSinceLastCheck 57  
 requestsTotal 57  
 response time 92  
 response times 40, 137  
 Restart 155  
 RMEndpointPersistentStore 80  
 RMI Connector 68  
 RMSequencePersistentStore 80  
 router 119, 129  
 runtime MBeans 46

## S

scope 49  
 Selector 131  
 sequence\_guid 86  
 sequence ID 86  
 sequence id 81  
 server\_commands.txt 145, 160  
 server parameters 150  
 servers.conf 144, 160  
 service  
   attributes 56  
   managed components 55  
   methods 58  
   name 56  
   ObjectName 56  
 service consumer 95, 117  
 serviceCounters 56  
 service endpoint 95, 117  
 serviceGroups 62  
 Service Level Agreements 120, 132  
 serviceLookupErrors 60  
 serviceLookups 60  
 service name 80  
 services 50  
 serviceSessions 62  
 servicesMonitoring 50  
 session manager  
   managed attributes 62  
 setLoggingLevel 51  
 setLoggingLevelPropagate 51  
 Set to Debug 155  
 Set to Error 155

## INDEX

- Set to Info 155
- Simple Message Source 126
- SLAs 120, 132
- SOA management 112
- SOAP/HTTP 113
- SOAP over HTTP 100
- SOAP over JMS 100
- Start 155
- state 56
- Stop 155
- System Output Window 155

## T

- TabularData 64
- timeSinceLastCheck 57
- Tomcat 94, 124
- totalErrors 57
- transport 64
- troubleshooting 155

## U

- UNIX 148
- unqualified 129
- useFQN 131
- useIpAddr 131
- use\_performance\_logging 158

## W

- warnings 154
- Windows 148
- WSDL
  - operation 151
  - port 151
- WS-RM persistence 80

## X

- xmlfile\_log\_stream 35