

Sterling Commerce
An IBM Company

Sterling Multi-Channel Fulfillment Solution

High Availability Guide

Release 8.0

January 2008



Copyright Notice

Copyright © 1999 - 2008

Sterling Commerce, Inc. ALL RIGHTS RESERVED

STERLING COMMERCE SOFTWARE

TRADE SECRET NOTICE

THE STERLING COMMERCE SOFTWARE DESCRIBED BY THIS DOCUMENTATION ("STERLING COMMERCE SOFTWARE") IS THE CONFIDENTIAL AND TRADE SECRET PROPERTY OF STERLING COMMERCE, INC., ITS AFFILIATED COMPANIES OR ITS OR THEIR LICENSORS, AND IS PROVIDED UNDER THE TERMS OF A LICENSE AGREEMENT. NO DUPLICATION OR DISCLOSURE WITHOUT PRIOR WRITTEN PERMISSION. RESTRICTED RIGHTS.

This documentation, the Sterling Commerce Software it describes, and the information and know-how they contain constitute the proprietary, confidential and valuable trade secret information of Sterling Commerce, Inc., its affiliated companies or its or their licensors, and may not be used for any unauthorized purpose, or disclosed to others without the prior written permission of the applicable Sterling Commerce entity. This documentation and the Sterling Commerce Software that it describes have been provided pursuant to a license agreement that contains prohibitions against and/or restrictions on their copying, modification and use. Duplication, in whole or in part, if and when permitted, shall bear this notice and the Sterling Commerce, Inc. copyright notice.

U.S. GOVERNMENT RESTRICTED RIGHTS. This documentation and the Sterling Commerce Software it describes are "commercial items" as defined in 48 C.F.R. 2.101. As and when provided to any agency or instrumentality of the U.S. Government or to a U.S. Government prime contractor or a subcontractor at any tier ("Government Licensee"), the terms and conditions of the customary Sterling Commerce commercial license agreement are imposed on Government Licensees per 48 C.F.R. 12.212 or 227.7202 through 227.7202-4, as applicable, or through 48 C.F.R. 52.244-6.

These terms of use shall be governed by the laws of the State of Ohio, USA, without regard to its conflict of laws provisions. If you are accessing the Sterling Commerce Software under an executed agreement, then nothing in these terms and conditions supersedes or modifies the executed agreement

Sterling Commerce, Inc.
4600 Lakehurst Court
Dublin, Ohio
43016-2000

Copyright © 1999 - 2008

THIRD PARTY SOFTWARE AND OTHER MATERIAL

Portions of the Sterling Commerce Software may include products, or may be distributed on the same storage media with products, ("Third Party Software") offered by third parties ("Third Party Licensors"). Sterling Commerce Software may include Third Party Software covered by the following copyrights: Copyright © 1999-2005 The Apache Software Foundation. Copyright 1999-2007 Erik Arvidson and Emil A. Eklund. Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. Copyright © 2003 Infragistics, Inc. Copyright © 2001 LOOX Software, Inc. Copyright 2002-2004 © MetaStuff, Ltd. Copyright (C) Microsoft Corp. 1981-1998. Copyright © 1999-2005 Northwoods Software Corporation. Copyright © 2001 Peter Belesis. Copyright © 1995 - 1998 Purple Technology, Inc. Copyright © 2005 Sabre Airline Solutions. Copyright (c) 2006-2007 Sam Stephenson. Copyright © 2004 SoftComplex, Inc. Copyright © 2000-2004 Sun Microsystems, Inc. Copyright © 2001 VisualSoft Technologies Limited. Copyright © 1994 - 2007 World Wide Web Consortium (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). Copyright © 1998-2000 World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). Copyright © 2001 Zero G Software, Inc. All rights reserved by all listed parties.

The Sterling Commerce Software is distributed on the same storage media as certain Third Party Software covered by the following copyrights: Copyright © 1999-2006 The Apache Software Foundation. Copyright (c) 2001-2003 Ant-Contrib project. Copyright © 1998-2007 Bela Ban. Copyright © 2005 Eclipse Foundation. Copyright © 2002-2006 Julian Hyde and others. Copyright © 1997 ICE Engineering, Inc./Timothy Gerard Endres. Copyright © 1987-2006 ILOG, Inc. Copyright © 2000-2006 Infragistics. Copyright © 2002-2005 JBoss, Inc. Copyright LuMriX.net GmbH, Switzerland. Copyright © 1995-2002 MySQL AB. Copyright © 1999-2002 JBoss.org. Copyright Raghu K, 2003. Copyright © 2004 David Schweinsberg. Copyright © 2005-2006 Darren L. Spurgeon. Copyright © S.E. Morris (FISH) 2003-04. Copyright © 2006 VisualSoft Technologies. All rights reserved by all listed parties.

The Sterling Commerce Software is designed to be compatible with or implement a variety of standards issued by Third Party Licensors. The Sterling Commerce Software and related documentation may include copyrightable material of such Third Party Licensors, such as: Copyright © 2006 APCA [Australian Payments Clearing Association Limited]. Copyright © European Central Bank, Frankfurt am Main, Germany. Copyright © 2006 Fix Protocol Limited. Copyright Year 2002-2006 IFX Forum, Inc. The Licensor of the FpML Specifications is the International Swaps and Derivatives Association. Copyright © 2006 National Automated Clearinghouse Association. Open Financial Exchange Specification © 2006 by its publishers: CheckFree Corp., Intuit Inc., and Microsoft Corporation. Copyright © SAP AG 2006, Copyright © S.W.I.F.T. SCRL, Avenue Adele, 1, B-1310 La Hulpe, Belgium 2005. The Licensor of the TWIST Standards Specifications is the to be formed TWIST Standards Foundation. All rights reserved by all listed parties.

The FpML Specifications Version 2.0 referenced in the Sterling Commerce Software or related documentation are subject to the FpML Public License; you may not use the FpML Specifications except in compliance with the FpML Public License. You may obtain a copy of the FpML Public License at <http://www.FpML.org>. The FpML Specifications distributed under the FpML Public License are distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the FpML Public License for the specific language governing rights and limitations under the FpML Public License. The Licensor of the FpML Specifications is the International Swaps and Derivatives Association, Inc. All rights reserved.

The Target2 Standards are available from the European Central Bank through its website, <http://www.ecb.int/paym/target/target2/html/index.en.html>. Additionally, the Target2 Standards may be obtained online at <http://www.dnb.nl/dnb/pagina.jsp?pid=tcm:13-46726-64>.

The TWIST Standards Specifications referenced in the Sterling Commerce Software or related documentation are subject to the TWIST Standards Public License; you may not use the TWIST Standards Specifications except in compliance with the TWIST Standards Public License. The TWIST Specifications distributed under the TWIST Public License are distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the TWIST Standards Public License for the specific language governing rights and limitations under the TWIST Standards Public License. The Licensor of the TWIST Standards Specifications is the, to be formed, TWIST Standards Foundation, all rights reserved.

Third Party Software which is included, or are distributed on the same storage media with, the Sterling Commerce Software where use, duplication, or disclosure by the United States government or a government contractor or subcontractor, are provided with RESTRICTED RIGHTS under Title 48 CFR 2.101, 12.212, 52.227-19, 227.7201 through 227.7202-4, DFAR 252.227-7013(c) (1) (ii) and (2), DFAR 252.227-7015(b)(6/95), DFAR 227.7202-3(a), FAR 52.227-14(g)(2)(6/87), and FAR 52.227-19(c)(2) and (6/87) as applicable.

Additional information regarding certain Third Party Software is located at <install_dir>/Readme.html.

Some Third Party Licensors also provide license information and/or source code for their software via their respective links set forth below.

<http://www.sun.com/software/xml/developers/xsdlib2>

<http://www.dhtmlab.com/>

<http://java.sun.com/j2se/downloads.html>

<http://java.sun.com/products/jsse/index-103.html>

<http://danadler.com/jacob/>

<http://www.dom4j.org>

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). This product includes software developed by the Ant-Contrib project (<http://sourceforge.net/projects/ant-contrib>). This product includes software developed by the JDOM Project (<http://www.jdom.org/>). This product includes code licensed from RSA Data Security (via Sun Microsystems, Inc.). Sun, Sun Microsystems, the Sun Logo, Java, JDK, the Java Coffee Cup logo, JavaBeans, JDBC, JMX and all JMX based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. All other trademarks and logos are trademarks of their respective owners.

JBoss Software

The Sterling Commerce Software is distributed on the same storage media as the JBoss Software (Copyright © 1999-2002 JBoss.org) ("JBoss Software"). The JBoss Software is independent from and not linked or compiled with the Sterling Commerce Software. The JBoss Software is a free software product which can be distributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License or any later version.

A copy of the GNU Lesser General Public License is provided at:
<install_dir>\jar\jboss\4_2_0\LICENSE.html

This license only applies to the JBoss Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The JBoss Software was modified slightly in order to allow the ClientSocketFactory to return a socket connected to a particular host in order to control the host interfaces, regardless of whether the ClientSocket Factory specified was custom or not. Changes were made to org.jnp.server.Main. Details concerning this change can be found at
http://sourceforge.net/tracker/?func=detail&aid=1008902&group_id=22866&atid=376687.

Source code for the modifications completed by Sterling Commerce on August 13, 2004 is located at:
http://sourceforge.net/tracker/?func=detail&aid=1008902&group_id=22866&atid=376687. Source code for all other components of the JBoss Software is located at <http://www.jboss.org>.

The Eclipse Software Foundation

The Sterling Commerce Software is also distributed with or on the same storage media as the following software:

com.ibm.icu.nl1_3.4.4.v200606220026.jar, org.eclipse.ant.core.nl1_3.1.100.v200606220026.jar, org.eclipse.ant.ui.nl1_3.2.0.v200606220026.jar, org.eclipse.compare.nl1_3.2.0.v200606220026.jar, org.eclipse.core.boot.nl1_3.1.100.v200606220026.jar, org.eclipse.core.commands.nl1_3.2.0.v200606220026.jar,

org.eclipse.core.contenttype.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.expressions.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.filebuffers.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.filesystem.nl1_1.0.0.v200606220026.jar,
org.eclipse.core.jobs.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.resources.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.runtime.compatibility.auth.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.runtime.compatibility.nl1_3.1.100.v200606220026.jar,
org.eclipse.core.runtime.nl1_3.2.0.v200606220026.jar,
org.eclipse.core.variables.nl1_3.1.100.v200606220026.jar,
org.eclipse.debug.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.debug.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.equinox.common.nl1_3.2.0.v200606220026.jar,
org.eclipse.equinox.preferences.nl1_3.2.0.v200606220026.jar,
org.eclipse.equinox.registry.nl1_3.2.0.v200606220026.jar,
org.eclipse.help.appserver.nl1_3.1.100.v200606220026.jar,
org.eclipse.help.base.nl1_3.2.0.v200606220026.jar, org.eclipse.help.nl1_3.2.0.v200606220026.jar,
org.eclipse.help.ui.nl1_3.2.0.v200606220026.jar, org.eclipse.jdt.appt.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.appt.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.core.manipulation.nl1_1.0.0.v200606220026.jar,
org.eclipse.jdt.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.debug.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.doc.isv.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.doc.user.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.junit4.runtime.nl1_1.0.0.v200606220026.jar,
org.eclipse.jdt.launching.nl1_3.2.0.v200606220026.jar, org.eclipse.jdt.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.jface.databinding.nl1_1.0.0.v200606220026.jar,
org.eclipse.jface.nl1_3.2.0.v200606220026.jar, org.eclipse.jface.text.nl1_3.2.0.v200606220026.jar,
org.eclipse.ltk.core.refactoring.nl1_3.2.0.v200606220026.jar,
org.eclipse.ltk.ui.refactoring.nl1_3.2.0.v200606220026.jar,
org.eclipse.osgi.nl1_3.2.0.v200606220026.jar, org.eclipse.osgi.services.nl1_3.1.100.v200606220026.jar,
org.eclipse.osgi.util.nl1_3.1.100.v200606220026.jar, org.eclipse.pde.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.pde.doc.user.nl1_3.2.0.v200606220026.jar,
org.eclipse.pde.junit.runtime.nl1_3.2.0.v200606220026.jar,
org.eclipse.pde.nl1_3.2.0.v200606220026.jar, org.eclipse.pde.runtime.nl1_3.2.0.v200606220026.jar,
org.eclipse.pde.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.platform.doc.isv.nl1_3.2.0.v200606220026.jar,
org.eclipse.platform.doc.user.nl1_3.2.0.v200606220026.jar,
org.eclipse.rcp.nl1_3.2.0.v200606220026.jar, org.eclipse.search.nl1_3.2.0.v200606220026.jar,
org.eclipse.swt.nl1_3.2.0.v200606220026.jar, org.eclipse.team.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.team.cvs.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.team.cvs.ssh.nl1_3.2.0.v200606220026.jar,
org.eclipse.team.cvs.ssh2.nl1_3.2.0.v200606220026.jar,
org.eclipse.team.cvs.ui.nl1_3.2.0.v200606220026.jar, org.eclipse.team.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.text.nl1_3.2.0.v200606220026.jar, org.eclipse.ui.browser.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.cheatsheets.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.console.nl1_3.1.100.v200606220026.jar,
org.eclipse.ui.editors.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.externaltools.nl1_3.1.100.v200606220026.jar,
org.eclipse.ui.forms.nl1_3.2.0.v200606220026.jar, org.eclipse.ui.ide.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.intro.nl1_3.2.0.v200606220026.jar, org.eclipse.ui.navigator.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.navigator.resources.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.presentations.r21.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.views.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.views.properties.tabbed.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.workbench.nl1_3.2.0.v200606220026.jar,
org.eclipse.ui.workbench.texteditor.nl1_3.2.0.v200606220026.jar,
org.eclipse.update.configurator.nl1_3.2.0.v200606220026.jar,
org.eclipse.update.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.update.scheduler.nl1_3.2.0.v200606220026.jar,

org.eclipse.update.ui.nl1_3.2.0.v200606220026.jar,
com.ibm.icu_3.4.4.1.jar, org.eclipse.core.commands_3.2.0.I20060605-1400.jar,
org.eclipse.core.contenttype_3.2.0.v20060603.jar,
org.eclipse.core.expressions_3.2.0.v20060605-1400.jar,
org.eclipse.core.filesystem.linux.x86_1.0.0.v20060603.jar,
org.eclipse.core.filesystem_1.0.0.v20060603.jar, org.eclipse.core.jobs_3.2.0.v20060603.jar,
org.eclipse.core.runtime.compatibility.auth_3.2.0.v20060601.jar,
org.eclipse.core.runtime_3.2.0.v20060603.jar, org.eclipse.equinox.common_3.2.0.v20060603.jar,
org.eclipse.equinox.preferences_3.2.0.v20060601.jar, org.eclipse.equinox.registry_3.2.0.v20060601.jar,
org.eclipse.help_3.2.0.v20060602.jar, org.eclipse.jface.text_3.2.0.v20060605-1400.jar,
org.eclipse.jface_3.2.0.I20060605-1400.jar, org.eclipse.osgi_3.2.0.v20060601.jar,
org.eclipse.swt.gtk.linux.x86_3.2.0.v3232m.jar, org.eclipse.swt_3.2.0.v3232o.jar,
org.eclipse.text_3.2.0.v20060605-1400.jar,
org.eclipse.ui.workbench.texteditor_3.2.0.v20060605-1400.jar,
org.eclipse.ui.workbench_3.2.0.I20060605-1400.jar, org.eclipse.ui_3.2.0.I20060605-1400.jar,
runtime_registry_compatibility.jar, eclipse.exe, eclipse.ini, and startup.jar

(collectively, "Eclipse Software").

All Eclipse Software is distributed under the terms and conditions of the Eclipse Foundation Software User Agreement (EFSUA) and/or terms and conditions of the Eclipse Public License Version 1.0 (EPL) or other license agreements, notices or terms and conditions referenced for the individual pieces of the Eclipse Software, including without limitation "Abouts", "Feature Licenses", and "Feature Update Licenses" as defined in the EFSUA .

A copy of the Eclipse Foundation Software User Agreement is found at
<install_dir>/rcpdependencies/windows/eclipse/notice.html,
<install_dir>/rcpdependencies/windows/eclipse/plugins/notice.html,
<install_dir>/rcpdependencies/gtk.linux.x86/eclipse/notice.html, and
<install_dir>/rcpdependencies/gtk.linux.x86/eclipse/plugins/notice.html.

A copy of the EPL is found at
<install_dir>/rcpdependencies/windows/eclipse/plugins/epl-v10.htm,
<install_dir>/rcpdependencies/windows/eclipse/epl-v10.htm,
<install_dir>/rcpdependencies/gtk.linux.x86/eclipse/plugins/epl-v10.html, and
<install_dir>/rcpdependencies/gtk.linux.x86/eclipse/epl-v10.html.

The reference to the license agreements, notices or terms and conditions governing each individual piece of the Eclipse Software is found in the directory files for the individual pieces of the Eclipse Software as described in the file identified as installdir/SCI_License.txt.

These licenses only apply to the Eclipse Software and do not apply to the Sterling Commerce Software, or any other Third Party Software.

The Language Pack (NL Pack) piece of the Eclipse Software, is distributed in object code form. Source code is available at
http://download.eclipse.org/eclipse/downloads/drops/L-3.2_Language_Packs-200607121700/index.php.
In the event the source code is no longer available from the website referenced above, contact Sterling Commerce at 978-513-6000 and ask for the Release Manager. A copy of this license is located at
<install_dir>/rcpdependencies/windows/eclipse/plugins/epl-v10.htm and
<install_dir>/rcpdependencies/gtk.linux.x86/eclipse/plugins/epl-v10.html.

The org.eclipse.core.runtime_3.2.0.v20060603.jar piece of the Eclipse Software was modified slightly in order to remove classes containing encryption items. The org.eclipse.core.runtime_3.2.0.v20060603.jar was modified to remove the Cipher, CipherInputStream and CipherOutputStream classes and rebuild the org.eclipse.core.runtime_3.2.0.v20060603.jar.

ICE SOFTWARE and TEE UTILITY SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the ICE Software (Copyright © 1997 ICE Engineering, Inc./Timothy Gerard Endres.) ("ICE Software") and the Tee Utility Software (Copyright © 2002 Karl M. Syring) ("Tee Utility Software"). The ICE Software and the Tee Utility Software are independent from and not linked or compiled with the Sterling Commerce Software.

The ICE Software and Tee Utility Software are free software products which can be distributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License or any later version.

A copy of the GNU General Public License is provided at <install_dir>/jar/jniregistry/1_2/ICE_License.txt and at <install_dir>/Tee_Utility.txt. This license only applies to the ICE Software and the Tee Utility Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The ICE Software was modified slightly in order to fix a problem discovered by Sterling Commerce involving the RegistryKey class in the RegistryKey.java in the JNIRegistry.jar. The class was modified to comment out the finalize () method and rebuild of the JNIRegistry.jar file.

Source code for the bug fix completed by Sterling Commerce on January 8, 2003 is located at: install_dir/jar/jniregistry/1_2/RegistryKey.java. Source code for all other components of the ICE Software is located at <http://www.trustice.com/java/jnireg/index.shtml>.

The ICE Software and Tee Utility Software are distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

IEEMU.JS

The Sterling Commerce Software is distributed with or on the same storage media as the IEEMU.js (Copyright (c) 1999-2007 Erik Arvidsson and Emil A. Eklund.) ("IEEMU Software"). Created by Erik Arvidsson and Emil A. Eklund <http://webfx.eae.net/contact.html#erik> for WebFX (<http://webfx.eae.net/>). The IEEMU Software is distributed under the terms of the MOZILLA PUBLIC LICENSE Version 1.1. A copy of this license is located at <install_dir>/3rdParty/ieemu license.doc. The IEEMU.js code is distributed in source form and is located at <install_dir>/repository/eardata/platform/war/yfcscripsts/ieemu.js. Neither the Sterling Commerce Software nor any other Third Party Code is a Modification or Contribution subject to the Mozilla Public License. Pursuant to the terms of the Mozilla Public License, the following notice applies only to the IEEMU.js code (and not to the Sterling Commerce Software or any other Third Party Software):

"The contents of the file located at <install_dir>/3rdParty/ieemu license.doc are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is ieemu.js. The Initial Developer is Erik Arvidsson and Emil A. Eklund. Portions created by _____ None Listed _____ and the Initial Developer are Copyright © 1999-2007. All Rights Reserved. Contributor(s): _____ none listed _____.

Alternatively, the contents of this file may be used under the terms of the _____ license (the "[_____] License"), in which case the provisions of [_____] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [_____] License and not allow others to use your version of this file under the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [_____] License. If you do not delete the provisions above, a recipient may use your version of this file under either the MPL or the [_____] License."

The preceding license only applies to the IEEMU Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

JGO SOFTWARE

The Sterling Commerce Software is distributed with, or on the same storage media, as certain redistributable portions of the JGo Software provided by Northwoods Software Corporation under a commercial license agreement (the "JGo Software"). The JGo Software is provided subject to the disclaimers set forth above and the following notice:

U.S. Government Restricted Rights

The JGo Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (C)(1)(ii) of the

Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (C)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor / manufacturer of the JGo Software is Northwoods Software Corporation, 142 Main St., Nashua, NH 03060.

MYSQL SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the MySQL Software (Copyright © 1995-2002 MySQL AB) ("MySQL Software"). Before installing the MySQL Software, the terms and conditions of the MySQL license must be accepted.

A copy of the MySQL license is provided at <install_dir>/mysql/MySQL_License.txt. This license only applies to the MySQL Software and does not apply to the Sterling Commerce Software, or any other Third Party Licensor Software.

THE APACHE SOFTWARE FOUNDATION SOFTWARE

The Sterling Commerce Software is also distributed with or on the same storage media

as the following software products (or components thereof): Ant, Antinstaller, Axis, Apache Commons Lang, Apache Jakarta Commons Collections, Apache Jakarta Commons Pool, Apache Jakarta ORO, Xerces version 2.7, Apache Log4J, Apache SOAP, and Apache Xalan 2.7.0 (collectively, "Apache 2.0 Software"). Apache 2.0 Software is free software which is distributed under the terms of the Apache License Version 2.0. A copy of License Version 2.0 is found in the following directory files for the individual pieces of the Apache 2.0 Software:

```
<install_dir>/ant/Ant_License.txt,  
<install_dir>/jar/antInstaller/0_8/antinstaller_License.txt  
<install_dir>/jar/commons_pool/1_2/Commons_License.txt  
<install_dir>/jar/jakarta_oro/2_0_8/JakartaOro_License.txt  
<install_dir>/jar/log4j/1_2_11/LOG4J_License.txt  
<install_dir>/Xalan_License.txt  
<install_dir>/jar/soap/2_3_1/Apache_SOAP_License.txt  
<install_dir>/jar/commons_collections/2_1/Commons_License.txt  
<install_dir>/jar/commons_lang/2_1/Commons_Lang_License.txt
```

Unless otherwise stated in a specific directory, the Apache 2.0 Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to Apache 2.0 Software, nor other Third Party Code is a Derivative Work or a Contribution as defined in License Version 2.0. License Version 2.0 applies only to the Apache 2.0 Software which is the subject of the specific directory file and does not apply to the Sterling Commerce Software or to any other Third Party Software.

Rico Software

The Sterling Commerce Software is also distributed with or on the same storage media as the Rico.js software (Copyright © 2005 Sabre Airline Solutions) ("Rico Software"). Rico Software is free software which is distributed under the terms of the Apache License Version 2.0. A copy of License Version 2.0 is found <install_dir>/3rdParty/rico license.doc.

The Rico Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to the Rico Software, nor other Third Party Code is a Derivative Work or a Contribution as defined in License Version 2.0. License Version 2.0 applies only to the Rico Software which is the subject of the specific directory file and does not apply to the Sterling Commerce Software or to any other Third Party Software. License Version 2.0 includes the following provision:

"Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License."

SUN MICROSYSTEMS

The Sterling Commerce Software is distributed with or on the same storage media as the following software products (or components thereof): Sun Activation Framework, Sun JSSE, Sun JNet, and Sun JavaMail (collectively, "Sun Software"). Sun Software is free software which is distributed under the terms of the Sun Microsystems, Inc. Binary Code License Agreement ("BCLA"). A copy of the specific BCLA is found in the following directory files for the individual pieces of the Sun Software:

SUN JSSE and JNET JARS - <install_dir>/noapp/lib
SUN JavaMail - <install_dir>/jar/javamail/1_3_2

The licenses relating to the following Sun products are included in the directory files located at:

SUN COMM JAR - <install_dir>/repository/eardata/platform/war/yfscommon
SUN ACTIVATION JAR - <install_dir>/jar/jaf/1_0_2

The Sterling Commerce Software is also distributed with or on the same storage media as the Web-app_2_3.dtd software (Copyright © 2007 Sun Microsystems, Inc.) ("Web-App Software"). Web-App Software is free software which is distributed under the terms of the Common Development and Distribution License ("CDDL"). A copy of the CDDL is found in <install_dir>/repository/eardata/platform/war/WEB-INF/web_app_licence.txt

The source code for the Web-App Software may be found at:
<install_dir>/repository/eardata/platform/war/WEB-INF/web-app+2_3.dtd.

Such licenses only apply to the Sun product which is the subject of such directory and does not apply to the Sterling Commerce Software or to any other Third Party Software.

W3C Software

The Sterling Commerce Software is distributed on the same storage media as the W3C Software to which the following notice applies:

W3C XML Schema

Copyright © 1994-2007 World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. This work is distributed under the W3C® Software License [1] in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[1] <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>

WARRANTY DISCLAIMER

This documentation and the Sterling Commerce Software which it describes are licensed either "AS IS" or with a limited warranty, as set forth in the Sterling Commerce license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE. The applicable Sterling Commerce entity reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

The Third Party Software is provided "AS IS" WITHOUT ANY WARRANTY AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. FURTHER, IF YOU ARE LOCATED OR ACCESSING THIS SOFTWARE IN THE UNITED STATES, ANY EXPRESS OR IMPLIED WARRANTY REGARDING TITLE OR NON-INFRINGEMENT ARE DISCLAIMED.

Without limiting the foregoing, the ICE Software, IEMU Software, and JBoss Software, are all distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Contents

Preface

| | |
|---|-------|
| Structure | xv |
| Prerequisites | xvii |
| References | xviii |
| Scope | xviii |
| Sterling Multi-Channel Fulfillment Solution Documentation | xviii |
| Conventions | xx |

1 Changes

2 Availability

| | | |
|-------|---|---|
| 2.1 | Availability Design and Principles | 1 |
| 2.1.1 | Business Drives High Availability Requirements | 1 |
| 2.1.2 | Keep It Simple Strategy | 2 |
| 2.1.3 | Configuring for Higher Availability or Resilience Is Like Buying Insurance 2 | |
| 2.1.4 | The 9's | 3 |
| 2.2 | Problem with the 9's | 4 |
| 2.3 | High Availability Motivation | 4 |

3 Sterling Multi-Channel Fulfillment Solution Architecture

| | | |
|-----|------------------------------------|---|
| 3.1 | Application Components | 7 |
| 3.2 | Application Server | 9 |
| 3.3 | Agent or Integration Servers | 9 |

| | | |
|-----|-------------------|----|
| 3.4 | JNDI Service..... | 10 |
|-----|-------------------|----|

4 Limited Redundancy Single-Site Configuration

| | | |
|-------|--|----|
| 4.1 | Single Point of Failure | 12 |
| 4.2 | Loss of Data | 12 |
| 4.2.1 | Loss of Database..... | 13 |
| 4.2.2 | Loss of Database Transaction Logs..... | 13 |
| 4.3 | Applicability..... | 13 |

5 High Availability Within A Single Site

| | | |
|---------|---|----|
| 5.1 | Single Points of Failure..... | 15 |
| 5.2 | Node | 16 |
| 5.2.1 | Active/Passive Cluster Failover Configurations | 17 |
| 5.3 | Database Server | 19 |
| 5.3.1 | DBMS Software Failures | 19 |
| 5.3.2 | Human and Operator Errors..... | 20 |
| 5.3.3 | Hardware Failures | 20 |
| 5.3.3.1 | Active/Passive Failover Configurations..... | 20 |
| 5.3.3.2 | Active/Active Failover Configurations..... | 25 |
| 5.4 | SAN or Disk Subsystem..... | 34 |
| 5.5 | Sterling Multi-Channel Fulfillment Solution Components..... | 35 |
| 5.5.1 | Application Server | 35 |
| 5.5.1.1 | Stateful Sessions..... | 36 |
| 5.5.1.2 | Stateless Sessions..... | 36 |
| 5.5.2 | Sterling Multi-Channel Fulfillment Solution Agent and Integration Server 36 | |
| 5.6 | Server Registry | 37 |
| 5.7 | Message Queues | 38 |
| 5.7.1 | Integration Queues for the Sterling Multi-Channel Fulfillment Solution Integration Servers | 38 |
| 5.7.2 | Agent Queues for the Sterling Multi-Channel Fulfillment Solution Agent Servers..... | 39 |
| 5.7.3 | WebLogic JMS..... | 39 |
| 5.7.3.1 | Persistence..... | 40 |
| 5.7.3.2 | Message Paging | 40 |

| | | |
|---------|---|----|
| 5.7.3.3 | Dedicated Integration JMS Server | 41 |
| 5.7.3.4 | Shared Disk Subsystem | 41 |
| 5.7.3.5 | JMS Server on a Different Application Server..... | 41 |
| 5.7.4 | WebSphere Messaging | 42 |
| 5.7.4.1 | WebSphere MQ | 42 |
| 5.8 | Networked File Systems (NFS) | 44 |

6 Architectural Patterns

| | | |
|-----|---|----|
| 6.1 | Asynchronous Integration..... | 46 |
| 6.2 | Caching | 46 |
| 6.3 | Hot Deployment of Code, Configuration, and Fixes..... | 47 |
| 6.4 | Deployment Processes and Regression Testing | 48 |

7 Disaster Recovery

| | | |
|-------|---|----|
| 7.1 | Disaster Recovery from a Sterling Multi-Channel Fulfillment Solution Perspective | 49 |
| 7.2 | For Testing Purpose..... | 50 |
| 7.3 | Cold Site Recovery | 51 |
| 7.4 | Warm and Hot Site Recovery | 52 |
| 7.5 | Key to Disaster Recovery | 52 |
| 7.5.1 | Recovery Procedures | 52 |
| 7.5.2 | Database Backups and Transaction Log Files..... | 52 |
| 7.5.3 | Integration Queue Replication | 53 |
| 7.5.4 | Service Names Instead of IP Addresses | 53 |

Index

Preface

The Sterling Distributed Order Management (DOM) and the Sterling Warehouse Management System (WMS) applications are often deployed in an integrated network of external systems and business partners to form a cohesive business ecosystem. Prolonged application or system outages can have significant business consequences.

This document describes approaches that can increase the resiliency of the Sterling Multi-Channel Fulfillment Solution to environmental, hardware, or software faults or failure. The discussions and material are framed from the perspective of the Sterling Multi-Channel Fulfillment Solution. We have deliberately steered away from discussing non-Sterling Multi-Channel Fulfillment Solution components such as how to protect hardware devices. There are many excellent books on how to design systems for availability.

This document also presents techniques or architectural patterns that can minimize the impact on the overall ecosystem in the event of a planned or unplanned Sterling Multi-Channel Fulfillment Solution system outage.

This document emphasizes a fairly pragmatic approach to availability recognizing that one cannot implement highly available techniques at the detriment of other architectural considerations such as capital cost, ongoing total cost of ownership, or manageability.

Structure

This manual contains the following sections:

Chapter 1, "Changes"

This chapter presents the list of changes to high availability introduced in the release.

Chapter 2, "Availability"

This chapter introduces availability engineering, the often-quoted “nines” and the “five nines” and the problems with relying on the nines as an availability requirement. It also introduces the design principles that Sterling Commerce adheres to when presenting the availability techniques. Sterling Commerce strongly encourages you to read through this section.

Chapter 3, "Sterling Multi-Channel Fulfillment Solution Architecture"

This chapter provides a very high level overview of the Sterling Multi-Channel Fulfillment Solution architecture with a focus on describing the component that needs protection. It assumes that you are familiar with the architecture of the Sterling Multi-Channel Fulfillment Solution. It also helps if you have also installed and used the Sterling Multi-Channel Fulfillment Solution.

Chapter 4, "Limited Redundancy Single-Site Configuration"

This chapter presents a simple, possibly entry-level, Sterling Multi-Channel Fulfillment Solution system that was configured with little attention to availability. In some cases, this simple configuration may meet the customer’s business or availability requirements. For others, this chapter serves as the baseline for comparison as we examine the configurations with increasingly higher levels of resiliency.

Chapter 5, "High Availability Within A Single Site"

This chapter examines common techniques deployed to ensure application availability against faults incurred within the four walls of the data center. It identifies all the hardware and software components used by the Sterling Multi-Channel Fulfillment Solution, all the potential single points of failures (SPOF), and the approaches to protecting these components.

Chapter 6, "Architectural Patterns"

Adding resiliency to the system is not limited to technology. It is important to note that developing a highly available Sterling Multi-Channel Fulfillment Solution-based solution has as much, if not more, to do with the design of the overall solution and integration points as it does with the Sterling Multi-Channel Fulfillment Solution architecture itself. This chapter presents techniques and patterns that can be used to insulate some of these integration points from the planned and unplanned Sterling Multi-Channel Fulfillment Solution downtime. In this way, even though portions of the solution may be unavailable, there is no downtime for the service as a whole especially as perceived by the end users or customers. More importantly, these architectural patterns can significantly simplify or reduce the availability of the Sterling Multi-Channel Fulfillment Solution, requirements, and implementation.

Chapter 7, "Disaster Recovery"

With the advent of 9/11 or the Northeast Blackout, events that disaster planners once dismissed as implausible or far-fetched are now required considerations as part of disaster recovery and business continuity planning. This chapter examines the commonly used techniques that can be employed so that the Sterling Multi-Channel Fulfillment Solution can continue running at remote sites in the event of disasters.

Prerequisites

This document assumes that you are familiar with the architecture of the Sterling Multi-Channel Fulfillment Solution. It also assumes that you have installed and used the Sterling Multi-Channel Fulfillment Solution.

For more detailed information, see the following Sterling Multi-Channel Fulfillment Solution documents:

- *Sterling Multi-Channel Fulfillment Solution Installation Guide* - Provides detailed information on how to install the Sterling Multi-Channel Fulfillment Solution.
- *Sterling Multi-Channel Fulfillment Solution Performance Management Guide* - Provides detailed information on how to configure the Sterling Multi-Channel Fulfillment Solution.

References

[1] IBM eServer p5 590 and 595 System Handbook, SG24-9119-00, IBM Corp, March 17, 2005

[2] NSM: Often the Weakest Link in Business Availability, AV-13-9473, July 3, 2001

Scope

There are many good reference books and material that cover high availability engineering for the data center and the network. This document assumes that you are familiar with high availability engineering for general systems. This document highlights areas in the Sterling Multi-Channel Fulfillment Solution system that are single points of failures, and the approaches that one can take to making them more resilient.

This document assumes that the network is resilient. If your Sterling Multi-Channel Fulfillment Solution configuration supports a large number of users from the Web, Sterling Commerce assumes that your wide-area network deployment is configured in such a way that single (or multiple) faults do not cause an outage.

This document does not address security issues. Sterling Commerce recognizes that in today's connected world, security attacks and security fraud is on the rise. Preventing security hacks from taking down systems is a large and complex area and warrants a separate detailed study.

This document does not address environmental or infrastructure availability. Sterling Commerce assumes that the data center is built with redundant power circuits, redundant cooling, and so forth, so that the infrastructure remains available with single or multiple environmental faults. Sterling Commerce also assumes that the data center is sufficiently equipped with an uninterrupted power supply (UPS) so that all the hardware components can operate under brief power fluctuations and is equipped with power generators to continue working during prolonged power outages.

Sterling Multi-Channel Fulfillment Solution

Documentation

For more information about the Sterling Multi-Channel Fulfillment Solution[®] components, see the following manuals:

- *Sterling Multi-Channel Fulfillment Solution[®] Release Notes*
- *Sterling Selling and Fulfillment Suite[®] Release Notes*
- *Sterling Multi-Channel Fulfillment Solution[®] Installation Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Upgrade Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Configuration Deployment Tool Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Performance Management Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] High Availability Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] System Management Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Localization Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Customization Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Integration Guide*
- *Sterling Selling and Fulfillment Suite[®] Integration Guide*
- *Sterling Multi-Channel Fulfillment Solution[®] Product Concepts*
- *Sterling Warehouse Management System[®] Concepts Guide*
- *Sterling Multi-Channel Fulfillment Solution Platform[®] Configuration Guide*
- *Sterling Distributed Order Management[®] Configuration Guide*
- *Sterling Supply Collaboration[®] Configuration Guide*
- *Sterling Global Inventory Visibility[®] Configuration Guide*
- *Sterling Product Management[®] Configuration Guide*
- *Sterling Logistics Management[®] Configuration Guide*
- *Sterling Reverse Logistics[®] Configuration Guide*
- *Sterling Warehouse Management System[®] Configuration Guide*

- *Sterling Multi-Channel Fulfillment Solution Platform® User Guide*
- *Sterling Distributed Order Management® User Guide*
- *Sterling Supply Collaboration® User Guide*
- *Sterling Global Inventory Visibility® User Guide*
- *Sterling Logistics Management® User Guide*
- *Sterling Reverse Logistics® User Guide*
- *Sterling Warehouse Management System® User Guide*
- *Sterling Multi-Channel Fulfillment Solution Mobile Application® User Guide*
- *Sterling Multi-Channel Fulfillment Solution Analytics® Guide*
- *Sterling Multi-Channel Fulfillment Solution® Javadocs*
- *Sterling Multi-Channel Fulfillment Solution® Glossary*
- *Sterling Parcel Carrier Adapter® Guide*

Conventions

The following conventions may be used in this manual:

| Convention | Meaning |
|------------------|--|
| . . . | Ellipsis represents information that has been omitted. |
| < > | Angle brackets indicate user-supplied input. |
| mono-spaced text | Mono-spaced text indicates a file name, directory path, attribute name, or an inline code example or command. |
| / or \ | Slashes and backslashes are file separators for Windows, UNIX, and Linux operating systems. The file separator for the Windows operating system is "\" and the file separator for UNIX and Linux systems is "/". The UNIX convention is used unless otherwise mentioned. |
| <INSTALL_DIR> | User-supplied location of the Sterling Multi-Channel Fulfillment Solution installation directory. This is only applicable for Release 8.0 or above. |

| Convention | Meaning |
|-------------------|---|
| <INSTALL_DIR_OLD> | User-supplied location of the Sterling Multi-Channel Fulfillment Solution installation directory for previously installed releases. This is only applicable for Release 8.0 or above. |
| <YANTRA_HOME> | User-supplied location of the Sterling Supply Chain Applications installation directory. This is only applicable for Release 7.7, 7.9, and 7.11. |
| <YANTRA_HOME_OLD> | User-supplied location of the Sterling Supply Chain Applications installation directory for previously installed releases. This is only applicable for Releases 7.7, 7.9, and 7.11. |
| <YFS_HOME> | For releases 7.3, 7.5, and 7.5 SP1, this is the user-supplied location of the Sterling Supply Chain Applications installation directory. For releases 7.7, 7.9, and 7.11, this is the user-supplied location of the <YANTRA_HOME>/Runtime directory. For release 8.0, the <YANTRA_HOME>/Runtime directory is no longer used and this is the same location as <INSTALL_DIR>. |
| <YFS_HOME_OLD> | This is the <YANTRA_HOME>/Runtime directory of previously installed releases. This is only applicable for Releases 7.7, 7.9, and 7.11. |
| <ANALYTICS_HOME> | User-supplied location of the Sterling Multi-Channel Fulfillment Solution Analytics installation directory. Note: This convention is used only in the <i>Sterling Multi-Channel Fulfillment Solution Analytics Guide</i> . |
| <COGNOS_HOME> | User-supplied location of the Cognos installation directory. Note: This convention is used only in the <i>Sterling Multi-Channel Fulfillment Solution Analytics Guide</i> . |

| Convention | Meaning |
|------------------------|---|
| <MQ_JAVA_INSTALL_PATH> | <p>User-supplied location of the IBM WebSphere MQ Java components installation directory.</p> <p>Note: This convention is used only in the <i>Sterling Multi-Channel Fulfillment Solution System Management Guide</i>.</p> |
| <DB> | Refers to the Oracle, DB2, or MSSQL depending on the database server. |
| <DB_TYPE> | Depending on the database used, considers the value oracle, db2, or sqlserver. |

1

Changes

This chapter presents the list of high availability changes that were introduced in the Sterling Multi-Channel Fulfillment Solution Release 8.0. See the *Sterling Multi-Channel Fulfillment Solution Release Notes* for the complete list of changes.

The following limitations have been removed in this release:

- In previous releases, we noted that IBM DB2 HADR's Automatic Client Reroute (ACR) feature did not work with the Sterling Supply Chain Applications integration and agent servers. This was due to the fact that our integration and agent servers obtain their database connections through the `java.sql.DriverManager` interface and ACR-only supported database connections obtained from the `javax.sql.DataSource` interface.

This restriction/limitation has been lifted in this release because DB2 9.1 FP3 has extended ACR support to connections obtained from the `java.sql.DriverManager` interface. (see [Section 5.3.3.1.2, "IBM UDB Active/Passive Using HADR"](#)).

- In the previous release, we noted that the Sterling Supply Chain Applications integration and agent servers used the Java Naming and Directory Interface (JNDI) to find its peers. We also noted that the JNDI, as a non-clustered application server resource, was a single point of failure. Servers use the JNDI to locate peers for system management purposes such as to notify peers to flush their table cache.

In this release, the list of servers is now maintained in the database in the `YFS_HEARTBEAT` table. As a result, the mechanism to locate active servers has the same availability posture as the database server.

The following correction has been made in this release:

- In the previous release, we incorrectly stated that the IBM DB2 HADR standby database can be opened for reporting purposes. That reference has been removed.

This chapter provides a brief introduction to availability engineering, with focus on the Sterling Multi-Channel Fulfillment Solution.

2.1 Availability Design and Principles

Availability design is a complex subject that covers a large optimization exercise balancing many requirements.

To help with the availability design, we propose the following principles:

- [Business Drives High Availability Requirements](#)
- [Keep It Simple Strategy](#)
- [Configuring for Higher Availability or Resilience Is Like Buying Insurance](#)

2.1.1 Business Drives High Availability Requirements

High availability (HA) requirements should be driven by business and not implemented for the sake of technology. Seeking to use a new high availability technology, supporting a clustered file system, etc. may be interesting and intellectually satisfying. The most important consideration is that these technologies advances the business goals without making the system overly complicated or expensive.

In some cases, the business may be able to tolerate a certain amount of outage and a simple backup and restore may suffice. Of course, there are others where an hour of downtime is very expensive and as a result require that every component from power to the database be made as resilient as possible through redundancy and automated failover. In

addition, they may mandate geographically dispersed disaster recovery capabilities.

High availability designs cannot be performed in isolation. As in most worthy engineering endeavors, high availability requirements must be balanced against other architectural choices including acquisition cost, maintenance costs, scalability, maintainability, ease of use, impact to business, and so forth.

2.1.2 Keep It Simple Strategy

If possible, you need to manage the complexity of the system, the approaches to high availability and the recovery procedures. Complex systems:

- make it harder for people to understand and manage
- increase the risk of failures
- could make the fault recovery more difficult and in some cases more risky

2.1.3 Configuring for Higher Availability or Resilience Is Like Buying Insurance

In selecting insurance policies, you typically weigh the cost of the insurance against the likelihood that the insurance is needed, whether the insurance is required by law, and the significance of the potential loss if you don't have insurance.

For example, you would likely not take a flood insurance policy regardless of the premium cost if you live on a hilltop in a desert but you would buy a high premium flood insurance if you live in a hurricane zone along the coast. Similarly, when procuring system hardware, you buy servers with high reliability, availability and serviceability (RAS) built in to ensure that hardware faults do not result in an outage. For example, your servers may come equipped with as many as six redundant cooling fans and power supplies.

In some cases, the law may require you to purchase insurance. Similarly, in some business sectors, regulations require business continuity and disaster recovery plans.

At the extremes, if a business is willing and able to tolerate prolonged outage periods, the HA requirements are few. In some cases, having good backups may suffice.

On the other hand, if a business can only tolerate a down time of less than 30 minutes for each outage, you may have to consider having duplicated or redundant components for any component that can fail especially if they are the SPOF.

At the other end of the spectrum, a company may have very high availability requirements and can only tolerate less than five minute downtime for each outage. In that environment, the data center may have to be staffed around the clock, the failure detection must be quick, failover procedures must be automated, and so forth.

2.1.4 The 9's

Specifying availability requirements is not as simple as the often quoted "99.999% availability". In its simplest form, the 9's is an indication of how much downtime an application is allowed to incur. [Table 2-1](#) below shows that each additional 9 drops the amount of time the application could be down for by an order of magnitude.

Table 2-1 The 9's

| Percentage Uptime | Percentage Downtime | Amount of Downtime each Year | Amount of Downtime each Month |
|-------------------|---------------------|------------------------------|-------------------------------|
| 98.0% | 2% | 7.3 days | 14.6 hours |
| 99.0% | 1% | 3.7 days | 7.3 hours |
| 99.8% | 0.2% | 17.5 hours | 1.5 hours |
| 99.9% | 0.1% | 8.8 hours | 43.8 minutes |
| 99.99% | 0.01% | 52.6 minutes | 4.4 minutes |
| 99.999% | 0.001% | 5.3 minutes | 26.3 seconds |
| 99.9999% | 0.0001% | 31.5 seconds | 2.6 seconds |

Therefore, specifying that a system have 99.999% availability means that the system can only be down for less than 5.3 minutes in a year.

2.2 Problem with the 9's

The problem with using the 9's as a requirements is that not all outages are the same. In fact, some customers could architect their solution to tolerate a certain level or type of outage. For example, [Section 6.1, "Asynchronous Integration"](#) on page 46 states that a customer can integrate the customer-facing Web site to the Sterling Multi-Channel Fulfillment Solution, using asynchronous messages. With this architectural pattern, the Sterling Multi-Channel Fulfillment Solution can be taken offline for maintenance (such as upgrades) without impacting the services provided by the Web site.

The use of the 9's also do not account for the different strategies or level of availability of certain workloads. For example, during failures, customers may want to consider shutting down lower-priority workloads (see example in [Section , "Target Utilization"](#) on page 33).

In general, if architected correctly, the Sterling Multi-Channel Fulfillment Solution, which is typically used as a backroom order processing engine, does not have high availability requirements. In contrast, some applications, for example, Internet facing applications, have very high availability requirements because they are customer facing. We will discuss some of the architectural choices later in the document.

2.3 High Availability Motivation

Architecting highly availability systems is not new. They are, in fact, commonplace in industries such as financial. However, many recent events have heightened interests and requirements in availability:

- Catastrophic events such as September 11, 2001 or the Northeast Power Blackout of 2003 have pushed availability to the foreground. Situations that were unimaginable five years ago are now a serious part of business continuity planning. In fact, many corporate managers reject business continuity plans that do not incorporate wide scale disasters.
- Emerging regulations are forcing availability. In the health care industry, the Health Insurance Portability and Accountability Act (HIPAA) mandates business continuity and availability planning. Section 404 of Sarbanes-Oxley specifies that corporations must protect the systems used to report financial information. At a

minimum, corporations are forced to think about the ability to recover those systems.

- Your corporation may be part of a supply chain where inventory needs to be available just-in-time. You may demand or are demanded by your partners to have your systems available to ensure that business partners can communicate. In some situations, trading partners may demand business continuity plans or disaster recovery plans ensuring that services can be restored within a set period of time after catastrophes.

3

Sterling Multi-Channel Fulfillment Solution Architecture

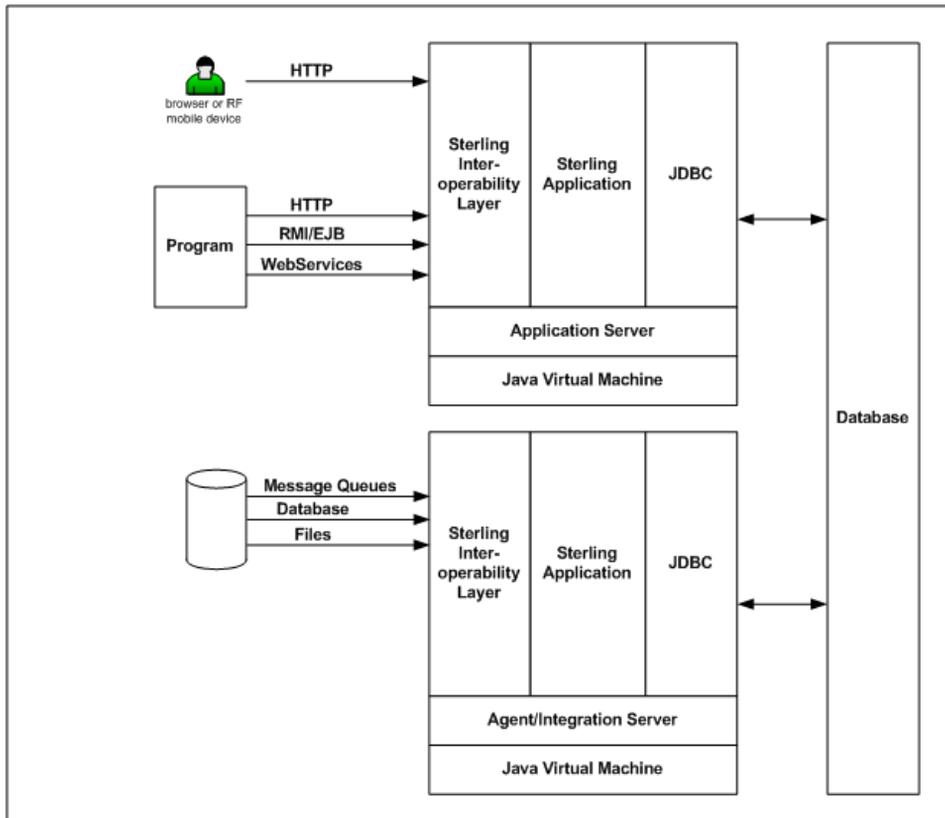
This chapter presents the architecture of the Sterling Multi-Channel Fulfillment Solution at a high level with a focus on describing the component that you have to protect.

3.1 Application Components

The Sterling Multi-Channel Fulfillment Solution runs on one of the following server components (as depicted in [Figure 3–1](#)).

- Application server
- Sterling Multi-Channel Fulfillment Solution agent or integration server

Figure 3–1 The Sterling Multi-Channel Fulfillment Solution Architecture



These components run inside a Java Virtual Machine (JVM). As a result, each component exists as a process in the system. You can have multiple instances of each component. For example, you can run the Sterling Multi-Channel Fulfillment Solution in multiple instances of the application server. Each instance is a separate JVM.

The components use the following services:

- Message queue
- JNDI
- Database server

- LDAP (optional)

3.2 Application Server

The application servers are the processes that handle synchronous requests to provide real-time access to the features and application logic within the Sterling Multi-Channel Fulfillment Solution.

The most common type of requests that an application server handles are the requests originating from clients using the Sterling Multi-Channel Fulfillment Solution Consoles. The application servers are always deployed using an industrial-strength server application such as BEA WebLogic, IBM WebSphere or JBoss Application Server.

The application servers handle real-time requests from users or programs. Requests can be sent in different protocols such as HTTP, servlet calls, EJB/RMI calls, and so forth.

Typical usage scenarios include:

- The call center representative uses the Sterling Multi-Channel Fulfillment Solution Consoles to interact with the Sterling Multi-Channel Fulfillment Solution. For example, to create, query or modify orders, shipments or inventory. Requests come in as HTTP requests.
- Program runs transactions – calls through Remote Method Invocation (RMI), EJB, servlet calls, and so forth.

3.3 Agent or Integration Servers

Integration Servers are Java-based processes that run in the background to process various tasks. Integration servers allow the Sterling Multi-Channel Fulfillment Solution to collaborate with different systems, organizations, and businesses—all through a standard, uniform interface to all systems. Integration Servers and the tasks that they perform are configured through the means of the Service Definition Framework. For more information, see the *Sterling Multi-Channel Fulfillment Solution System Management Guide*.

The integration servers that process information from external systems can get work from message queues, database tables, and files. Integration servers that send work to external systems do so through a

variety of transport mechanisms such as message queues, email, database tables, files, etc.

An agent server is a specialized sub-class of the integration server that runs the Sterling Multi-Channel Fulfillment Solution-defined “time-triggered” transactions. These include transactions to schedule orders. In the transaction configuration screen, you can designate transactions to an agent server. Multiple transactions could be assigned to an agent server. You can also specify that a transaction should run in multiple threads.

For example, if you associate both the Schedule and Release Order transactions to an agent server (`sched_rel_ord_agent`) with 3 threads each, when you start an instance of the `sched_rel_ord_agent` agent server, that server will have six processing threads – three for the Schedule Order transaction and three for the Release Order transaction.

You can also start multiple agent server instances. For example, if you start four `sched_rel_ord_agent` servers, you will see four Java processes running in the system. Each Java process has 3 threads of the Schedule Order and Release Order transactions. In total, you get 12 threads of the Schedule Order and 12 threads of the Release Order transaction.

The agent server relies on the JNDI service. At startup, it registers itself to the JNDI. This allows other servers to locate it.

3.4 JNDI Service

All the servers register themselves in the JNDI on startup. This allows servers to locate other servers. One reason is to refresh the reference data cache. The Sterling Multi-Channel Fulfillment Solution servers cache reference data records for speed and scalability. When a server modifies a reference data record, it notifies all the servers in the JNDI list to refresh their cache.

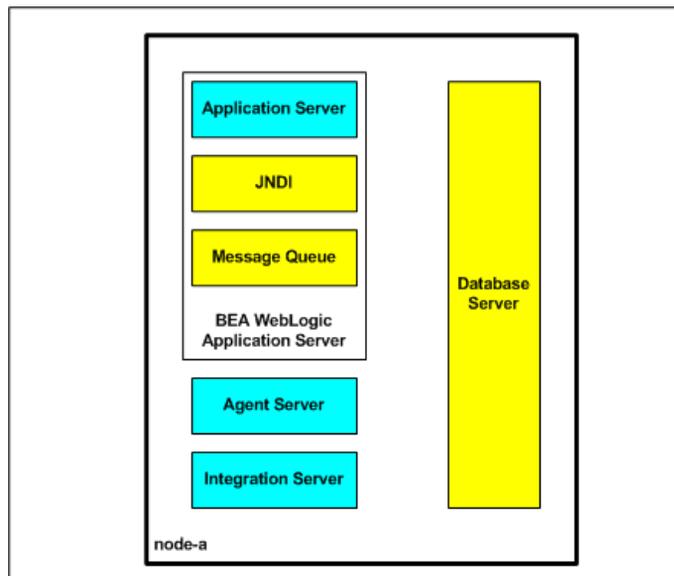
The agent server also uses the JNDI to look for the WebSphere MQ message queue service.

4

Limited Redundancy Single-Site Configuration

We will start the discussion on availability by discussing the attributes of a simple entry-level configuration based on standard off-the-shelf products without additional resources or configuration for availability. It is unlikely that this configuration will be used in production. Its value is as a baseline from where we can build availability into in the subsequent chapters.

Figure 4–1 Limited Redundancy Single-Site Configuration



In [Figure 4–1](#), there is

- a single database instance where application and configuration data are persisted to. The database files are implemented on non-redundant internal drives.
- a single (non-clustered) application server where the JMS message queues are also implemented in. This application server runs on a single JVM.
- a single agent server where all the Sterling Multi-Channel Fulfillment Solution time-triggered transactions are configured to run in
- a single integration server where all integration services run in

As an entry-level configuration, all three servers – the application server, integration server and agent server – run on a single node (node-a) along with the database server.

4.1 Single Point of Failure

As expected, this system has many single-points-of-failure (SPOF) where a single fault can cause a partial or complete Sterling Multi-Channel Fulfillment Solution system or application outage. For example,

- failure of the node (node-a) will cause a complete system outage.
- failure of the Sterling Multi-Channel Fulfillment Solution agent server will affect all services provided by the application server. In addition, it would halt all services running the integration and application servers that depend on the message queues.
- failure of the database instance due to software errors or disk errors will cause a complete system outage since the application is strongly reliant on the data in the database.

The list goes on.

4.2 Loss of Data

These SPOFs will, at a minimum, cause a system outage. Assuming the system is backed up on a regular basis, one should be able to restore services. However, some of these SPOFs in this entry-level system can result in loss of application or business data. We will discuss these error conditions so that you can protect against them in your system.

4.2.1 Loss of Database

First, it should be obvious that losing the disks that the database files are implemented on will result in the loss of the database data. One can easily protect the data by

- ensuring that the database files are stored on external redundant storage. These devices could range from entry-level storage devices (like Dell's PowerVault MD1000) to high end SAN storage devices (like EMC Symmetrix).

4.2.2 Loss of Database Transaction Logs

First, failure of the non-redundant internal disk can result in the loss of transaction data in the database which can result in loss of transaction data. A database management system (like Oracle, DB2 and SQL Server) guarantees the integrity of its data. When a transaction commits its work, the DBMS guarantees that all the changes are either in the database disks or can be recovered from transaction logs.

When a database instance crashes, the DBMS is designed to automatically perform "rollforward or instance recovery". In a nutshell, when you restart the instance, the DBMS will ensure that "committed" changes in its retransaction log are applied to the database files. Similarly, if you had to recover the database from backup, you could also initiate a rollforward recovery from the transaction logs to reapply all the transactions since the backup was taken.

The loss of the transaction logs typically means that at best, the DBMS cannot perform rollforward recovery and at worse, you have to recover the database from the last backup. In either case, this system could lose transaction data.

- never place transaction logs or database files on non-redundant internal disks. These critical files should be placed on redundant storage devices.

4.3 Applicability

In some cases, this simple configuration may be suitable especially if the system meets the customer's availability requirements and represents an

appropriate balance of risk and benefits. As we mentioned earlier, availability design must be driven from a business perspective.

In practice, we rarely see such systems used in production. Instead, the system above is generally used for development, proof-of-concepts, or demonstrations.

If you want to use a similar system in production, consider the following:

- The ability to recover the database. At a minimum, backup the system and database regularly. Also backup your database transaction logs to allow rollforward recovery from the database backups.
- The backup tapes and archived database transaction logs should be stored off-site. This prevents a data center disaster, such as fire, from destroying not only the database server node but also all the backup tapes.

Even with these considerations, this configuration has the following issues:

- Loss of transactions – if you lose the database server and you have to restore the database to a different server, you will lose recent transactions. After a database restore, you have to rollforward or replay all the transactions found in the transaction logs created after that database backup. Typically, the most current active transaction log, in simple configurations, are only saved when the log closes. If you lose the log, you have lost all the recent transactions captured in that log after the database restore.
- Loss of transactions in the integration queues – if you lose the disk on which integration queues are kept, you will lose all the unprocessed transactions in those queues.

5

High Availability Within A Single Site

From a Sterling Multi-Channel Fulfillment Solution system perspective, this chapter identifies:

- All the hardware and software components used by the Sterling Multi-Channel Fulfillment Solution
- All the potential single points of failure (SPOF)
- The approaches to eliminate or reduce the impact of these SPOF

5.1 Single Points of Failure

In the previous chapter, we presented a simple system that was configured with little or no protection against faults. The number of SPOF increases as the system grows in size and complexity. This chapter addresses the following SPOF (within the four walls of the data center) and the means to protect them from faults:

- [Node](#)
- [Database Server](#)
- [SAN or Disk Subsystem](#)
- [Sterling Multi-Channel Fulfillment Solution Components](#)
- [Server Registry](#)
- [Message Queues](#)
- [Networked File Systems \(NFS\)](#)

5.2 Node

The term computer 'node' or computer systems refers to the physical computing hardware on which the Sterling Multi-Channel Fulfillment Solution runs.

Fortunately, due to advancements in hardware design, component redundancy, and automatic fault detection and correction, node failures due to hardware fault are rare events. Take for example memory on an industrial-strength computer. Error Checking and Correcting (ECC) codes are built into the memory to correct single bit errors and to detect double bit errors. If needed, parts of the memory can be selectively disabled. Through techniques such as bit-scattering, memory chips are organized such that failure of an entire memory module only affects a single bit within the ECC word. In addition, with techniques such as bit-steering, bits can be dynamically routed to spare memory chips. [1]

Similarly, nodes typically are configured with multiple critical components such as power and fans so that they can continue to run after one or more components fail. Most of these components are also hot swappable allowing one to replace failed components without the need to shut down the node.

Unfortunately, if the node fails, the mean-time-to-repair (MTTR) could be very high. In the best case, you may only have to restart the node, restart the services, initiate recovery and make the service available. Depending on the size of the configuration, this could take up to 20 minutes or more. In the worst case, for example if the fault was due to a hardware failure, you may have to wait for replacement parts. In those situations, the MTTR could be days.

The impact of a node outage depends on the service that runs on that node. If the node was running a few agent servers, the impact could be isolated to just the services provided by those agents. In contrast, if the outage was in the database server node (and the database is not clustered), the outage will be to the entire application.

If your tolerance for downtime is low, you have the following options

- you could first ensure that your nodes are composed of high redundant servers (as described above) to reduce the likelihood of a node outage caused by hardware faults
- you could use active/passive or primary/standby failover configuration where one or more passive or standby nodes are

available to take over for failed nodes. See [Section 5.2.1, "Active/Passive Cluster Failover Configurations"](#) on page 17 for more information. You can use this approach in subsequent sections to protect critical components such as message queues and the application and agent servers. In [Section 5.3.3.1, "Active/Passive Failover Configurations"](#), we present active/passive configurations for database servers.

- you could use the clustering capabilities built into application servers and in Oracle Real Application Cluster to protect against outage from a single node failure. [Section 5.3.3.2, "Active/Active Failover Configurations"](#) on page 25 describes in the use of an active/active clustered database failover configuration.

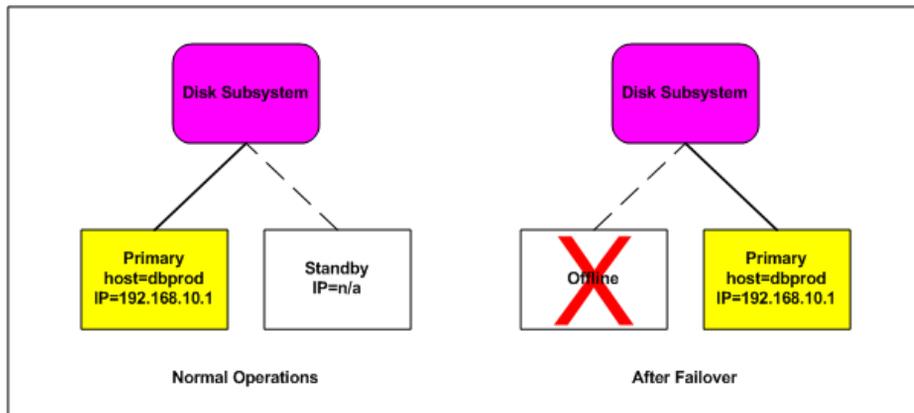
5.2.1 Active/Passive Cluster Failover Configurations

Generally, in an active/passive cluster failover configuration, one or more passive or standby nodes are available to take over for failed nodes. Only the primary node is used for processing. When a node fails, the standby node takes over the resources and the identity of the failed node. The services provided by the failed node are started on the standby node. After the "take over", clients are able to access the services unaware that the services are being provided by a different node.

[Figure 5–1](#) illustrates an active/passive database failover configuration. Both the active/passive nodes share the same disk subsystem although only the primary database server has access to the disk subsystem. The path from the standby node to the shared disk subsystem is not activated.

During normal operations, the application connects to the database server with a hostname of `dbprod` that gets resolved to an IP address of `192.168.10.1`.

Figure 5–1 Active/Passive Database Failover Configuration



During a node failure, the following typically occurs:

On the original primary node:

1. If the primary node is still up, the services on the primary node are brought down.
2. All resources (specifically the disk subsystem) from the primary node are released.
3. The service IP address (192.168.10.1) is released.

On the standby node:

1. The disk subsystem is brought online.
2. File systems are checked and repairs are made if needed.
3. The service IP address (192.168.10.1) is configured.
4. The services are started – database rollforward recovery is initiated as necessary.
5. The database services are opened.

These failover or takeover steps can be automated. Some of the software that can be used include:

- IBM HACMP (only available on AIX)
- Veritas Cluster Service (VCS)

- HP MC/ServiceGuard
- Microsoft Cluster Server (MCS)

Fully automated, the failover could take 5 to 10 minutes.

In subsequent sections, we present the use of active/passive failover configurations to protect many of the Sterling Multi-Channel Fulfillment Solution components in more detail.

5.3 Database Server

The database server is a critical system component. The entire system is unavailable if the database server crashes. There are many reasons why the database server can come down, including:

- [DBMS Software Failures](#)
- [Human and Operator Errors](#)
- [Hardware Failures](#)

5.3.1 DBMS Software Failures

As with any large complex software, there are bugs in the Oracle and UDB database servers. Some of these bugs can cause instance crash or performance degradation. In rare extreme cases, these bugs can corrupt the database.

The best means to protect against software failures is testing. Your testing must exercise transactions from a broad range of application functionality and not a small subset of transactions. The tests must also run at transaction volumes at or higher than anticipated peak production periods. These tests are the only reliable means for identifying load, concurrency, or integrity issues in the database management system and the application.

You should be aware of any support or service alerts associated with or new issues introduced with your database server release. The list of issues is not static – new bugs are discovered as customers use the release, existing bugs are be fixed, and so forth. Therefore, you should check this list periodically to see if there are any new issues that could potentially affect your system.

Additionally, you should be careful that you don't apply all the patches available for that database release. From our experience, you may destabilize a database release when you apply too many individual patches. In some cases, individual patches may conflict with each other.

For software bugs that crash the instance, the fastest recourse is to restart the instance. For a corrupted database, your recourse may range from trying to repair the damage using SQL to restore the database from the last backup and performing rollforward recovery until the point before the corruption. Either way, the MTTR is likely to be very high.

5.3.2 Human and Operator Errors

A Gartner report "shows that an average of 80 percent of mission-critical application service downtime is directly caused by people or process failures. The other 20 percent is caused by technology failure, environmental failure or a disaster." [2]

The best prevention is strict change control, documented procedures, training, and supervision.

Recovery from human-induced outages could range from restarting services to recovering a corrupted database.

5.3.3 Hardware Failures

Node failures are extremely rare. Unfortunately, when they do occur, the MTTR can be unacceptably high for your business.

To protect the database server from node failures, you can use either active/passive or an active/active cluster failover configurations.

5.3.3.1 Active/Passive Failover Configurations

5.3.3.1.1 IBM UDB Active/Passive Using Cluster Failover Software

Conceptually, UDB active/passive failover configurations using cluster failover software operates as described in [Section 5.2.1, "Active/Passive Cluster Failover Configurations"](#) on page 17. The standby node takes over the primary node's resources (the database files, logs) and identity (IP address, SAN WWNN). The database service is then started on the standby node. During the startup, UDB goes through its normal crash

recovery and ensures committed changes are made to the database and incomplete transactions are rolled back. When UDB is finished with crash recovery, the database service is made available.

From the Sterling Multi-Channel Fulfillment Solution perspective, you can expect the following to occur after the primary node fails (and the database server is unavailable).

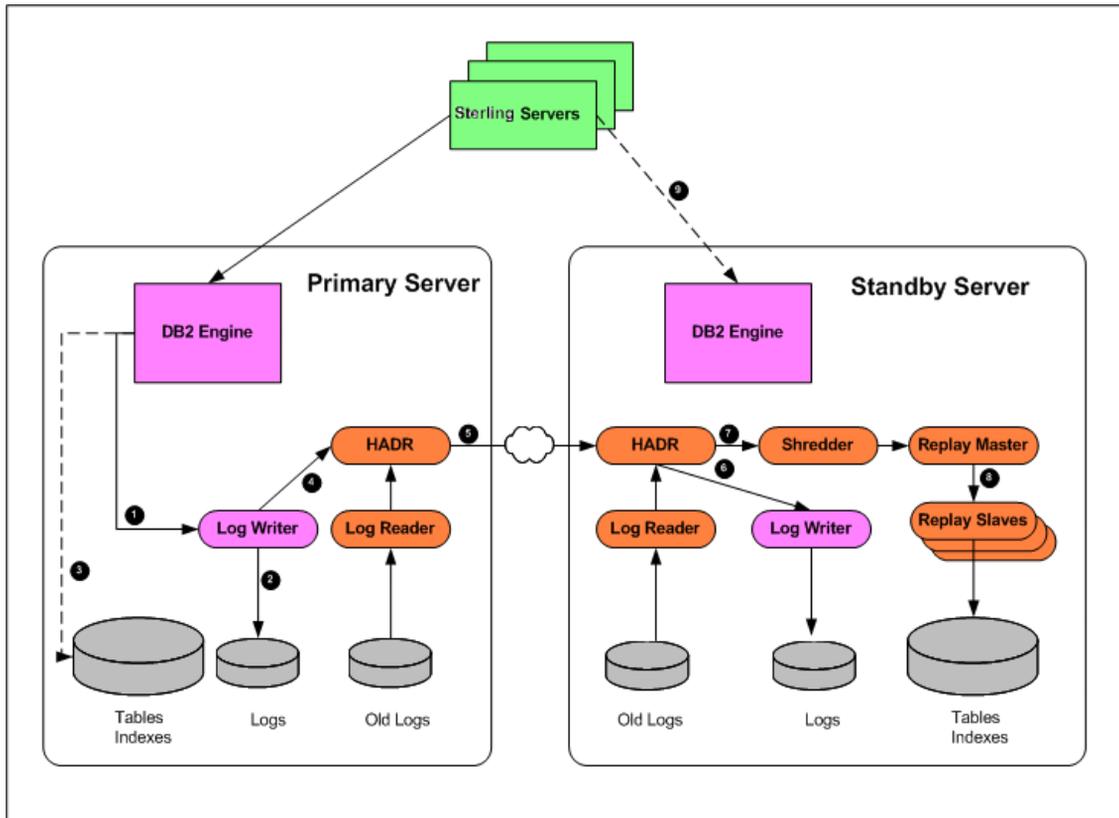
- Transactions in the application, agent and integration servers that were actively processing throw a SQL error message. The changes from those transactions are correctly rolled back later when the database server comes up on the standby node.
- The Sterling Multi-Channel Fulfillment Solution servers continually reissue the transactions until the database service is restored. You do not have to restart the Sterling Multi-Channel Fulfillment Solution servers during the transition to the standby node.
- If the source of the work request (specifically for the agents and integration servers) came from message queues, the messages remain in the message queue. When the database service is restored, these messages are processed.

Setting up and testing an active/passive failover configuration can be tricky with many interdependencies and related parts. We strongly encourage you to contact the cluster failover vendors for assistance in planning and implementing your cluster failover.

5.3.3.1.2 IBM UDB Active/Passive Using HADR

High Availability Disaster Recovery (HADR) is a transaction log replication approach that keeps a standby database server in or near synch with changes in the primary database server. In the event of a failover, HADR on the standby database server takes over and becomes the primary as described in [Figure 5–2](#).

Figure 5–2 UDB Active/Passive Using HADR



At a high level, the log writer on the primary database server records changes to its local transaction logs. These logs are critical for crash and instance recovery. The primary HADR sends the log records to the standby HADR where the logs are written out to the standby server's transaction logs. The changes are then applied to the standby server's database. At some point in time, the changes on the primary server are asynchronously written to the database.

The standby database server is kept in "perpetual rollforward" mode applying transaction log entries as they are replicated from the primary.

HADR provides many benefits over the traditional active/passive cluster failover provided by software such as HACMP, VCS or MC/ServiceGuard.

First HADR recovery is faster because you do not have to start the standby database server – the standby database server is always running and is either in or near synch with the primary database server. Similarly, you do not have to spend a lot of time in database crash recovery because by design, the standby HADR database server is in or near peer state. Also, you do not have to spend time releasing resources on the failed node and acquiring resources on the standby node. With HADR, the standby database is already connected to and using a separate disk subsystem.

Second, the standby HADR database server does not share disk subsystems with the primary database server. Therefore, with HADR, you can survive a disk subsystem failure whereas cluster failover, which relies on a shared SAN, could incur a potentially prolonged outage until the disk subsystem is repaired.

HADR is provided as part of ESE. With HACMP or the other cluster failover software, you have to purchase additional software licenses.

From a recovery perspective, the HADR provides a less risky failover approach. With HADR, the standby database is already running. In contrast, with cluster failover, resources have to be acquired, services started, etc. There are potential startup risks during the recovery process.

UDB 9.1's HADR implementation has the following limitations:

- HADR can only replicate to one standby database server – therefore, from the primary database server, you can either HADR to a local standby for local site failover or to a remote site for disaster recover (but not both).
- HADR is only supported on UDB ESE
- You cannot backup from the standby – you must backup from the primary

Please refer to the IBM UDB documentation for more detail.

Client Reroute

Client Reroute was introduced in UDB 8.2 along with HADR to enable client applications to automatically reconnect to the standby HADR database server when the primary server fails. Client reroute works by informing the client of the alternate or standby database when it connects to the primary.

The alternate database information is defined on the primary database server with the following command:

```
db2 update alternate server for database <dbname> using  
hostname <hhh> port <ppp>
```

For example, if your primary database DB2PROD is on node N1 port 50000 and the alternate is on node N2 port 50000, issue the following command on node N1:

```
db2 update alternate server for database DB2PROD using  
hostname N2 port 50000
```

Alternates are propagated from the server to the client dynamically when the client issues a CONNECT or CONNECT RESET. This dynamically propagated alternate server information is stored in global driver memory, and is also updated in the JNDI store of DB2 active servers.

Initially, DB2 Universal JDBC Driver client reroute support was available only for connections that use the `javax.sql.DataSource` interface. In DB2 9.1 FP3, IBM added client reroute support to `java.sql.DriverManager`.

5.3.3.1.3 Oracle Active/Passive Using Cluster Failover

Oracle active/passive failover configurations are very similar to the UDB active/passive configuration described in [Section 5.3.3.1.1, "IBM UDB Active/Passive Using Cluster Failover Software"](#) on page 20. Given the popularity of Oracle Real Application Cluster (RAC), it is our belief that customers are trending towards implementing active/active Oracle failover configuration instead. This is described in [Section 5.3.3.2, "Active/Active Failover Configurations"](#) on page 25.

5.3.3.1.4 Microsoft SQL Server Active/Passive Using MSCS

Microsoft SQL Server active/passive failover configurations are very similar to the [Section 5.3.3.1.2, "IBM UDB Active/Passive Using HADR"](#) on page 21.

The clustered nodes use the heartbeat to check whether each node is alive, at both the operating system and SQL Server level. At the operating system level, the nodes in the cluster compete for the resources of the cluster. The primary node reserves the resource every 3 seconds, and the competing node every 5 seconds. The process lasts for 25 seconds and then starts over again. For example, if the node owning

the instance fails due to a problem (network, disk, and so on), at second 19. The competing node detects it at the 20-second mark, and if it is determined that the primary node no longer has control, the competing node takes over the resource.

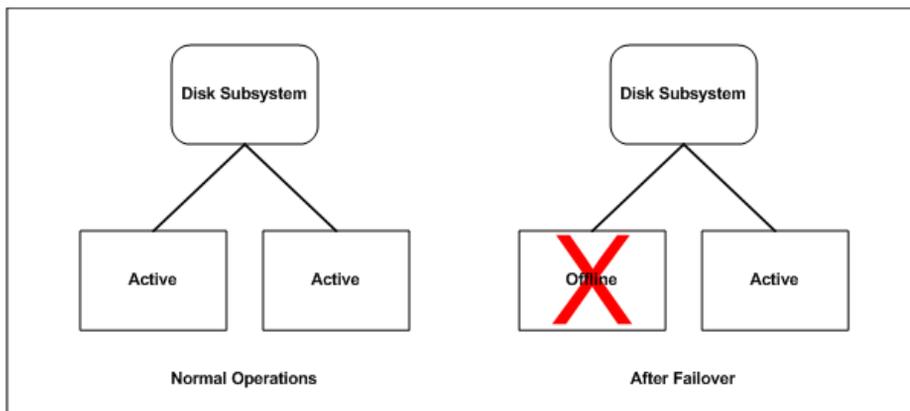
From a SQL Server perspective, the node hosting the SQL Server resource does a looks-alive check every 5 seconds. This is a lightweight check to see whether the service is running and may succeed even if the instance of SQL Server is not operational. The IsAlive check is more thorough and involves running a `SELECT @SERVERNAME Transact SQL` query against the server to determine whether the server itself is available to respond to requests; it does not guarantee that the databases are up. If this query fails, the IsAlive check retries five times and then attempts to reconnect to the instance of SQL Server. If all five retries fail, the SQL Server resource fails. Depending on the failover threshold configuration of the SQL Server resource, Windows Clustering attempts to either restart the resource on the same node or fail over to another available node. The execution of the query tolerates a few errors, such as licensing issues or having a paused instance of SQL Server, but ultimately fails if its threshold is exceeded.

During the fail over from one node to another, Windows clustering starts the SQL Server service for that instance on the new node, and goes through the recovery process to start the databases. The fail over of the SQL Server virtual server takes a short time (probably seconds). After the service is started and the master database is online, the SQL Server resource is considered to be up. Now the user databases go through the normal recovery process, which means that any completed transactions in the transaction log are rolled forward, and any incomplete transactions are rolled back. The length of the recovery process depends on how much activity must be rolled forward or rolled back upon startup. Set the recovery interval of the server to a low number to avoid long recovery times and to speed up the failover process.

5.3.3.2 Active/Active Failover Configurations

In an active/active failover configuration, two or more database nodes are clustered together to provide a common service to applications. Unlike active/passive failover configurations, all nodes of an active/active failover configuration are actively processing transactions. If a node fails, the remaining nodes continue to provide the service. [Figure 5-3](#) illustrates this concept:

Figure 5–3 Active/Active Failover Configuration



5.3.3.2.1 Oracle RAC Active/Active

The Oracle Real Application Cluster (RAC) is a share-everything database cluster with a distributed lock manager. As a share-everything database, all RAC nodes access and update the same database data files. The distributed lock manager controls which node updates the data. It does not matter which node the transaction is performed on. Each node has equal rights to all data in the shared database.

Each RAC node has a listener process that is responsible for processing database connection requests from client programs. When the listener receives a request, it could spawn off a new database process to which the client program connects to. If server-side load balancing is enabled, the listener could send the request to the listener on the least busy RAC node.

Configuration

When configuring the Sterling Multi-Channel Fulfillment Solution with Oracle RAC, you want the RAC nodes to be reasonably balanced so that all RAC nodes, over a period of time, are about the same utilization. During a node failure, you also want the connections from the failed node to automatically reconnect to the surviving RAC node. You can do this using Oracle features on the client-side and server-side.

Client-Side Load Balancing

On the client-side, set up the JDBCURL parameter, which the JDBC driver uses to connect to Oracle, as follows:

```
"jdbc:oracle:thin:@
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE = yes)
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode02)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = rac)
    )
  )
```

In this example, the JDBCURL shows two RAC nodes (dbnode01 and dbnode02). The (load_balance=yes) instructs the JDBC Driver to progress through the list of addresses in a random sequence spreading the load to all the listeners.

In a failure situation, if the connection request goes to the “downed” node, the connection requests timeouts. The driver then transparently sends the connection request to the remaining node. The connection timeout can be tuned down.

With client-side load balancing, the connection request eventually reaches an Oracle listener.

Server-Side Load Balancing

On the RAC nodes, enable server-side load balancing so that the listener routes database connections to the RAC instance on the least busy (loaded) nodes.

The listener on each RAC node is aware of all RAC instances. In addition, the Oracle PMON (process monitor) periodically updates the listeners with the node utilization. The update can occur as quickly as a minute on heavily loaded nodes or as much as ten minutes on lightly loaded nodes. Depending on the load information, the listener decides to which instance to send the incoming client request. The listener typically selects an instance on the least loaded (busy) node. If the least busy node has multiple RAC instances, the listener then chooses the least loaded instance on that node.

You can enable server-side load balancing by setting the following parameters:

| hostname | service name | sid name | instance_name |
|----------|--------------|----------|---------------|
| dbnode01 | rac | rac1 | rac1 |
| dbnode02 | rac | rac2 | rac2 |

```
spfile
*.remote_listener='LISTENERS_rac'
rac1.local_listener='LISTENER_rac1'
rac2.local_listener='LISTENER_rac2'

*.db_name='rac'

rac1.instance_name='rac1'
rac2.instance_name='rac2'
```

dbnode01 listener.ora file

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /u01/app/oracle/product/9.2.0.5)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (ORACLE_HOME = /u01/app/oracle/product/9.2.0.5)
      (SID_NAME = rac1)
    )
  )
)
```

dbnode01 and dbnode02 tnsnames.ora file

```

LISTENERS_RAC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode02)(PORT = 1521))
    )
  )
)
RAC1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = rac)
      (INSTANCE_NAME = rac1)
    )
  )
)
RAC2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode02)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = rac)
      (INSTANCE_NAME = rac2)
    )
  )
)
RAC =
  (DESCRIPTION =
    (LOAD_BALANCE = yes)
    (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode02)(PORT = 1521))
    (CONNECT_DATA =
      (SERVICE_NAME = rac)
    )
  )
)

```

```
dbnode01 only tnsnames.ora

LISTENER_rac1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode01)(PORT = 1521))
  )
```

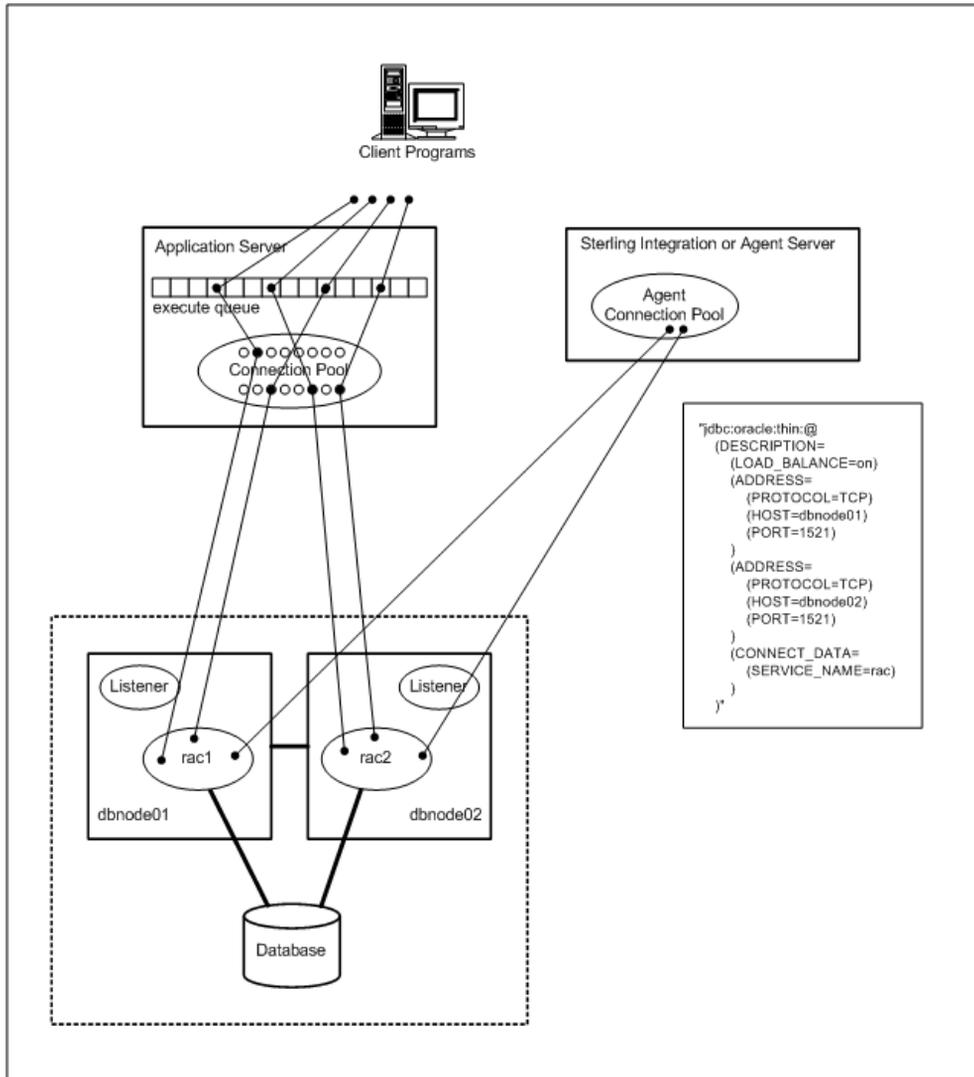
```
dbnode02 only tnsnames.ora

LISTENER_rac2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = dbnode02)(PORT = 1521))
  )
```

LISTENERS_RAC, LISTENER_rac1, and LISTENER_rac2, are the net_service_name (connect descriptor) for remote_listener and local_listener. On the client side, you do not need these net_service_name parameters.

RAC is the net_service_name for client-side load balancing as illustrated in [Figure 5-4](#).

Figure 5-4 Load Balancing



From a Sterling Multi-Channel Fulfillment Solution perspective, you can expect the following to occur after a RAC server instance failure:

- Transactions in the application, agent and integration servers that were actively processing throws a SQL error message. The remaining RAC nodes roll the changes from those transactions back.
- The Sterling Multi-Channel Fulfillment Solution agent and integration servers continually attempt to reconnect. When it has connected to one of the remaining RAC instances, the failed transactions are reprocessed from the beginning. You do not have to restart the Sterling Multi-Channel Fulfillment Solution servers during the transition to the standby node.
- If the source of the work request (specifically for the agents and integration servers) came from message queues, the messages remain in the message queue. When the database service is restored, these messages are processed.

Target Utilization

When deploying an active/active failover configuration like Oracle RAC, take into consideration your target node utilization, especially during the seasonal peak periods. Although RAC is an active/active configuration, you have to be careful when you run the RAC nodes at close to max utilization. First, RAC needs CPU to maintain the Cache Fusion. Secondly, during failover, all the work from the failed node eventually spills over to the remaining node. For example, at the peak hour, the combination of high volume agent processing and application server processing is driving a 2-node RAC configuration to 80% busy. If a node fails, the servers reconnect to the remaining node. The combined workloads drive the database to a point where performance degrades or at worst the system becomes unstable.

Ideally, you should try to keep the average node utilization below 40% utilization (for a 2-node RAC) to reduce the risk of overloading the remaining nodes. In the event of a node failure, the remaining node will likely grow to 80%.

Alternatively, if you typically run the nodes at higher utilization, you should consider classifying workloads into priority groups. For example, transactions that are initiated by customers may be put into the high priority group because they add to the overall customer experience. These could include application server workloads. Customers with short “click to release” service levels may consider agents that schedule and release orders as high priority workloads. Workloads such as order purge

or other maintenance work could be categorized into lower priority groups.

5.3.3.2.2 IBM DB2 Active/Active Using UDB ESE DPF

The Sterling Multi-Channel Fulfillment Solution is not certified to run on UDB DPF.

5.3.3.2.3 IBM DB2 Active/Active Using xkoto GRIDSCALE

The Sterling Multi-Channel Fulfillment Solution is not certified to run on xkoto GRIDSCALE.

5.4 SAN or Disk Subsystem

The disk subsystem is another critical system component. Disk failures could cause outage to parts of or the entire system.

Careful design and implementation must be placed on the disk subsystem. Your investment in failover configurations could be wasted if the disk subsystem fails.

The different areas where disks are used by the Sterling Multi-Channel Fulfillment Solution include:

- Database datafiles/objects
- Message queues
- Internal disks

To prevent outages due to disk failures, consider configuring the following:

- Disks as RAID groups (except for RAID-0) for redundancy and performance. We prefer RAID 10 or RAID 5 for performance and redundancy.
- Redundant disk systems to tolerate component failures
- Multiple access paths to the disks

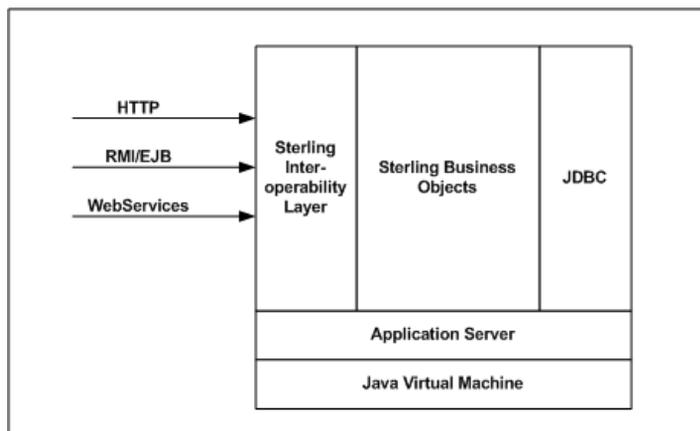
5.5 Sterling Multi-Channel Fulfillment Solution Components

As we described in [Chapter 5, "High Availability Within A Single Site"](#), the Sterling Multi-Channel Fulfillment Solution runs in the application server, agent server, and the integration server.

5.5.1 Application Server

The application server provides:

- Users with the ability to use the Sterling Multi-Channel Fulfillment Solution Consoles to create and manage orders, shipments, and inventory manage orders, shipments, and inventory
- Programs to call the Sterling Multi-Channel Fulfillment Solution business APIs using HTTP, servlets, EJB/RMI, or WebServices



Currently, the Sterling Multi-Channel Fulfillment Solution is supported on the following application servers:

- BEA WebLogic
- IBM WebSphere
- JBoss Application Server

All application server requests are transactional in that all the work performed is within a transaction boundary and is either fully committed

or nothing is committed. This guarantees that there is no partially completed work.

For resiliency, you can configure multiple application server instances in a cluster. If an instance fails, the workloads are sent to the remaining instances.

Generally, the following occurs when an application server instance fails. If there were active transactions running on the failed application server, those transactions can be reprocessed. The transactions are either fully committed or not at all. Subsequent transactions are sent to the remaining application server instances.

5.5.1.1 Stateful Sessions

The HTTP user interface (or the Sterling Multi-Channel Fulfillment Solution Consoles) sessions are “stateful”. When transactions complete, they leave information (state) on the J2EE application server for the subsequent transaction. By being stateful, all HTTP transactions return to the same application server instance. This can be accomplished by telling the load-balancers or proxies that the sessions are sticky.

By default, WebLogic and WebSphere application servers are configured for memory session persistence where the HTTP session-state is only stored in the application server instance that the transaction ran on. The session information is lost if that application server instance fails. If that happens, the user is redirected to the login page. After logging back into the application, the user is able to continue where they left off.

5.5.1.2 Stateless Sessions

All the other application server transactions, EJB/RMI, servlet calls, and WebServices, are stateless. In contrast, these transactions do not leave behind session state information. As a result, these transactions can be performed on any application server instance where the application is deployed.

5.5.2 Sterling Multi-Channel Fulfillment Solution Agent and Integration Server

The Sterling Multi-Channel Fulfillment Solution agent and integration servers are completely location or node independent. They can run from any node where the application has been deployed.

5.6 Server Registry

The Sterling Multi-Channel Fulfillment Solution maintains information on how to locate each of the server instances (e.g., the application server, agent and integration server instances) for system management purposes. For example, a server that changes cached reference data must notify its peers to update their cache. See the *Sterling Multi-Channel Fulfillment Solution Performance Management Guide* for more details.

When a server instances starts, it stores its server name and the URL to itself in a registry. In previous releases, the registry was implemented in a non-clustered Java Naming and Directory Interface (JNDI) service in one application server instance. As a result, in previous releases, this registry was a single point of failure. In the Sterling Multi-Channel Fulfillment Solution, Release 8.0, the registry was moved to the YFS_HEARTBEAT database table.

Any server instance can query the YFS_HEARTBEAT to find all the other server instances.

The Sterling Multi-Channel Fulfillment Solution uses the JNDI information for the following events:

- Reference data cache refresh – The Sterling Multi-Channel Fulfillment Solution implements a mid-tier data cache to cache commonly used reference data. If a server instance changes a cacheable record, that instance needs to broadcast that change to instruct all the other server instances to refresh their cache. For more information on the Sterling Multi-Channel Fulfillment Solution reference data cache implementation, refer to the *Sterling Multi-Channel Fulfillment Solution Performance Management Guide*.
- The system management console uses the registry to discover all the application instances. The SMC uses that list to build a list of instances to monitor.

In addition to the initial registration on start up, all servers have to periodically update its registry record to indicate that it is still alive.

In past releases, if the JNDI service was unavailable, you would not be able to start servers or notify peers of system management changes. In this release, the registry's availability posture is the same as the

database service. If the database service is down, the application would be down therefore, there would be no need for the registry.

5.7 Message Queues

The Sterling Multi-Channel Fulfillment Solution uses message queues extensively. The message queue usage in the Sterling Multi-Channel Fulfillment Solution can be grouped as:

- Integration queues for external communications
- Temporary work-in-progress queues for the Sterling Multi-Channel Fulfillment Solution and custom time-triggered transactions

These queues can be implemented in:

- BEA WebLogic JMS
- IBM WebSphere MQ
- IBM WebSphere default messaging providers
- JBoss MQ

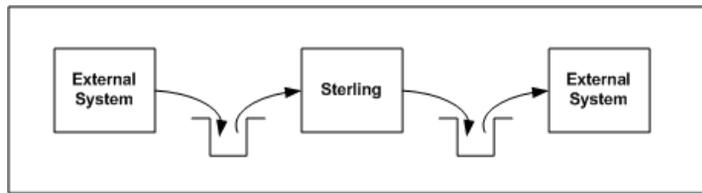
The Sterling Multi-Channel Fulfillment Solution agent and integration servers use messaging primarily for two reasons:

- Integration queues
- Agent work queues

5.7.1 Integration Queues for the Sterling Multi-Channel Fulfillment Solution Integration Servers

The integration servers use integration-based queues to communicate from or to external systems. For example, in [Figure 5–5](#), the first queue could be used by external systems like a Web store frontend to pass order creation requests to the Sterling Multi-Channel Fulfillment Solution. It could also be purchase orders from a purchasing system, shipment status updates from a logistics management system, and so forth.

Figure 5–5 Integration-based Queues



Similarly, the Sterling Multi-Channel Fulfillment Solution integration servers can use integration queues to send messages to external systems. For example, the Sterling Multi-Channel Fulfillment Solution can send ship notices to warehouses.

Since these messages are used for communicating between systems, the messages in the integration queues should be protected so that they are not lost in the event of a failure. In some cases, these messages can be difficult to recreate. For example, as described above, if messages from a web-store is lost, the web-store will have to resend the missing orders. The recovery will involve having to determine what orders have already been processed to find what is missing. Care will have to be taken to ensure that orders that have already been processed are not resent.

In general, integration queues should be implemented on reliable redundant persistent stores.

5.7.2 Agent Queues for the Sterling Multi-Channel Fulfillment Solution Agent Servers

The Sterling Multi-Channel Fulfillment Solution agent servers use the messages in the queues as a source of work. In contrast to integration queue messages, agent messages are typically read from the database and can be easily recreated. As a result, we recommend creating agent queues as non-persistent queues where the messages are kept in memory.

5.7.3 WebLogic JMS

To protect the WebLogic JMS queues, consider the following:

- [Persistence](#)
- [Message Paging](#)

- [Dedicated Integration JMS Server](#)
- [Shared Disk Subsystem](#)
- [JMS Server on a Different Application Server](#)

5.7.3.1 Persistence

By default, WebLogic JMS message queues are defined as non-persistent, which means that the messages are only kept in memory (the JVM heap). Non-persistent messages are lost when the JMS server shuts down or crashes.

To protect integration messages, you must define the integration queues as persistent. WebLogic allows messages to be persisted to files or the database.

The agent queues should be implemented as non-persistent queues. The Sterling Multi-Channel Fulfillment Solution agents are designed to be able to re-create the work-in-progress task messages.

5.7.3.2 Message Paging

By default, the messages in WebLogic JMS queues are kept in memory (in the Java heap). You should consider protecting them against situations where a large number of messages could cause the JMS server to fail because it has run out of space in its Java heap. This could happen if there are a significantly large number of messages in the queue or exceptionally large message bodies. In those situations, the JVM could run out of space in its heap.

The best way to find out how much JVM heap you need is to create a large number of messages in your test queues and see how much memory is used. The amount of heap required differs for each implementation.

To protect against situations where the WebLogic JMS heap fills up, you could enable byte or message paging. When the number of bytes or messages exceeds specified thresholds, WebLogic JMS server pages out the message body (but not the message header) to a paging file store. This approach can reduce the likelihood of a JVM crash. You can still run out of JMS Server heap if the queue has a lot of message headers.

As a draconian measure, you can set the maximum message parameter. When set, this parameter puts a hard limit on the number of messages

that can be in the queue. When this threshold is reached, new messages are rejected with an error message. As a result, you have to ensure that message producers are architected to correctly handle message maximum exceptions – for example, the message producers may want to queue the messages and retry later.

5.7.3.3 Dedicated Integration JMS Server

Since integration queues can grow unbounded, you should define the integration queues to one or more dedicated JMS servers. These JMS servers should be targeted to one or more dedicated WebLogic managed servers. For example, assume you have ten integration queues. Some of your options are:

- Define all ten integration queues to a single JMS server which is targeted to a dedicated application server. That application only has the JMS server targeted to it.
- For very large integration queues, target 5 integration queues to one JMS server and the other five to another JMS server. The two JMS servers are then targeted to two dedicated application servers.

5.7.3.4 Shared Disk Subsystem

The persistence and paging file stores should be implemented on a shared disk subsystem (such as a SAN) and not on local disks. The shared disk subsystem should be accessible from standby servers to prevent disk failures or node outages from causing a prolonged JMS Server outage. This configuration allows you to restart the JMS Server from another node.

5.7.3.5 JMS Server on a Different Application Server

You can protect the task-based queues by ensuring the JMS server can be restarted on a different node or a different application server. You don't have to worry about preserving the content of the queues because they can be recreated from the database. You also don't have to worry about protecting them against large number of messages because the agents only fetch a finite number of messages (default is 5,000).

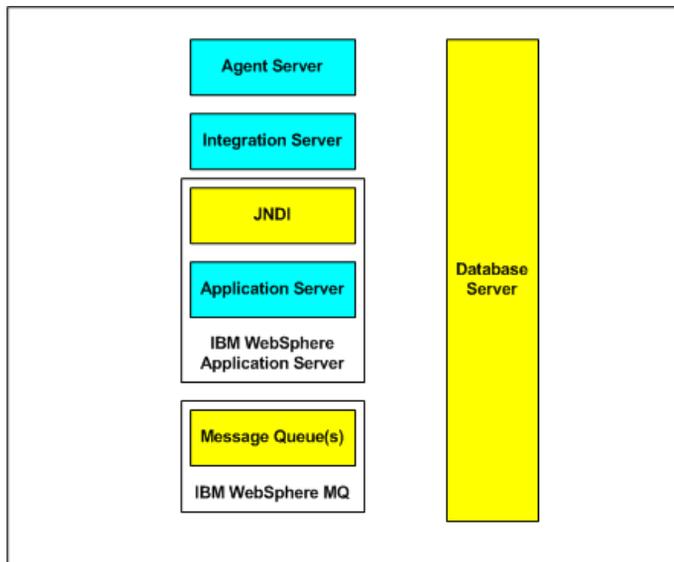
5.7.4 WebSphere Messaging

5.7.4.1 WebSphere MQ

For WebSphere customers, as depicted in [Figure 5–6](#), the integration and agent queues can be implemented in

- WebSphere MQ, which is an external message queue, or
- WebSphere default messaging which is the messaging component inside the WebSphere Application Server.

Figure 5–6 WebSphere MQ

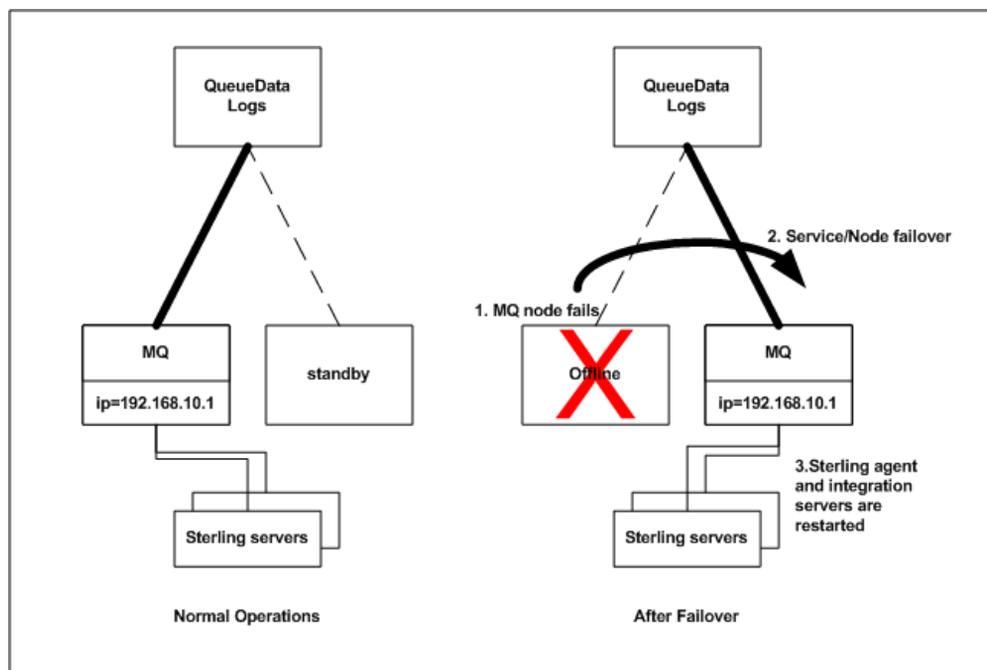


5.7.4.1.1 Protecting WebSphere MQ HA Using Cluster Failover

One approach to protecting WebSphere MQ is through cluster failover using cluster software such as HACMP, MC ServiceGuard, or Veritas Cluster Service. In [Figure 5–7](#), the MQ (the MQ queue manager and the local queues) on the active node with IP address of 192.168.10.1 is running and accepting and distributing MQ messages. In the event of a node failure and the node cannot be restarted, the cluster software activates the standby node. From the agent and integration servers' perspective, the MQ service was unavailable during the time it took to

failover. Once the failover is completed, the agent and integration servers can be restarted. At that time, the standby node looks the same as the former primary node. Therefore, there are no changes required.

Figure 5–7 Cluster Failover



5.7.4.1.2 Implementing Message Persistence Files on SAN

To prevent disk failures or node outages from causing a prolonged MQSeries outage, you should consider putting the MQ logs and files used to store messages on a SAN that is accessible from multiple nodes. This allows you to restart MQ from another node.

5.7.4.1.3 Risks

In the previous examples, the shared storage remains the single point of failure. You could lose data if the shared storage was. One option is to configure active/passive SAN devices with the primary SAN replicating data to the standby SAN. In the event of a SAN failure, the primary MQ node would release the primary SAN and acquire the standby SAN.

5.7.4.1.4 Important Notes

- Configuring MQ in an active/standby configuration with HACMP requires specialized knowledge. Please consult your IBM representative for assistance.
- Sterling Commerce has not tested or certified the Sterling Multi-Channel Fulfillment Solution on clustered MQ.

5.8 Networked File Systems (NFS)

Some customers prefer to implement the Sterling Multi-Channel Fulfillment Solution on a networked file system (NFS). With this approach, all mounted nodes can access all the shared folders. Changes are made to the shared files. In contrast, changes do not have to be pushed out to every node if files were stored on local disks.

If you choose this approach, consider implementing a highly available NFS to prevent an outage of the NFS server from creating an application outage. Losing the NFS server crashes all the servers of the Sterling Multi-Channel Fulfillment Solution.

6

Architectural Patterns

The Sterling Distributed Order Management (DOM) and Sterling Warehouse Management System (WMS) applications are often deployed in an integrated network of external systems to form a cohesive business ecosystem. Prolonged application or system outages can have significant business consequences.

Decoupling and component independence is an extremely powerful architectural pattern to insulate critical portions of the overall ecosystem solution from downtime or faults in other areas. The availability and uptime of the Sterling Multi-Channel Fulfillment Solution-based solution can be greatly enhanced by adopting one or more of the following patterns during solution design. Each of these patterns makes it possible to decouple one or more parts of the application from other portions thus providing increased availability and uptime for critical areas like external users and customers.

Each of these design patterns can be applied to provide increased application resilience. While these examples talk about website integration, these patterns can be applied to other areas of integration as well.

A well-designed solution around the Sterling Multi-Channel Fulfillment Solution system can actually increase the availability and uptime of the solution as a whole to levels above what the Sterling Multi-Channel Fulfillment Solution delivers out-of-the-box. In some critical areas for example, the solution can continue to be available even when the product is taking a planned or unplanned outage.

Finally, there are a few other process and deployment related solution design considerations that can actually provide better gains in availability and uptime at a much lower cost than technological and redundancy based solutions.

6.1 Asynchronous Integration

The most common decoupling technique is asynchronous message communication between business entities. Take, for example, the need to send orders created at different external systems to the Sterling Multi-Channel Fulfillment Solution. These systems could send the order creation requests to the Sterling Multi-Channel Fulfillment Solution:

- Synchronously using protocols such as HTTP, WebServices, EJB/RMI and so forth; or
- Asynchronously using messages.

Both approaches have their strengths and weaknesses. From a high-availability standpoint, the loosely-coupled asynchronous approach allows the Sterling Multi-Channel Fulfillment Solution to be unavailable as a result of a scheduled or unscheduled outage, without affecting the external systems. The external systems can queue up the order creation requests into an integration message queue.

In contrast, if the communication is synchronous, the Sterling Multi-Channel Fulfillment Solution must be available for the external system to create the request. In this architecture, the availability requirements of the Sterling Multi-Channel Fulfillment Solution have to be the greatest of all the availability requirements of all tightly connected systems combined.

This scenario is, of course, simplistic since it may not take into account other synchronous interfaces (like inventory lookups or pricing) that the order creation process is dependent upon. These requirements are addressed in [Section 6.2, "Caching"](#).

6.2 Caching

Another common decoupling technique is the use of local caching. In this pattern, the consuming application (for example, store web store) gets information such as item attributes, inventory balance, or item availability from a local data cache. This approach reduces the need to synchronously query the Sterling Multi-Channel Fulfillment Solution.

The local information cache can be updated by utilizing a variety of algorithms that offer various degrees of sophistication, performance and accuracy. Not only does this technique provide a way to decouple two areas of the solution, but it also provides significant performance,

response time, and scalability advantages that are especially useful in end-user or website scenarios.

As an example, one area where the Sterling Multi-Channel Fulfillment Solution typically recommends utilizing this algorithm is for caching ATP (Available to Promise) data on the Web site. In some customer environments where shopping cart abandonment rates are very high, for example 100 item lookups to 1 item ordered, it is better to have the Sterling Multi-Channel Fulfillment Solution push out item availability to the Web storefront using the Sterling Multi-Channel Fulfillment Solution Real-time Inventory Monitor. With this approach, most inventory lookups that are part of the customer's browsing and ordering experience can be served from the Web site without any synchronous calls to the Sterling Multi-Channel Fulfillment Solution. Based on business requirements, if the inventory levels are sufficiently high, the web storefront can sell that item. The Web storefront would revert to synchronous inventory availability check when the inventory levels are below a certain threshold. More importantly, the Sterling Multi-Channel Fulfillment Solution can be down without affecting the Web storefront.

While this cookie cutter approach to inventory caching may not work for all scenarios, techniques such as these can be invariably applied to almost all critical interfaces to provide simplistic but "safe" algorithms to counter planned or unplanned downtime without affecting end users or disabling critical functionality areas altogether.

6.3 Hot Deployment of Code, Configuration, and Fixes

While the methodologies and design patterns presented insulate critical areas of the solution from downtime, there are deployment techniques that are provided by Sterling Commerce or are inherent within the architecture of the Sterling Multi-Channel Fulfillment Solution that allow you to hot deploy incremental changes, configuration, and fixes on critical synchronous application components. Some of these capabilities include the ability to:

- Deploy changes to incremental configuration or master data without having to bring down any application areas.
- Hot deploy incremental software or code changes on the synchronous application components by utilizing capabilities offered by the

application server or by utilizing application server independent techniques like clusters and rolling restarts.

Theoretically, there could be scenarios where even a small change to a component may require multiple areas of the application to be updated simultaneously due to interdependencies, thus causing an outage. However, in reality, a large number of incremental changes and product fixes can utilize these techniques even without any explicit hot deployment design considerations.

Explicitly factoring in requirements to be able to hot deploy changes during incremental solution design phases leads to the ability to hot deploy all changes with a few exceptions. This situation is further mitigated by the dependence of the Sterling Multi-Channel Fulfillment Solution on asynchronous processing for complex algorithms. This significantly reduces the solution footprint that external synchronous interfaces like those from the website, rely on. This, in turn, reduces the probability of many changes or fixes in these areas.

6.4 Deployment Processes and Regression Testing

One of the most important and most overlooked areas that can significantly affect availability and uptime of an application is the presence of a strictly enforced process to promote, characterize, verify, and regression test incremental rollouts or hot-fixes and upgrades. In industry studies and based on Sterling Commerce's experience, the lack of sufficient automated integration testing, human and operator error, and lack of appropriate software change management processes to prevent those errors, is the single biggest factor that causes application downtime when there is no actual infrastructure failure. The cost of setting up and investing in a robust and isolated testing environment that mirrors the configuration and a small amount of representative transactional data from production is usually much lower in comparison to implementing redundant systems and complex processes to handle issues with new solution rollouts and software fixes. Any investment in this area goes a long way to prevent issues with failure and downtime.

Disaster Recovery

With the approaches described in [Chapter 5, "High Availability Within A Single Site"](#), you should be able to withstand most single and possible multiple component failures without incurring an outage. With the appropriate architectural patterns described in [Chapter 6, "Architectural Patterns"](#), you may be able to schedule downtime with less impact to the corporations overall availability.

There remains one major contingency you need to consider: what happens if a catastrophic event causes your primary data center to be partially or completely incapacitated? The reasons could range from the commonplace disasters such as fires in the building or natural disasters like floods or earthquakes. It may also be rare events like the Northeast Blackout of 2003 when wide regions covering over eight US states and one Canadian province lost power affecting over 50 million people.

This chapter presents the approaches you could take to ensure that the Sterling Multi-Channel Fulfillment Solution can continue running after a data center outage.

7.1 Disaster Recovery from a Sterling Multi-Channel Fulfillment Solution Perspective

In the event of a data center disaster, you may have almost no option other than to re-establish the Sterling Multi-Channel Fulfillment Solution in a disaster recovery site. This could be an internal site or an office space at a disaster recovery vendor.

Generally, when dealing with a disaster recovery service site, you have to decide on the level of recovery service - the higher the disaster recovery service, the higher the price. Keeping in mind the Insurance Principle, you need to weigh the likelihood of a disaster occurring, and the cost of

the disaster recovery service, against the potential impact to your business due to a prolonged outage.

In the disaster recovery industry the terms cold, warm, and hot site recovery are often used to describe the level of service. A cold site recovery is a term that typically refers to a recovery site that may or may not have equipment provisioned. Depending on your disaster recovery contract, you may have to bring all of your equipment, computing nodes, software, and so forth. In some cases, the disaster recovery vendor may have a pool of equipment that you can draw from. In either case, you have to entertain the possibility that you or your vendors may face a shortage of equipment if multiple customers simultaneously declare disasters.

Typically, the software and equipment are not pre-configured in cold site. Therefore, a cold-site recovery involves a very lengthy and complicated recovery from scratch that could take many days.

A warm recovery site is one where the application may be installed on pre-configured standby equipment and nodes. The data in a warm site are generally updated periodically. Recovery in a warm site typically involves bringing the standby database to the latest consistent state. This generally involves applying all the available transaction logs. A warm-site recovery could take up to a day.

A hot recovery site is one where the application is configured and available at a moments notice. The applications data, ranging from the database to configuration information, are synchronized with the primary data center. A hot-site recovery could take a few minutes to a few hours.

7.2 For Testing Purpose

A warm recovery site is one where the application may be installed on pre-configured standby equipment and nodes. The data in a warm site are generally updated periodically. Recovery in a warm site typically involves bringing the standby database to the latest consistent state. This generally involves applying all the available transaction logs. A warm-site recovery could take up to a day. For more information see, [Section 7.5, "Key to Disaster Recovery"](#).

7.3 Cold Site Recovery

A cold site recovery can be daunting, especially for a large complex system like the Sterling Multi-Channel Fulfillment Solution. At a minimum, you have to procure, install, and configure all the hardware equipment needed by the application ranging from network equipment, load balancers, mid-tier and database nodes, SAN, cabling for the SAN, and so forth.

Next, you have to install and configure all the system software ranging from the operating system, database management system, application server, the Sterling Multi-Channel Fulfillment Solution, and so forth. It is critically important that the software version and release, and even the same patches be installed the same as the primary data center. Installing different software versions may result in unexpected behavior.

Next, you have to configure the environment. At a minimum, this includes:

- Defining all the service, host, and server names to DNS
- Defining the message queues
- Setting all the configuration and performance parameters (for example, the operating system kernel parameters, the database parameters). Again, it is important that these parameters be set to the same values as the corresponding parameters in the primary site.
- Installing and preparing the SAN including defining the storage and file systems
- Loading the application database schema

After the infrastructure and environment is available:

- Restore the database from the backup tapes
- Roll forward all the transaction logs to bring the database up to the latest consistent state
- Configure the application servers (for example, connection pool)
- Restore the messages from the integration queues – if you do not have a backup of the messages, all the of the unprocessed messages in the integration queues are lost
- Install the Sterling Multi-Channel Fulfillment Solution and reapply all the custom code, extensions, custom XMLs, and the property files

- Reconfigure the load balancer or proxy to the application server cluster
- Define the service names to the IP address at the recovery site
- Establish connections to all the external systems (for example, credit card companies for credit authorization)

A cold site recovery could easily take days.

7.4 Warm and Hot Site Recovery

Warm and hot site recoveries are much faster and potentially less risky compared to the cold site recovery because the system is already installed and configured, and the data loaded. Customers who need faster recovery may have to go to with warm or hot recovery sites.

7.5 Key to Disaster Recovery

Consider the following key points for disaster recovery:

- [Recovery Procedures](#)
- [Database Backups and Transaction Log Files](#)
- [Integration Queue Replication](#)
- [Service Names Instead of IP Addresses](#)

7.5.1 Recovery Procedures

Given the extensive list of tasks to recover a system, especially for the cold site recovery, the disaster recovery process must be very well documented and tested. Equally important, these procedures and the entire system must be placed under strict change control and management. Changes to the system must be properly reflected in the recovery procedures. The procedures must be tested as part of the change.

7.5.2 Database Backups and Transaction Log Files

In a cold site recovery, the database has to be restored to the last successful backup and the transaction logs replayed to update the database with all the changes performed since the backup. Given the

importance of these files, many companies copy these files for off-site storage. In some cases, transaction logs are immediately copied to a remote site when the logs are closed.

Standard copy utilities can only copy files that are not opened for access. If you rely on standard copy utilities, you can not backup the currently active (and open) transaction log.

For warm and hot sites recoveries, you could use log-shipping technologies to not only replicate but also to apply the log transactions to the standby database.

For Oracle, you could use products such as Oracle Data Guard or Quest Shareplex.

For UDB, you could use UDB HADR. However, with HADR, you can replicate to only one standby database. If you want to have a standby UDB database server at the local site and the disaster recovery site, you may have to use a combination of cluster failover software and HADR respectively.

7.5.3 Integration Queue Replication

Integration queues are used to exchange data messages between the Sterling Multi-Channel Fulfillment Solution and external systems. The messages could be orders placed between supply chain partners, shipping notices to partner warehouses, and so forth. These messages should be persisted to either files or a database.

If you use file-based persistence, you should consider replicating the files to a remote site to prevent loss of messages from local site faults (for example, JMS server crashing, node crashing). You may also want to consider replicating these messages to a remote site to prevent loss of messages from a data center disaster.

As with transaction logs, you cannot rely on standard copy utilities since these files are continuously opened and updated. Instead, you may have to resort to disk-to-disk replication, such as EMC SRDF, to protect the messages in your integration queue.

7.5.4 Service Names Instead of IP Addresses

You must use service names or host names instead of IP addresses when specifying the location of services such as the JNDI, databases, and JMS

queues. The IP address scheme at the recovery site is not the same as the primary site. If you use IP addresses, you will point to non-existent nodes.

A

agent servers, 7, 9, 36
application servers, 9, 35
 stateful sessions, 36
 stateless sessions, 36
architectural patterns, 45
availability, 1, 4
 design, 1
 motivation, 4
 principles, 1
 requirements, 1

C

caching, 46
client reroute, 23

D

database servers, 12, 19, 22
DBMS
 failures, 19
deployment processes, 48
disaster recovery, 2, 5, 49
 cold site recovery, 51
 procedures, 52
 warm and hot site recovery, 52
disk subsystems, 34
downtimes, 16

E

environment variable
 INSTALL_DIR, xx
 INSTALL_DIR_OLD, xxi

F

failover configurations, 17, 20, 24, 25, 33

H

hardware failures, 20
hot deployment, 47
human errors, 20

I

INSTALL_DIR, xx
INSTALL_DIR_OLD, xxi
integration
 asynchronous, 46
integration servers, 9, 36
 queues, 38

J

Java Virtual Machine, 8
JNDI, 37
JNDI service, 10

L

load balacing
 server-side, 27
load balancing
 client-side, 27

M

mean-time-to-repair, 16
message queues, 21, 38
messages
 asynchronous, 4

N

networked file systems, 44
nodes, 16, 24

O

operator errors, 20
outages, 4, 16

R

RAC server instance failures, 32
regression testing, 48

S

single-points-of-failure, 12, 15
single-site configuration, 11
Sterling Multi-Channel Fulfillment Solution, 7

T

target node utilization, 33
time-triggered transactions, 12
transactions
 loss, 14
 loss in integration queues, 14

W

WebLogic JMS, 39
WebSphere MQ, 42
workloads, 4