Sterling Commerce
An IBM Company

Selling and Fulfillment Foundation: Customizing APIs Guide

Release 8.5

Last updated in HF16

June 2010

Copyright Notice

STERLING COMMERCE SOFTWARE

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

Third-Party Software

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 © 2006-2008 Andres Almiray. Copyright © 1999-2005 The Apache Software Foundation. Copyright (c) 2008 Azer Koçulu http://azer.kodfabrik.com. Copyright © Einar Lielmanis, einars@gmail.com. Copyright (c) 2006 John Reilly (www.inconspicuous.org) and Copyright (c) 2002 Douglas Crockford (www.crockford.com). Copyright (c) 2009 John Resig, http://jquery.com/. Copyright © 2006-2008 Json-lib. Copyright © 2001 LOOX Software, Inc. Copyright © 2003-2008 Luck Consulting Pty. Ltd. Copyright 2002-2004 © MetaStuff, Ltd. Copyright © 2009 Michael Mathews micmath@gmail.com. Copyright © 1999-2005 Northwoods Software Corporation. Copyright (C) Microsoft Corp. 1981-1998. Purple Technology, Inc. Copyright (c) 2004-2008 QOS.ch. Copyright © 2005 Sabre Airline Solutions. Copyright © 2004 SoftComplex, Inc. Copyright © 2000-2007 Sun Microsystems, Inc. Copyright © 2001 VisualSoft Technologies Limited. 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 2000, 2006 IBM Corporation and others. Copyright © 1987-2006 ILOG, Inc. Copyright © 2000-2006 Infragistics. Copyright © 2002-2005 JBoss, Inc. Copyright LuMriX.net GmbH, Switzerland. Copyright © 1998-2009 Mozilla.org. Copyright © 2003-2009 Mozdev Group, Inc. 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. Copyright © 2002-2009 Zipwise Software. All rights reserved by all listed parties.

Certain components of the Sterling Commerce Software are distributed on the same storage media as Third Party Software which are not listed above. Additional information for such Third Party Software components of the Sterling Commerce Software is located at: installdir/mesa/studio/plugins/SCI_Studio_License.txt.

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 installdir/SCI_License.txt.

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

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.

THE APACHE SOFTWARE FOUNDATION SOFTWARE

The Sterling Commerce Software is distributed with or on the same storage media as the following software products (or components thereof) and java source code files: Xalan version 2.5.2, Cookie.java, Header.java, HeaderElement.java, HttpException.java, HttpState.java, NameValuePair.java, CronTimeTrigger.java, DefaultTimeScheduler.java, PeriodicTimeTrigger.java, Target.java,

TimeScheduledEntry.java, TimeScheduler.java, TimeTrigger.java, Trigger.java, BinaryHeap.java, PriorityQueue.java, SynchronizedPriorityQueue.java, GetOpt.java, GetOptsException.java, IllegalArgumentException.java, MissingOptArgException.java (collectively, "Apache 1.1 Software"). Apache 1.1 Software is free software which is distributed under the terms of the following license:

License Version 1.1

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

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

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

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

4. The names "Commons", "Jakarta", "The Jakarta Project", "HttpClient", "log4j", "Xerces "Xalan", "Avalon", "Apache Avalon", "Avalon Cornerstone", "Avalon Framework", "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without specific prior written permission. For written permission, please contact apache@apache.org.

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

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

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. The GetOpt.java, GetOptsException.java, IlligalArgumentException.java and MissingOptArgException.java software was originally based on software copyright (c) 2001, Sun Microsystems., http://www.sun.com. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

The preceding license only applies to the Apache 1.1 Software and does not apply to the Sterling Commerce Software or to any other Third-Party 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, Apache File Upload Package, Apache Commons Beans, Apache Commons BetWixt, Apache Commons Collection, Apache Commons Digester, Apache Commons IO, Apache Commons Lang., Apache Commons Logging, Apache Commons Net, Apache Jakarta Commons Pool, Apache Jakarta ORO, Lucene, Xerces version 2.7, Apache Log4J, Apache SOAP, Apache Struts 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: installdir/jar/commons_upload/1_0/ CommonsFileUpload_License.txt,
installdir/jar/jetspeed/1_4/RegExp_License.txt,
installdir/ant/Ant_License.txt
<install>/jar/antInstaller/0_8/antinstaller_License.txt,
<install>/jar/commons_beanutils/1_7_0/commons-beanutils.jar (/META-INF/LICENSE.txt),
<install>/jar/commons_betwixt/0_8/commons-betwixt-0.8.jar (/META-INF/LICENSE.txt),

<install>/jar/commons_collections/3_2/LICENSE.txt,
<install>/jar/commons_digester/1_8/commons-digester-1.8.jar (/META-INF/LICENSE.txt),
<install>/jar/commons_io/1_4/LICENSE.txt,
<install>/jar/commons_lang/2_1/Commons_Lang_License.txt,
<install>/jar/commons_logging/1_0_4/commons-logging-1.0.4.jar (/META-INF/LICENSE.txt),
<install>/jar/commons_net/1_4_1/commons-net-1.4.1.jar (/META-INF/LICENSE.txt),
<install>/jar/smcfs/8.5/lucene-core-2.4.0.jar (/META-INF/LICENSE.txt),
<install>/jar/struts/2_0_11/struts2-core-2.0.11.jar (./LICENSE.txt),
<install>/jar/mesa/gisdav/WEB-INF/lib/Slide_License.txt,
<install>/mesa/studio/plugins/xerces_2.7_license.txt,
<install>/jar/commons_pool/1_2/Commons_License.txt,
<install>/jar/jakarta_oro/2_0_8/JakartaOro_License.txt,
<install>/jar/log4j/1_2_15/LOG4J_License.txt,
<install>/jar/xalan/2_7/Xalan_License.txt,
<install>/jar/soap/2_3_1/Apache_SOAP_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. 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."

NOTICE file corresponding to the section 4 d of the Apache License, Version 2.0, in this case for the Apache Ant distribution. Apache Ant Copyright 1999-2008 The Apache Software Foundation. This product includes software developed by The Apache Software Foundation (http://www.apache.org/). This product includes also software developed by :

- the W3C consortium (http://www.w3c.org) ,

- the SAX project (http://www.saxproject.org)

The <sync> task is based on code Copyright (c) 2002, Landmark Graphics Corp that has been kindly donated to the Apache Software Foundation.

Portions of this software were originally based on the following:

- software copyright (c) 1999, IBM Corporation., http://www.ibm.com.

- software copyright (c) 1999, Sun Microsystems., http://www.sun.com.

- voluntary contributions made by Paul Eng on behalf of the Apache Software Foundation that were originally developed at iClick, Inc., software copyright (c) 1999.

NOTICE file corresponding to the section 4 d of the Apache License, Version 2.0, in this case for the Apache Lucene distribution. Apache Lucene Copyright 2006 The Apache Software Foundation. This product includes software developed by The Apache Software Foundation (http://www.apache.org/). The snowball stemmers in contrib/snowball/src/java/net/sf/snowball were developed by Martin Porter and Richard Boulton. The full snowball package is available from http://snowball.tartarus.org/

Ant-Contrib Software

The Sterling Commerce Software is distributed with or on the same storage media as the Anti-Contrib software (Copyright (c) 2001-2003 Ant-Contrib project. All rights reserved.) (the "Ant-Contrib Software"). The Ant-Contrib Software is free software which is distributed under the terms of the following license:

The Apache Software License, Version 1.1

DOM4J Software

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

The preceding license only applies to the Dom4j Software and does not apply to the Sterling Commerce Software, or any other Third-Party Software.

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.apt.core.nl1_3.2.0.v200606220026.jar,
org.eclipse.jdt.apt.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>/SI/repository/rcp/rcpdependencies/windows/eclipse/notice.html,
<install_dir>/SI/repository/rcp/rcpdependencies/windows/eclipse/plugins/notice.html,
<install_dir>/SI/repository/rcp/rcpdependencies/gtk.linux_x86/eclipse/notice.html, and
<install_dir>/SI/repository/rcp/rcpdependencies/gtk.linux_x86/eclipse/plugins/notice.html.

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

<install_dir>/SI/repository/rcp/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.

Ehcache Software

The Sterling Commerce Software is also distributed with or on the same storage media as the ehache v.1.5 software (Copyright © 2003-2008 Luck Consulting Pty. Ltd.) ("Ehache Software"). Ehcache 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 <install>/jar/smcfs/8.5/ehcache-1.5.0.jar (./LICENSE.txt).

The Ehcache Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to the Ehcache 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 Ehcache 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."

EZMorph Software

The Sterling Commerce Software is also distributed with or on the same storage media as the EZMorph v. 1.0.4 software (Copyright © 2006-2008 Andres Almiray) ("EZMorph Software"). EZMorph 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 <install>/jar/ezmorph/1_0_4/ezmorph-1.0.4.jar (./LICENSE.txt).

The EZMorph Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to the EZMorph 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 EZMorph 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."

Firebug Lite Software

The Sterling Commerce Software is distributed with or on the same storage media as the Firebug Lite Software which is free software distributed under the terms of the following license:

Redistribution and use of this software in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

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

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

* Neither the name of Azer Koçulu. nor the names of any other contributors may be used to endorse or promote products derived from this software without specific prior written permission of Parakey Inc.

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

ICE 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"). The ICE Software is independent from and not linked or compiled with the Sterling Commerce Software. The ICE Software is a free software product 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 installdir/jar/jniregistry/1_2/ICE_License.txt. This license only applies to the ICE 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: installdir/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 is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

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 is not distributed by Sterling Commerce in its entirety. Rather, the distribution is limited to the following jar files: el-api.jar, jasper-compiler-5.5.15.jar, jasper-el.jar, jasper.jar, jboss-common-client.jar, jboss-j2ee.jar, jboss-jmx.jar, jboss-jsr77-client.jar, jbossmq-client.jar,

jnpserver.jar, jsp-api.jar, servlet-api.jar, tomcat-juli.jar.

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 note. 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.

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.

JSLib Software

The Sterling Commerce Software is distributed with or on the same storage media as the JSLib software product (Copyright (c) 2003-2009 Mozdev Group, Inc.) ("JSLib Software"). The JSLib Software is distributed under the terms of the MOZILLA PUBLIC LICENSE Version 1.1. A copy of this license is located at <install>\repository\eardata\platform_uifwk_ide\war\designer\MPL-1.1.txt. The JSLib Software code is distributed in source form and is located at http://jslib.mozdev.org/installation.html. 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 JSLib Software (and not to the Sterling Commerce Software or any other Third-Party Software):

"The contents of the file located at http://www.mozdev.org/source/browse/jslib/ 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/MPL-1.1.html.

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 Mozdev Group, Inc. code. The Initial Developer of the Original Code is Mozdev Group, Inc. Portions created by_Mozdev Group, Inc. are Copyright © 2003 Mozdev Group, Inc. All Rights Reserved. Original Author: Pete Collins <pete@mozdev.org>one 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 JSLib Software and does not apply to the Sterling Commerce Software, or any other Third-Party Software.

Json Software

The Sterling Commerce Software is also distributed with or on the same storage media as the Json 2.2.2 software (Copyright © 2006-2008 Json-lib) ("Json Software"). Json 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 <install>/jar/jsonlib/2_2_2/json-lib-2.2.2-jdk13.jar.

This product includes software developed by Douglas Crockford (http://www.crockford.com).

The Json Software was not modified. Neither the Sterling Commerce Software, modifications, if any, to the Json 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 Json 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."

Purple Technology

The Sterling Commerce Software is distributed with or on the same storage media as the Purple Technology Software (Copyright (c) 1995-1999 Purple Technology, Inc.) ("Purple Technology Software"), which is subject to the following license:

Copyright (c) 1995-1999 Purple Technology, Inc. All rights reserved.

PLAIN LANGUAGE LICENSE: Do whatever you like with this code, free of charge, just give credit where credit is due. If you improve it, please send your improvements to alex@purpletech.com. Check http://www.purpletech.com/code/ for the latest version and news.

LEGAL LANGUAGE LICENSE: Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

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

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

3. The names of the authors and the names "Purple Technology," "Purple Server" and "Purple Chat" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact server@purpletech.com.

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

The preceding license only applies to the Purple Technology Software and does not apply to the Sterling Commerce Software, or 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 in <install>/repository/eardata/platform/war/ajax/scripts/Rico_License.txt.

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."

Rhino Software

The Sterling Commerce Software is distributed with or on the same storage media as the Rhino js.jar (Copyright (c) 1998-2009 Mozilla.org.) ("Rhino Software"). A majority of the source code for the Rhino Software is dual licensed under the terms of the MOZILLA PUBLIC LICENSE Version 1.1. or the GPL v. 2.0. Additionally, some files (at a minimum the contents of toolsrc/org/Mozilla/javascript/toolsdebugger/treetable) are available under another license as set forth in the directory file for the Rhino Software.

Sterling Commerce's use and distribution of the Rhino Software is under the Mozilla Public License. A copy of this license is located at <install>/3rdParty/rico license.doc. The Rhino Software code is distributed in source form and is located at http://mxr.mozilla.org/mozilla/source/js/rhino/src/. 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 Rhino Software (and not to the Sterling Commerce Software or any other Third-Party Software):

"The contents of the file located at <install>/jar/rhino/1_7R1/js.jar 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 Rhino code, released May 6, 1999. The Initial Developer is Netscape Communications Corporation. Portions created by the Initial Developer are Copyright © 1997-1999. All Rights Reserved. Contributor(s):_____none listed.

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

Sun Microsystems

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

as the following software products (or components thereof): Sun JMX, and Sun JavaMail (collectively, "Sun Software"). Sun Software is free software which is distributed under the terms of the licenses issued by Sun which are included in the directory files located at:

SUN COMM JAR - <install>/Applications/Foundation/lib

SUN ACTIVATION JAR - <install>/ Applications/Foundation/lib

SUN JavaMail - <install>/jar/javamail/1_4/LICENSE.txt

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 http://kenai.com/projects/javamail/sources/mercurial/show.

The source code for the Web-App Software may be found at: <install>/3rdParty/sun/javamail-1.3.2/docs/JavaMail-1.2.pdf

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.

The Sterling Commerce Software is also distributed with or on the same storage media as the Sun Microsystems, Inc. Java (TM) look and feel Graphics Repository ("Sun Graphics Artwork"), subject to the following terms and conditions:

Copyright 2000 by Sun Microsystems, Inc. All Rights Reserved.

Sun grants you ("Licensee") a non-exclusive, royalty free, license to use, and redistribute this software graphics artwork, as individual graphics or as a collection, as part of software code or programs that you develop, provided that i) this copyright notice and license accompany the software graphics artwork; and ii) you do not utilize the software graphics artwork in a manner which is disparaging to Sun. Unless enforcement is prohibited by applicable law, you may not modify the graphics, and must use them true to color and unmodified in every way.

This software graphics artwork is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE GRAPHICS ARTWORK.

IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE GRAPHICS ARTWORK, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

If any of the above provisions are held to be in violation of applicable law, void, or unenforceable in any jurisdiction, then such provisions are waived to the extent necessary for this Disclaimer to be otherwise enforceable in such jurisdiction.

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

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

Contents

8 Invoking APIs and Services

Index

Preface

This manual explains how to invoke standard APIs for displaying data in the UI and customized APIs for executing your custom logic.

Intended Audience

This manual is intended for use by those who are responsible for customizing Selling and Fulfillment Foundation.

Structure

This document contains the following chapters:

Chapter 1, "Checklist for Customization Projects"
This chapter describes a checklist of the tasks you need to perform to customize the different components of Selling and Fulfillment Foundation.

Chapter 2, "Extending Services"
This chapter provides information about extending servies and instructions on how to invoke extended services.

Chapter 3, "Understanding APIs"
This chapter describes the behavior of both Selling and Fulfillment Foundationany extended (custom) APIs that you create.

Chapter 4, "Input XML Files for APIs"
This chapter provides information on how to modify the input XML for APIs and guidelines for forming API input.

Chapter 5, "Output XML Files for APIs"

This chapter provides information on how to modify the output XML template for APIs amd guidelines for creating custom output XML templates.

Chapter 6, "DTDs, XSDs, and Complex Queries"

This chapter provides information on DTD and XSD Generator and how to define complex queries.

Chapter 7, "Creating Extended APIs"

This chapter provides information on how to create and invoke extended APIs.

Chapter 8, "Invoking APIs and Services"

This chapter provides information on how to invoke services and standard APIs programmatically.

Selling and Fulfillment Foundation Documentation

For more information about the Selling and Fulfillment Foundation components, see the following manuals:

- *Selling and Fulfillment Foundation: Release Notes*
- *Selling and Fulfillment Foundation: Installation Guide*
- *Selling and Fulfillment Foundation: Upgrade Guide*
- *Selling and Fulfillment Foundation: Configuration Deployment Tool Guide*
- *Selling and Fulfillment Foundation: Performance Management Guide*
- *Selling and Fulfillment Foundation: High Availability Guide*
- *Selling and Fulfillment Foundation: System Management Guide*
- *Selling and Fulfillment Foundation: Localization Guide*
- *Selling and Fulfillment Foundation: Customization Basics Guide*
- *Selling and Fulfillment Foundation: Customizing APIs Guide*

- *Selling and Fulfillment Foundation: Customizing Console JSP Interface for End User Guide*
- *Selling and Fulfillment Foundation: Customizing the RCP Interface Guide*
- *Selling and Fulfillment Foundation: Customizing User Interfaces for Mobile Devices Guide*
- *Selling and Fulfillment Foundation: Customizing Web UI Framework Guide*
- *Selling and Fulfillment Foundation: Customizing Swing Interface Guide*
- *Selling and Fulfillment Foundation: Extending the Condition Builder Guide*
- *Selling and Fulfillment Foundation: Extending the Database Guide*
- *Selling and Fulfillment Foundation: Extending Transactions Guide*
- *Selling and Fulfillment Foundation: Using Sterling RCP Extensibility Tool Guide*
- *Selling and Fulfillment Foundation: Integration Guide*
- *Selling and Fulfillment Foundation: Product Concepts Guide*
- *Sterling Warehouse Management™ System: Concepts Guide*
- *Selling and Fulfillment Foundation: Application Platform Configuration Guide*
- *Sterling Distributed Order Management™: Configuration Guide*
- *Sterling Supply Collaboration: Configuration Guide*
- *Sterling Global Inventory Visibility™: Configuration Guide*
- *Catalog Management™: Configuration Guide*
- *Sterling Logistics Management: Configuration Guide*
- *Sterling Reverse Logistics™: Configuration Guide*
- *Sterling Warehouse Management System: Configuration Guide*
- *Selling and Fulfillment Foundation: Application 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*
- *Selling and Fulfillment Foundation: Mobile Application User Guide*
- *Selling and Fulfillment Foundation: Business Intelligence Guide*
- *Selling and Fulfillment Foundation: Javadocs*
- *Sterling Selling and Fulfillment Suite™: Glossary*
- *Parcel Carrier: Adapter Guide*
- *Selling and Fulfillment Foundation: Multitenant Enterprise Guide*
- *Selling and Fulfillment Foundation: Password Policy Management Guide*
- *Selling and Fulfillment Foundation: Properties Guide*
- *Selling and Fulfillment Foundation: Catalog Management Concepts Guide*
- *Selling and Fulfillment Foundation: Pricing Concepts Guide*
- *Business Center: Item Administration Guide*
- *Business Center: Pricing Administration Guide*
- *Business Center: Customization Guide*
- *Business Center: Localization Guide*

Conventions

In this manual, Windows refers to all supported Windows operating systems.

The following conventions may be used in this manual:

Convention	Meaning
. . .	Ellipsis represents information that has been omitted.

Convention	Meaning
< >	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 Selling and Fulfillment Foundation installation directory. This is only applicable for Release 8.0 or later.
<INSTALL_DIR_OLD>	User-supplied location of the Selling and Fulfillment Foundation installation directory (for Release 8.0 or later). **Note:** This is applicable only for users upgrading from Release 8.0 or later.
<YANTRA_HOME>	User-supplied location of the Sterling Supply Chain Applications installation directory. This is only applicable for Releases 7.7, 7.9, and 7.11.
<YANTRA_HOME_OLD>	User-supplied location of the Sterling Supply Chain Applications installation directory (for Releases 7.7, 7.9, or 7.11). **Note:** This is applicable only for users upgrading from Releases 7.7, 7.9, or 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 or above, the <YANTRA_HOME>/`Runtime` directory is no longer used and this is the same location as <INSTALL_DIR>.

Convention	Meaning
<YFS_HOME_OLD>	This is the <YANTRA_HOME>/Runtime directory for Releases 7.7, 7.9, or 7.11. **Note:** This is only applicable for users upgrading from Releases 7.7, 7.9, or 7.11.
<ANALYTICS_HOME>	User-supplied location of the Sterling Analytics installation directory. **Note:** This convention is used only in the *Selling and Fulfillment Foundation: Business Intelligence Guide*.
<COGNOS_HOME>	User-supplied location of the IBM Cognos 8 Business Intelligence installation directory. **Note:** This convention is used only in the *Selling and Fulfillment Foundation: Business Intelligence Guide*.
<MQ_JAVA_INSTALL_PATH>	User-supplied location of the IBM WebSphere® MQ Java components installation directory. **Note:** This convention is used only in the *Selling and Fulfillment Foundation: System Manangement and Administration Guide*.
<DB>	Refers to Oracle®, IBM DB2®, or Microsoft SQL Server® depending on the database server.
<DB_TYPE>	Depending on the database used, considers the value oracle, db2, or sqlserver.

Note: The Selling and Fulfillment Foundation documentation set uses the following conventions in the context of the product name:

- Yantra is used for Release 7.7 and earlier.
- Sterling Supply Chain Applications is used for Releases 7.9 and 7.11.
- Sterling Multi-Channel Fulfillment Solution is used for Releases 8.0 and 8.2.
- Selling and Fulfillment Foundation is used for Release 8.5.

1

Checklist for Customization Projects

This chapter provides a high-level checklist for the tasks involved in customizing or extending Selling and Fulfillment Foundation.

1.1 Customization Projects

Projects to customize or extend Selling and Fulfillment Foundation vary with the type of changes that are needed. However, most projects involve an interconnected series of changes that are best carried out in a particular order. The checklist identifies the most common order of customization tasks and indicates which guide in the documentation set provides details about each stage.

1. Prepare your development environment

   Set up a development environment that mirrors your production environment, including whether you deploy Selling and Fulfillment Foundation on a WebLogic, WebSphere, or JBoss application server. Doing so ensure that you can test your extensions in a real-time environment.

   You install and deploy Selling and Fulfillment Foundation in your development environment following the same steps that you used to install and deploy Selling and Fulfillment Foundation in your production environment. Refer to Selling and Fulfillment Foundation system requirements and installation documentation for details.

   An option is to customize Selling and Fulfillment Foundation with Microsoft COM+. Using COM+ provides you with advantages such as increased security, better performance, increased manageability of server applications, and support for clients of mixed environments. If

this is your choice, see the *Selling and Fulfillment Foundation: Customization Basics Guide* about additional installation instructions.

2. Plan your customizations

   Are you adding a new menu entry, customizing the Sign In screen and logo, creating new themes, customizing views and wizards, or adding new screens? Each type of customization varies in scope and complexity. For background, see the *Selling and Fulfillment Foundation: Customization Basics Guide*, which summarizes the types of changes that you can make.

   Important guidelines about file names, keywords, and other conventions are found in the *Selling and Fulfillment Foundation: Customization Basics Guide*.

3. Extend the Database

   For many customization projects, the first task is to extend the database so that it supports the other UI or API changes that you make later. For instructions, see the *Selling and Fulfillment Foundation: Extending the Database Guide* which include information about the following topics:

   - Important guidelines about what you can and cannot change in the database.
   - Information about modifying APIs. If you modify database tables so that any APIs are impacted, you must extend the templates of those APIs or you cannot store or retrieve data from the database. This step is required if table modifications impact an API.
   - How to generate audit references so that you improve record management by tracking records at the entity level. This step is optional.

4. Make other changes to APIs

   Selling and Fulfillment Foundation can call or invoke standard APIs or custom APIs. For background about APIs and the services architecture in Selling and Fulfillment Foundation, including service types, behavior, and security, see the *Selling and Fulfillment Foundation: Customizing APIs Guide*. This guide includes information about the following types of changes:

- How to invoke standard APIs for displaying data in the UI and also how to save the changes made to the UI in the database.
- Invoke customized APIs for executing your custom logic in the extended service definitions and pipeline configurations.
- APIs use input and output XML to store and retrieve data from the database. If you don't extend these API input and output XML files, you may not get the results you want in the UI when your business logic is executing.
- Every API input and output XML file has a DTD and XSD associated to it. Whenever you modify input and output XML, you must generate the corresponding DTD and XSD to ensure data integrity. If you don't generate the DTD and XSD for extended Application XMLs, you may get inconsistent data.

5. Customize the UI

   Sterling Commerce applications support several UI frameworks. Depending on your application and the customizations you want to make, you may work in only one or in several of these frameworks. Each framework has its own process for customizing components like menu items, logos, themes, and etc. Depending on the framework you want, consult one of the following guides:

   - *Selling and Fulfillment Foundation: Customizing Console JSP Interface for End User Guide*
   - *Selling and Fulfillment Foundation: Customizing the Swing Interface Guide*
   - *Selling and Fulfillment Foundation: Customizing User Interfaces for Mobile Devices Guide*
   - *Selling and Fulfillment Foundation: Customizing the RCP Interface Guide* and *Selling and Fulfillment Foundation: Using the Sterling RCP Extensibility Tool Guide*
   - *Customizing the Web UI Framework Guide*

6. Extend Transactions

   You can extend the standard Selling and Fulfillment Foundation to enhance the functionality of your implementation of Selling and Fulfillment Foundation and to integrate with external systems. For background about transaction types, security, dynamic variables, and extending the

Condition Builder, see the *Selling and Fulfillment Foundation: Extending Transactions Guide Selling and Fulfillment Foundation: Extending the Condition Builder Guide* . These guides includes information about the following types of changes:

- How to extend Selling and Fulfillment Foundation Condition Builder to define complex and dynamic conditions for executing your custom business logic and using a static set of attributes.
- How to define variables to dynamically configure properties belonging to actions, agents, and services configurations.
- How to set up transactional data security for controlling who has access to what data, how much they can see, and what they can do with it.
- How to create custom time-triggered transactions. You can invoke and schedule these custom time-triggered transactions in much the same manner as you invoke and schedule Selling and Fulfillment Foundation standard time-triggered transactions. Finally, you can coordinate your custom, time-triggered transactions with external transactions and run them either by raising an event, calling a user exit, or invoking a custom API or service.

7. Build and deploy your customizations or extensions

   After performing the customizations that you want, you must build and deploy your customizations or extensions. First, build and deploy these customizations or extensions in the test environment for verification. When you are ready, repeat the same process to build and deploy your customizations and extensions in the production environment. For instructions, see the *Selling and Fulfillment Foundation: Customization Basics Guide* which includes information about the following topics:

   - How to build and deploy standard resources, database, and other extensions (such as templates, user exits, java interfaces).
   - How to build and deploy Enterprise-Level extensions.

2

Extending Services

2.1 About Extending Services

In the Selling and Fulfillment Foundation terminology, a service is core business logic component that is stateless and does not contain presentation logic. Each service (either provided out-of-the-box by Selling and Fulfillment Foundation or those that are custom created using the Service Definition Framework) represents a logical unit of processing that can be independently performed without any loss of data integrity and within one transaction boundary. Using the Service Definition Framework, one or more services can be aggregated into larger composite services which can in turn be used to create other services. This provides a way to build small reusable components that can be linked together to provide complex business processing.

All services within the Service Definition Framework can be invoked bidirectionally either through internal Selling and Fulfillment Foundation business processes or through external systems. Services deployed in the Service Definition Framework are stateless, each having their own transaction commitment boundaries.

A service can be invoked by Selling and Fulfillment Foundation by associating the service with an event through an action. You can use a standard interoperability event handler or implement your own custom event handler. You can then configure Selling and Fulfillment Foundation to invoke the event handlers when certain events are raised and conditions are met. For more information about configuring events, conditions, and actions, see the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Selling and Fulfillment Foundation provides several user exits to extend business logic. User exits invoked from within transactions can be associated to a service when configuring transactions. Note that

templates are not supported for user exits. For more information on configuring user exits, see the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Once services have been configured, they can also be invoked programmatically by a client.

The service invocation configuration depends on the location of the client invoking the service in relation to the location of the Selling and Fulfillment Foundation installation, as described in the following situations:

- If the invoking client ***do not*** have Selling and Fulfillment Foundation installed - Configure for remote invocation.
- If the invoking client ***do*** have Selling and Fulfillment Foundation installed - Configure for local invocation.

2.2 Invoking Services Synchronously or Asynchronously

Depending upon the mode of invocation, services can be classified into two major categories:

- Synchronously invoked services (on demand) - These services can perform all their processing and return the result in single call.
- Asynchronously invoked services (message driven)

2.2.1 Synchronously Invoked Services

These services can perform all their processing and return the result in a single call, on demand.

2.2.2 Asynchronously Invoked Services

These services automatically perform all their processing whenever triggered by a message from an external system or from within Selling and Fulfillment Foundation. The trigger could be in the form of a file, a database record or a message in a message queue depending upon the mode of integration. These services do not return any value and are purely used for background processing such as sending out emails or

automatically receiving updates from or sending updates to an external system.

In general, asynchronous services provide a lower cost to performance ratio than synchronous services and should be preferred wherever possible. However, asynchronous services queue up and process messages in the order they are received. The time to process a certain transaction after it's been queued can vary widely depending upon peaks in your processing cycle and a host of other factors. Therefore, they are not suitable for certain specific scenarios where an SLA (service level agreement) requires that a transaction has to be processed within a specified short time frame. However, these scenarios are rare for most businesses and business processes and asynchronous processing is efficient enough for the majority of transactions at a significantly lower cost while still providing a high service level.

2.3 Business Functions To Use In Services

A service typically consists of one or more messaging components (or components that define how messages to and from the service are handled), one or more utility components (such as email or alert handlers) and one or more business processing components. For information about the utility and messaging components available for services defined in Selling and Fulfillment Foundation, see the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. This section describes how to work with service, customize and extending the business processing components, and make them usable in services.

Selling and Fulfillment Foundation is shipped with an extensive out-of-the-box business function library. Each function in this library is known as a standard API. For detailed information on the input, output, and behavior of each standard API, see the *Selling and Fulfillment Foundation: Javadocs*.

This chapter discusses common programming, customization and extensibility patterns that are applicable to large portions of the Selling and Fulfillment Foundation business function library.

You can also write your own business functions and use them in services. Each such function is known as an extended API.

While standard APIs can be aggregated and linked together to form more complex services, for most cases the API provides all the functionality

that is required for a business transaction and is therefore not required to be linked together with other components or APIs. To ease working with this most common scenario, all of the standard APIs are automatically available for synchronous invocation without the need to model each one as a service using the Service Definition Framework. You can think of these APIs as "automatically defined" synchronous services. However, extended APIs and asynchronous invocation of these APIs requires that you explicitly model them as services first using the Service Definition Framework.

2.4 Message Size For Asynchronous Services

If a database table reaches its maximum size and a send( ) function attempts to insert a message in the table, the Service Definition Framework throws an exception. See the size limitations noted in the Table 2–1:

Table 2–1 Message Size for Asychronous Transports

Mechanism	Data Type	Size
Database (Oracle)	CLOB	4 GB
Database (Microsoft SQL Server)	TEXT	4 GB
JMS Queue	TextMessage	4 GB
MSMQ Queue	Message	2 GB

2.5 Exception Handling and Services

The Alert Console displays all exceptions logged by the Service Definition Framework. It also enables you to reprocess exceptions that occur in transactions configured to be asynchronous. When using a database or queue, calls are asynchronous.

The Service Definition Framework uses the log4j utility for logging exception information. The log4j utility writes both trace and debug information to a log file. You can configure the logger to send different categories of messages to different destinations. Categories are organized hierarchically, which permits inheritance. Each category can be configured with a priority indicating a severity level. If a category is not configured with a priority, it inherits the priority of its closest ancestor with an assigned priority.

All exceptions that occur during an API call or during use of an event handler are logged.

3

Understanding APIs

3.1 About APIs

You can use both standard APIs that are supplied by Selling and Fulfillment Foundation and any extended (custom) APIs that you have created. Selling and Fulfillment Foundation provide standard APIs to handle the most common business scenarios. For example, there are APIs that create an order, allocate an order, and report shipment confirmation. Standard APIs can be invoked directly or aggregated into more complex services.

3.2 API Behavior

Each API takes an XML document as input and returns another XML document as output. The `YFSEnvironment` input parameter represents a runtime state under which this API is being invoked. It is used by Selling and Fulfillment Foundation for the following tasks:

- Security audits and logging
- Transaction control
- Achieving invocation-specific API behavior

For an asynchronous service, Selling and Fulfillment Foundation automatically creates an instance of this object and passes it to each API part of the service. To programmatically invoke a synchronous service, you have to create an instance of this environment by calling the `createEnvironment()` API.

Note: In general, input to APIs should not contain any "BLANK" elements or attributes. A blank element can be defined as an element containing all the attributes with blank values. If a blank element is passed, the API behavior is unpredictable.

All APIs (whether standard or extended) have the same signature with respect to input parameters and return values. This signature is of the form

```
org.w3c.dom.Document APIName(YFSEnvironment env, org.w3c.dom.Document input);
```

In order for custom APIs to access custom values, the API should implement the `com.yantra.interop.japi.YIFCustomApi` interface. If entered, these name/value pairs are passed to the Custom API as a Properties object. See the *Selling and Fulfillment Foundation: Javadocs* for more information about the `com.yantra.interop.japi.YIFCustomApi` interface.

3.3 Types of APIs

An API processes records based on key attribute values, processing records with a primary key first. If the primary key is not found, the API then searches for the logical keys and then processes those records. For example, the `ChangeOrder()` API first looks for the OrderHeaderKey key attribute and then for the combination of the OrderNo and EnterpriseCode key attributes.

3.3.1 Select APIs

Typically prefixed with `get`, select APIs return one record for an entity (for example, the `getOrderDetails()` API returns the details of one order). They do not update the database.

Since select APIs return only one record, they require unique key attributes to be passed in the input XML. If a unique key attribute is not passed in the input XML, the API uses blanks for those attributes in the criteria to select the record. There can be more than one unique key combination, and in that combination you must pass any one of the multiple combinations.

For example, an order is uniquely identified either by the OrderHeaderKey key attribute or by a combination of the OrderNo and EnterpriseCode attributes. So, when calling the getOrderDetails() API, you must pass either the OrderHeaderKey attribute or the combination of the OrderNo, EnterpriseCode and DocumentType key attributes. If you pass only OrderNo, the API returns the order that matches OrderNo and has a blank enterprise code. In order to identify the unique key combinations for each API, see the *Selling and Fulfillment Foundation: Javadocs*.

However, getOrderDetails() API uses a select for update on YFS_ORDER_HEADER so that its internal processes such as user exits, events, etc., have a lock on the order elements while the thread working on it is active. This enables to maintain a transaction cache until the final commit. Hence, you need to avoid using nested transactions to overcome the locking mechanism by performing:

1. Commit or rollback only once for all event of the order. Keep in mind, that all the events are set to rollback if one of them fails.
2. Select the order for each event and process. Also keep in mind, that if age of the orders having multiple events are higher it could have an impact on the performance.

3.3.2 List APIs

Typically prefixed with get, list APIs return a list of records for an entity that match the criteria specified through the input XML, for example, the getOrderList() API returns a list of orders. For more information about specifying the search criteria, see Section 4.3, "Forming Queries in the Input XML of List APIs". If any attribute in the input XML has a blank value, it is ignored. List APIs do not update the database. You can also get the paginated data from a list API by calling the getPage API and passing the list API as the input to the getPage API. For more information about the getPage API, see the *Selling and Fulfillment Foundation: Javadocs*.

3.3.3 Update APIs

Update APIs insert new records into the database. They also modify or delete existing records in the database. Update APIs that modify or delete existing records use the same logic as select APIs to identify which record to update. If no record is found, update APIs throw an exception.

3.4 API Security

When calling an API, you must pass through the following two levels of security:

1. Authentication with a user ID, a certificate or both. The login API is called before any other API is called.
2. Authorization, which verifies which API that you can access.

This security procedure is for every API call that is made through an application server process. By default, agent and integration servers always have full access to APIs.

Once you have passed the authentication check, an authorization check determines what APIs and resources you can access. This authorization check is in addition to the user interface (UI) security. For example, the UI security might allow you access to a screen that lists users. To generate a list of users at the screen, you might also have to pass an authorization check for the getUserList API that lists the users.

Other examples of authorization checks include:

- If you use the getCommonCodeList API for display purposes, you should not be able to get user information that is explicitly restricted from the output of the API.
- If you call the getUserList API before assigning an alert, you should not be able to get user passwords.
- If you use the UserHierarchy API to change your password:
  - You should not be able to change your own IsSuperUser flag.
  - You should not be able to modify another user's information.
  - You should no be able to subscribe to additional user groups, which would give you more system access.

This security is implemented using the apisecurity specific template files. These apisecurity template files are XML files that documents the input and output elements to which (by default) all APIs are restricted. These files are automatically generated during XAPI deployment, even when document generation is turned off.

Note: Services do not use the apisecurity file.

Templates are used for the input and output authorization checks. These templates override the regular templates.

For example, an input template with the lines OrganizationCode=#PROHIBITED# and IsSuperUser=#PROHIBITED# would prevent you from subscribing to more user groups and gain more permissions.

The output template supplements the filtering performed by the default documentation-based template. If an element is restricted because it is not configured in the apisecurity file, it will never be returned in the output, even if present in the documentation-based template.

Note: At certain points in the input and output, APIs like multiApi and getPage have authorization access for any element. But other APIs that are called by these APIs must go through the authorization check.

To include the apisecurity file in the documentation, and package it in the EAR (enterprise archive) file, do the following:

1. Place the extensions in the `<INSTALL_DIR>/xapidocs/extn/input` directory.
2. Rebuild the documentation (including the apisecurity files) by running the following command:

   ```
   deployer.sh -t xapideployer -l info [new target]
   ```

3. Build the EAR file. The apisecurity template files will be packaged from:

   ```
   <INSTALL>/repository/xapi/template/merged/apisecurity
   ```

Access to API security and the permission level are controlled in the following properties in the yfs.properties file. All authorization failures are logged to a logging category named sci.apisecurity.

- api.security.enabled
  - Y (default)—Enable API security

  - N—Do not enable API security
- api.security.mode
  - STRICT—If any validation fails, throw an exception. This is appropriate for production systems, if all permissions are configured properly.
  - LAX—Filter out and log invalid input, but continue processing. The filtering allows the system to mostly work despite incorrect input or output, while the logging helps to identify places that need change.

    Note: The system may still throw an exception when the filtering produces an ambiguous behavior.

  - DEBUG—Log invalid input and output, but do not filter anything or throw exceptions. This is only appropriate during initial development, to identify the permissions required by various processes.

If you do not specify a security mode, there is no filtering, thrown exceptions, or authorization checking. There is limited logging.

- api.security.override.<apiName>.mode

  Use this setting to override permissions on individual APIs. This property uses the same values as api.security.mode.
- api.security.smc.enabled
  - Y—Enable API security for the Applications Manager
  - N (default)—Do not enable API security for the Applications Manager
- api.security.console.enabled
  - Y—Enable API security for the Application Console
  - N (default)—Do not enable API security for the Application Console

When upgrading, you should initially disable this feature and grant all access through properties. In an upgraded system, you can phase in this feature by enabling security one API at a time, as you define and test

permissions. If enabled, only the system user group has grant permission to the APIs; for all other custom user groups, appropriate permission has to be given. For information about user group permissions, see the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

3.5 Date and Time Handling by APIs

Selling and Fulfillment Foundation handles values for both date-time and date. Date-time refers to values that contains a date and time component, where Date refers to values that contain only a date component.

Date values can be made nullable by specifying `Nullable="true"` in the entity XML. Thereby the Date values in the table is blanked out. The expected behavior of a date column is marked as Y is described in Table 3–1.

Table 3–1 Nullable Date Behavior

Action	Description
Insert	When the field is not populated in a database object (is null), the database infrastructure automatically inserts a null value into the column in the database.
Update	When the application nulls out a date, it sets the corresponding field to null value in the database.
Select or List	When a column is defined as nullable and the date from the database is returned as null, it is automatically nulled out. So, the corresponding get method returns a null.
Search by date	Can pass null value as needed when specified to do so.

Note: If you have specified the date value as 01/01/2400 in versions prior to Release 8.5, those values are now treated as null. The dates with special significance are:

- Null date - 01/01/2400
- High date - 01/01/2500
- Low date - 01/01/1900

3.5.1 Specifying Time Zones

Dates and times are time zone aware. Time zones are relative to the Coordinated Universal Time (UTC).

For example, if an order is created on the system on 06/15/2003 at 16:00:00 in New York, (USA/New York time zone) a user in Chicago who examines that the order observes that the order creation date-time as 06/15/2003 at 15:00:00, (USA/Chicago time zone).

For a time published from Boston that is -5:00 hours from UTC, the string literal "-5:00" is appended to the current date-time attribute published from APIs. The input "2003-04-23T14:15:32-05:00" gives the date, time, and time zone reference for a transaction.

The `yfs.install.localecode` parameter in the `yfs.properties` file determines the Selling and Fulfillment Foundation time zone. For example, `yfs.install.localecode=en_US_EST`

To configure the Selling and Fulfillment Foundation time zone, set the `yfs.install.localecode` property to `en_US_EST` in the `<INSTALL_DIR>/properties/customer_overrides.properties` file. For additional information about modifying properties and the `customer_overrides.properties` file, see the *Selling and Fulfillment Foundation: Installation Guide*.

3.5.2 Using Date-Time Syntax

All APIs, user exits, and events that use date-time fields have a uniform syntax (a combination of the basic and extended formats of the ISO 8601 specification). This syntax is the expected format for all input as well as output.

Date Only Syntax

YYYY-MM-DD

Date-Time Syntax

YYYY-MM-DDTHH**:**MI:SS±HH:MM

Values in bold are placeholders for literals. For example, the format for March 5, 2003, 11:30:59 p.m. is 2003-03-05T23:30:59.

> **Note:** This syntax is an ISO Date-Time syntax and not the database syntax. Using a syntax other than the ISO Date-Time format may cause problems. For example, the time element in the Date-Time syntax may be overlooked or calculated incorrectly.
>
> For example, if you provide the Date-Time input as "2007-05-18-19.10.28.000000", the system may interpret it as just "2007-05-18" because the T symbol is missing in the input.

Syntax Parameters

YYYY - Required. Four-digit year. Used in both date-time and date fields.

MM - Required. Two-digit month. Used in both date-time and date fields.

DD - Required. Two-digit day of the month. Used in both date-time and date fields.

T - Required. The literal value T, which separates the date and time component. Used only in date-time fields.

HH - Required. Two-digit hour of the day. For example, 11 p.m. is displayed as 23:00:00. Used only in date-time fields.

MI - Required. Two-digit minutes of the hour. For example, 59 minutes is displayed as 00:59:00. Used only in date-time fields.

SS - Required. Two-digit seconds of the minute. For example, 21 seconds is displayed as 00:00:21. Used only in date-time fields.

±**HH:MI** - Optional. Two-digit hours and minutes, separated by a colon (":"). Indicates how many hours from UTC, using - to indicate earlier than UTC and + to indicate later than UTC. If this value is not passed in input, the time zone of Selling and Fulfillment Foundation are assumed.

4

Input XML Files for APIs

4.1 About Input XML Files for APIs

APIs retrieve data using input XML files that define which records need to be selected or used. When extending the database to include additional fields, you need to also extend the input XML to populate those fields.

> **Caution:** Do not pass a blank element (an element containing all the attributes with blank values) to an API. Also, do not pass attributes that have leading or trailing spaces. The result of either situation is not predictable.

Example 4–1 shows an input XML modification.

Example 4–1 Example of Input XML Modification

The following example modifies the input XML file for the YFS_createOrder() API:

```
<Orders AuthenticationKey="">
   <Order EnterpriseCode="DEFAULT" OrderNo="DB04" OrderName="DB04"
OrderDate="20010803" OrderType="Phone" PriorityCode="1" PriorityNumber="1"
ReqDeliveryDate="20010810" ReqCancelDate="" ReqShipDate="20010810"
SCAC="FEDEX" CarrierServiceCode="Express Saver Pak"
CarrierAccountNo="112255" NotifyAfterShipmentFlag="N" NotificationType="FAX"
NotificationReference="" ShipCompleteFlag="N" EnteredBy="Iain "
ChargeActualFreightFlag="Y" AORFlag="Y" SearchCriteria1="Search"
SearchCriteria2="Search Again" >
      <OrderLines>
         <OrderLine PrimeLineNo="1" SubLineNo="1" OrderedQty="1"
ReqDeliveryDate="20010810" ReqCancelDate="20010810" ReqShipDate="20010810"
SCAC="FEDEX" CarrierServiceCode="Express Saver Pak" PickableFlag="Y"
```

```
HoldFlag="N" CustomerPONo="11" >
            <Extn ExtnAcmeLineType="Type1"/>
            <Item ItemID="ITEM1" ProductClass="A" ItemWeight="1"
ItemDesc="paintball gun" ItemShortDesc="pball gun" UnitOfMeasure="EACH"
CustomerItem="Spectra Flex" CustomerItemDesc="GEGRG" SupplierItem="Spectra
Flex @ supplier" SupplierItemDesc="Spectra Flex Desc @ supplier"
UnitCost="15.99"  CountryOfOrigin="CA"/>
           <PersonInfoShipTo Title="Mr" FirstName="Quigley" MiddleName="Al"
LastName="Johns" Company="Company" JobTitle="Project Clert"
AddressLine1="Address Line 1 -3 Main Street" AddressLine2="ShipTo Address
line 2" AddressLine3="ShipTo Address line 3" AddressLine4="ShipTo Address
line 4" AddressLine5="ShipTo Address line 5" AddressLine6="ShipTo Address
line 6" City="Acton" State="MA" ZipCode="01720" Country="US"
DayPhone="978-635-9242" EveningPhone="978-635-9252"
MobilePhone="978-888-8888" Beeper="" OtherPhone="other555-5555" DayFaxNo=""
EveningFaxNo="" EMailID="jquigley@maine.com"
AlternateEmailID="hfournier@ontario.com" ShipToID=""/>
         </OrderLine>
         <NumberOfOrderLines/>
      </OrderLines>
     <PersonInfoShipTo Title="MR" FirstName="s" MiddleName="X" LastName="T"
Suffix="T" Department="T" Company="SD" JobTitle="SS" AddressLine1="SS"
AddressLine2="SS" AddressLine3="SS" AddressLine4="SS" AddressLine5="SS"
AddressLine6="SS" City="REDWOOD" State="CA" ZipCode="01852" Country="USA"
DayPhone="3456789234" EveningPhone="3456789234" MobilePhone=""
EveningFaxNo="SS" />
       <PersonInfoBillTo Title="mj" FirstName="m" MiddleName="JJ"
LastName="KK" Suffix="lll" Department="l" Company="kj" JobTitle="k"
AddressLine1="HJHKK" AddressLine2="HJKHK" AddressLine3="HKHJ"
AddressLine4="" AddressLine5="" AddressLine6="" City="UUU" State="IUI"
ZipCode="78787" Country="USA"  />
  </Order>
  <NumberOfOrders/>
</Orders>
```

Important: In order for the factory setup scripts to operate properly, when you add a column to a database table, be sure that the column is not null and that it has a default value. If you need to make the column nullable, the default value must not be present.

Also, when you are specifying XML Name and XML Group, keep in mind that the values should be valid Document Object Model (DOM) strings. (The values must not contain spaces or special characters that are not supported by the DOM specification.)

The following example XML file adds a column to the YFS_ORDER_LINE table:

```
<?xml version="1.0" encoding="UTF-8" ?>
<DBSchema>
  <Entities>
    <Entity TableName="YFS_ORDER_LINE">
      <Attributes>
        <Attribute ColumnName="EXTN_ACME_LINE_TYPE" DecimalDigits="" Default
         Value="' '" Size="10" Type="CHAR" XMLGroup="Extn"
         XMLName="ExtnAcmeLineType"/>
      </Attributes>
    </Entity>
  </Entities>
</DBSchema>
```

4.2 Guidelines for Forming API Input

When coding API input parameters, follow the guidelines in this section for using literals and formatting API input.

Do not pass a blank element (an element containing all the attributes with blank values) to an API. Also, do not pass attributes that have leading or trailing spaces. The result of either situation is not predictable.

4.2.1 Using Literals in Maps and XMLs

Use literals in maps and XMLs. Using literals enables you to write code with fewer bugs because the compiler catches the use of incorrect names in the <name>=<value> pair. In addition, using literals simplifies the

maintenance of your code; if you change the <name>, all you need to do is recompile your code instead of editing one or more <name>s within it first.

4.2.2 Using Special Characters

The fields that are a part of the logical key for any record in the Selling and Fulfillment Foundation schema (such as OrganizationCode and OrderNo) have some restrictions. For such fields, Selling and Fulfillment Foundation does not support the use of special characters listed in Table 4–1.

Table 4–1 Special Character Descriptions

Special Character	Description
&	Ampersand
>	Greater Than
<	Less Than
%	Percent
"	Quotation Mark
+	Plus sign
'	Apostrophe
(	Parenthesis
)	Parenthesis
[	Square Brackets
]	Parenthesis

Note: You can use the plus (+) and ampersand (&) signs only in the ItemID field.

In addition, Sterling Commerce recommends against using third-party vendors' reserved special characters. For example, in certain situations, data with underscore characters ("_") on an Oracle database could result in unexpectedly slow query performance because the database deciphers the underscore as a single character wild-card.

The following fields have no restrictions and support all characters:

- All description fields (for example, item description)
- All name fields (for example, organization name)
- All address fields (for example, billing address)

> **Note:** However when creating address fields through the UI, the information entered after the quotation mark (") is truncated and appears as a new entry in YFS_PERSON_INFO table. To work around this problem, use apostrophe (') instead of quotations.

- All instruction fields (for example, gift wrapping)
- All text fields (for example, reasons and comments)

> **Note:** The Selling and Fulfillment Foundation Mobile Device does not support the use of the ampersand (&) character.

4.2.3 XML-Based APIs

The following table lists all of the following special characters that should follow the XML escape format.

Table 4–2 Special Characters in Attributes of XML-Based APIs

For This Character	Enter This Sequence
quotation mark (")	&quot;
single quotation (')	&apos;
greater than symbol (>)	&gt;
less than symbol (<)	&lt;
ampersand (&)	&amp;

4.2.4 Support for CreateTS and ModifyTS in Input and Output XML Files

CreateTS and ModifyTS can be used in getAPIs(input or output) if an entity in the input or output XML file for which these attributes are requested have corresponding tables. These attributes indicate when a record was created or modified in the database.

4.3 Forming Queries in the Input XML of List APIs

The input XML of list APIs enable queries on conditions such as *starts with*, *contains*, *is greater than*, and so forth. Example 4–2 shows a fragment of the input XML that returns a list of items at a specific shipping node that fall within a specific weight range and to be shipped during a specific date range.

Example 4–2 getOrderList API Input XML with Query Type Values

```
<Order ReqShipDateQryType="DATERANGE" FromReqShipDate="20010113"
ToReqShipDate="20030113" /Order>
<OrderLine ShipNode="Atlantic" /OrderLine>
<Item ItemWeightQryType="BETWEEN" FromItemWeight="2" ToItemWeight="20"/>
<OrderRelease CarrierServiceCodeQryType="FLIKE" CarrierServiceCode="Priority" />
```

Note: Some APIs do not support QryType for an input attribute. These APIs are:

- getAssignedPricelistHeaderList
- getCarrierServiceList
- getNodeSCACAccountList
- getOrderLineStatusList
- getPaymentStatusList
- getPriceListForOrdering
- getQueryTypeList
- getReceiptLinesForTransaction
- getRegionList
- getServerList
- getShipmentListForOrder

- getSurroundingNodeList
- getTaskQueueDataList
- getTraceableComponentList
- getTraceList

getZoneListForDiscount

To form queries:

1. Edit the custom input XML of any list API, and append QryType to any attribute you want to query on. Any attribute that is not appended with QryType can also be queried on, using the default query type value EQ, as shown for ShipNode in Example 4–2.
2. For attributes appended with QryType, specify a query type value from Table 4–3. This is case sensitive.
3. Specify the values that are applicable to your search criteria.

The values for the QryType attributes vary depending on the datatype of the field. Table 4–3 lists the supported query type values for each datatype.

Table 4–3 Query Type Values Used by List APIs

Field DataType	Supported Query Type Values
Char/VarChar2	• EQ - Equal to • FLIKE - Starts with • LIKE - Contains • GT - Greater than • LT - Less than
Number	• BETWEEN - Range of values • EQ - Equal to • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to

Table 4–3 Query Type Values Used by List APIs

Field DataType	Supported Query Type Values
Date	• DATERANGE - Range of dates • EQ - Equals • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to
Date-Time	• BETWEEN - Range of dates • EQ - Equals • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to
Null	• ISNULL - Return records that are null. • NOTNULL - Return records that are not null. **Note:** These two query types are used when the column or attribute is set to Nullable in the entity XML.

4.3.1 Setting Query Timeouts for XAPIs

You can add individual query timeouts for APIs. To do this, specify the query timeout value in seconds for the API in the in the API's input XML. For example:

```
<ApiInput QueryTimeout="10">
...
...
...
</ApiInput>
```

Note: The value of the QueryTimeout attribute overrides the value of the yfs.ui.queryTimeout property in the yfs.properties file. But the value of this attribute is valid only for a single API call. After the API execution is complete, the query timeout is set to the old value based on the value of the yfs.ui.queryTimeout property in the yfs.properties file.

4.4 Sorting Through OrderBy Element in the Input XML of List APIs

The input XML of list APIs supports sorting based on the OrderBy element. You can also do nested sorting using the OrderBy element. The OrderBy element supports ordering of the attributes in both Ascending and Descending order. By default the results are sorted in the Ascending Order.

Example 4–3 shows a fragment of the input XML that returns a list of organizations and results are sorted by the `OrganizationName` attribute.

Example 4–4 shows a fragment of the input XML that returns a list of organizations and results are sorted by `OrganizationName` and `LocaleCode` attributes.

Example 4–3 getOrganizationList API Input XML with OrderBy Element

```
<Organization IgnoreOrdering="N" MaximumRecords="5000">
<OrderBy>
<Attribute Name="OrganizationName"/>
</OrderBy>
</Organization>
```

Example 4–4 getOrganizationList API Input XML with Nested OrderBy Element

```
<Organization IgnoreOrdering="N" MaximumRecords="5000">
<OrderBy>
<Attribute Name="OrganizationName"/>
<Attribute Name="LocaleCode"/>
</OrderBy>
</Organization>
```

To form queries:

Edit the custom input XML of any list API, and add the `OrderBy` element. Add the `Attribute` child element and in the `Name` attribute specify the name of the field based on which you want to sort the results. You can also perform nested sorting using the OrderBy element as shown in Example 4–4.

By default, the results are sorted in the Ascending order. If you want to sort the results in descending order add the Desc attribute to the `Attribute` element and set it to Y.

Example 4–5 shows a fragment of the input XML that returns a list of organizations and results are sorted by the `OrganizationName` attribute in the descending order.

Example 4–5 getOrganizationList API Input XML with OrderBy Element and Desc Attribute

```
<Organization IgnoreOrdering="N" MaximumRecords="5000">
<OrderBy>
<Attribute Name="OrganizationName" Desc="Y"/>
</OrderBy>
</Organization>
```

5

Output XML Files for APIs

5.1 About Output XML Files and Templates for APIs

APIs return data using two types of output XML files that define which elements and attributes are required by an API.

- Output XML File - Defines the outer limits of the data an API can return. Do **not** modify output XML files.
- Template XML File - Defines the data returned by an API for the record specified in the input XML file and restricts the amount of data to a subset of the output XML. You can modify this file to incorporate a subset of the attributes and elements from the output XML.

5.1.1 Output XML Templates

Many APIs use a corresponding output template. The output template is in XML format and is read in by an API in order to determine the elements and attributes for which it should return. The standard output template defines the elements and attributes returned for any specific API. (To see the entire range of possible values an API can return, see its output XML in *Selling and Fulfillment Foundation: Javadocs*.) The standard template can be a subset of the entire range of values returned, as determined by the output XML in the *Selling and Fulfillment Foundation: Javadocs*.

Note: Ensure that when adding elements and attributes to the output template, use ***only*** those that are documented in the *Selling and Fulfillment Foundation: Javadocs*. While the APIs can output additional elements and attributes, only those that are documented in the *Selling and Fulfillment Foundation: Javadocs* are supported.

For example, the standard output template of the `getOrderList()` API returns the header-level information of an order and the standard output template of the `getOrderDetails()` API returns in depth information about an order.

Besides the standard output XML template, you can create custom output templates for APIs to use for your own business requirements, such as different output for different document types.

5.1.2 Document Types

If you use a variety of business-related document types such as orders, planned orders, purchase orders, and returns, you can use custom templates that enable an API to return the values that pertain to each unique document type.

For example, you can use one template with the `getOrderDetails()` API to return information about Planned Orders and another template for the `getOrderDetails()` API to return different information about Orders.

5.1.3 Standard Output Template Behavior

The set of values that the standard output template returns covers a variety of business scenarios. With such a large range of possibilities, an API using the standard output template may return much more data than you need for your business purposes (and take much more time to process than you prefer).

If you want to customize the information returned by an API, you can do so by creating and using a custom template, using our guidelines and procedures.

5.2 Extending an Output XML Template

Many APIs use an output XML template to define what is returned. Each API has its own XML template, which is picked up from the <INSTALL_DIR>/repository/xapi/template/merged/api/<apiName>.xml file. The files in this directory are part of the product and should not be altered. However, these templates can be overridden by implementing template extensions.

To extend a template file:

1. Copy the template <INSTALL_DIR>/repository/xapi/template/merged/api/<apiName>.xml file, to <INSTALL_DIR>/extensions/global/template/api/ directory, keeping the same file name.
2. Modify the copied file, as needed. To extend a template file, add the Extn tag under the entity tag. For example, if you have added a column EXTN_COLOR to YFS_ITEM table, you also must add the tag Extn under the tag Item in the getItemDetails.xml file as follows:

```
<Item ItemKey="".....>
    <PrimaryInfo MasterCatalogID="" .../>
    ...
    <Extn ExtnColor=""/>
</Item>
```

Note: If you are extending an output XML template, place your extended files in the <INSTALL_DIR>/extensions/global/template/api directory (create this directory structure if necessary).

But when providing the name of the template api file during service definition, the path should be /global/template/api/<CUSTOM-TEMPLATE-API>.

5.3 Best Practices for Creating Custom Output XML Templates

Whenever you call an API, you need to pass your own customized template, not the sample provided by Selling and Fulfillment Foundation. This section helps guide your decision-making processes in planning how to design custom output templates.

In general, when you customize an output template, you do so by editing a copy of the standard template. You cannot modify the standard output template.

There are two ways of customizing and calling the output template. Which function you choose depends on the size and type of the data set you want returned by the API.

5.3.1 Gather Information Relevant to the API

Custom output templates provide the flexibility to return whatever data you wish, so it is important to understand that it is possible to modify an output template in such a way that it returns information that is not quite relevant to the API.

For example, it is possible to modify the output template of the `getOrderList()` API in such a way that it returns detailed information about an order rather than just header-level information. You should modify an output template in such a way that it takes advantage of the unique aspects of its corresponding API. Keeping each template unique to its API prevents any ambiguity about which API to use in any specific situation.

5.3.2 Gather Information Relevant to Your Business Needs

Since the standard output template returns all attributes, even for empty elements in the template, you might want to tailor information to your specific business needs. If you don't exclude the attributes you don't require, you receive more data than you need and the extra data may slow the performance of the API.

For example, if you are using the `getOrderDetails()`API to return only `OrderLine` attributes but your custom output template includes `Schedule` attributes, all attributes for `OrderLine` and `Schedule` are returned.

5.3.3 Choose an Appropriate Template Mechanism

In general, the format of any template should follow the same structure as the standard template. Keeping this general rule in mind, there are two ways to customize the standard template, differentiated by the amount of data they return and how they can be called:

- Static templates
- Dynamic templates

Static templates provide the ability to add new elements but not remove any of the defaults. A static template is pervasive, as it is picked up by default by an API whenever that API is invoked.

Dynamic templates provide the ability to add new elements and remove any of the default elements from the standard template. A dynamic template is an instance, as it is picked up only for a specific API call, such as when configured to do so during user interface extensibility.

A comparison of the differences between the two types of template mechanisms is summarized in Table 5–1.

Table 5–1 Comparison of API Output Template Mechanisms

Template Types	Allowed XML Elements	Behavior
Static Template	Default template elements cannot be removed. New elements can be added.	Pervasive. Picked up by default by an API.
Dynamic Template	Default template elements can be removed. New elements can be added.	Instance. Picked for a specific API call, as configured during user interface extensibility.

Choose which of these mechanisms best fits your business needs and adhere to it.

Remember that when you define a dynamic template, all possible values are returned. In order to return the smallest amount of data for an element, when you are pruning away elements you don't need, you need to include its parent with at least one of its attributes.

If you leave an element blank or include unwanted attributes in the parent element all values are returned, as illustrated in Example 5–1.

Example 5–1 A Poorly Pruned Dynamic Template

```
<!-- getOrderDetails Output XML -->
<Order>
    <OrderLines>
    <!--1 or more order line-->
        <OrderLine>
```

```
            <Item CountryOfOrigin="" ItemDesc="" ItemID=""/>
            <Schedules>
            <Schedule Attr1 ..... />
            </Schedules>
        </OrderLine>
    </OrderLines>
<Order>
```

Since Example 5–1 specifies all OrderLine attributes as well as a few Item and Schedule attributes, the API returns values similar to those in Example 5–2.

Example 5–2 Values Returned by a Poorly Pruned Dynamic Template

```
<OrderLine AllocationDate="03/28/2002" CarrierAccountNo="112233"
CarrierServiceCode="Next Day Air" Createprogid="SterlingTester"
Createts="03/28/2002" Createuserid="SterlingTester" CustomerLinePONo="999"
CustomerPONo="111" DeliveryCode="AIR" DepartmentCode="Clothing" ExtendedFlag=""
ExternalReference1="" ExternalReference2="" ExternalReference3=""
ExternalReference4="" ExternalReference5="" FreightTerms="Buyer" HoldFlag="N"
HoldReasonCode="HoldReas" ImportLicenseExpDate="08/08/2002"
ImportLicenseNo="225588" InternalReference1="" InternalReference2=""
InternalReference3="" InternalReference4="" InternalReference5="" KitCode=""
LineClass="" LineSeqNo="1.1" LineType="Single" Lockid="1" MarkForKey=""
Modifyprogid="SterlingTester" Modifyts="03/28/2002"
Modifyuserid="SterlingTester" OrderClass="NEW"
OrderHeaderKey="200203281036245174" OrderLineKey="200203281036245175"
OrderedQty="5.00" OrigOrderLineKey="" OriginalOrderedQty="5.00"
OtherCharges="0.00" OtherChargesPerLine="0.00" OtherChargesPerUnit="0.00"
PackListType="Bill" PersonalizeCode="PersCode" PersonalizeFlag=""
PickableFlag="Y" PricingDate="01/01/2500" PrimeLineNo="1" Purpose="Purpose"
ReceivingNode="B1N1" ReqCancelDate="01/01/2500" ReqDeliveryDate="04/04/2002"
ReqShipDate="03/30/2002" ReservationID="" ReservationPool="" SCAC="UPS"
ShipNode="E1N1" ShipToID="" ShipToKey="" ShipTogetherNo="Y" SplitQty="0.00"
SubLineNo="1" TotalDiscountAmount="0.00" TotalOtherCharges="0.00">
<Item CountryOfOrigin="" ItemDesc="" ItemID=""/>
<Schedules>
<Schedule ExpectedDeliveryDate="" ExpectedShipmentDate="" TagNumber=""
OrderHeaderKey="" OrderLineKey="" OrderLineScheduleKey="" ScheduleNo=""
ShipByDate="" Quantity=""  PromisedApptStartDate="" PromisedApptEndDate=""/>
</Schedules>
</OrderLine>
</OrderLines>
</Order>
```

In Example 5–3, the dynamic template has been trimmed down, keeping in mind the following guidelines:

- The structure of the custom output template mirrors the structure of the standard output template.
- Excess elements (regarding kits, schedules, addresses, and so forth) are pruned away.
- Parent elements are populated with one attribute in order to suppress excess detail. For example, specifying the OrderNo attribute for the Order element suppresses all of the other Order attributes.

Example 5–3 A Carefully Pruned Custom Output Template

```
<!-- getOrderDetails Output XML -->
<Order OrderNo="">
    <OrderLines>
        <!--1 or more order line-->
        <OrderLine PrimeLineNo="">
            <Item CountryOfOrigin="" ItemDesc="" ItemID=""/>
        </OrderLine>
    </OrderLines>
</Order>
```

Since Example 5–3, "A Carefully Pruned Custom Output Template" specifies only a few Item attributes, and only one attribute for its parent element, the getOrderDetails() API returns only the values shown in Example 5–4.

Example 5–4 Values Returned by a Carefully Pruned Output Template

```
<?xml version="1.0" encoding="UTF-8" ?>
<Order OrderNo=Y00000765>
    <OrderLines>
        <OrderLine PrimeLineNo="1">
            <Item CountryOfOrigin="IN" Item Description" ItemDesc="Green Sari"
            ItemID="GNSARI5LT" />
        </OrderLine>
        <OrderLine PrimeLineNo="3">
            <Item CountryOfOrigin="CA" ItemDesc="Pink Scarf" ItemID="PKSCARF4LT"
            />
        </OrderLine>
    </OrderLines>
</Order>
```

This method of pruning the templates improves the performance as database access to order schedules and other unwanted elements has been prevented.

5.3.4 Develop Useful Templates

The supplied templates located in the <INSTALL_DIR>/repository/xapi/template/merged/api/ directory are sample guides. Use them to help you develop your own output XML templates. Using your own customized templates gives you much more flexibility, greater performance, and more assurance of appropriate data output. You can either pass your template through the env or put it in the extension folder.

5.3.5 Keep Performance Needs in Mind

Besides tailoring the templates to your business needs, it is important to keep technological considerations in mind. For performance-related information about using output, see the *Selling and Fulfillment Foundation: Performance Management Guide*.

5.4 Defining and Deploying a Static Template for Output XML

If you want to use a template that has more elements in addition to those in the standard output template, create a static output template. This function enables you to create a template that includes all of the elements in the standard output template *plus* any new ones you add. For example, you may need to add UI fields for any database columns you have added. Note that if you use this function, you *cannot* remove any elements that exist in the standard template.

To define and deploy a static template:

1. Copy the standard output template for the API that you want to modify from the <INSTALL_DIR>/repository/xapi/template/merged/api/<FileName>.xml file to <INSTALL_DIR>/extensions/global/template/api/<FileName>[.<DocType>].xml.

- Keep the file name of your new template the same as the standard template.

  The name of the output template corresponds with the name of the API or event associated with it. For example, the `getOrderDetails()` API takes the output template file `getOrderDetails.xml`.

- If the template references a document type, include the document type code in the filename.

  For example, to create an output template for the `getOrderDetails()` API for an Order (`0001`) document type, the name of the template XML is `getOrderDetails.0001.xml`.

2. Modify the copied template in the `/extensions/global/template/api/` directory as required, keeping in mind the guidelines described in Section 5.3, "Best Practices for Creating Custom Output XML Templates" on page 5-33

   Note: You may add any elements you wish, but you cannot remove any of the elements present in the standard output template.

3. Call the API as typical and it automatically picks up the custom output template from the directory containing the custom templates.

5.5 Defining and Deploying a Dynamic Template for Output XML

If you want to use a template that contains a subset of the elements in the standard output template, create a dynamic output template. If you want the ability to remove some elements from the standard template and perhaps add your own elements, you do that by passing your XML data or a file name into the YFSEnvironment object.

To define and deploy a dynamic template:

1. Copy the standard output template for the API that you want to modify from the `<INSTALL_DIR>/repository/xapi/template/merged/api/<FileName>.xml` file to

<INSTALL_DIR>/extensions/global/template/api/extn_<FileName>.xml.

When naming your new template file, use the same name as the standard template and you *must* prefix it with extn_ to indicate that it is an extension.

The name of the output template corresponds with the name of the API or event associated with it. For example, the getOrderDetails() API takes the output template file getOrderDetails.xml.

2. Modify the copied template in the /extensions/global/template/api directory as required, keeping in mind the guidelines described in Section 5.3, "Best Practices for Creating Custom Output XML Templates" on page 5-33.

3. During user interface extensibility, call the setApiTemplate() function on the YFSEnvironment object. This enables you to specify an output template before calling an API, using one of the following functions:

   – XML ***data*** as a variable - as in the following example:

```
YFSEnvironment env = createEnv();
Document doc = getTemplateDocument();
env.setApiTemplate("getOrderDetails", doc);
private YFSEnvironment createEnv() {
//create new environment by passing the user id, program id, etc.
}
private Document getTemplateDocument() {
//create a Document object containing the desired template XML.
}
```

   – XML ***file*** as a variable - as in the following example:

```
YFSEnvironment env = createEnv();
env.clearApiTemplates();
env.setApiTemplate("getOrderDetails", "extn_myOrderDetails");
private YFSEnvironment createEnv() {
//create new environment by passing the user id, program id, etc.
}
```

The API then uses the template passed in through YFSEnvironment to produce the output XML document. For details about the YFSEnvironment interface, see the *Selling and Fulfillment Foundation: Javadocs*.

5.6 Sequence of Precedence for Output XML Templates

Since Selling and Fulfillment Foundation enables you to define multiple types of templates, in addition to the standard templates that you cannot modify, it is important to understand the order of precedence in which Selling and Fulfillment Foundation implements when reading the templates. This section describes how Selling and Fulfillment Foundation uses the API and event templates.

5.6.1 API Templates

Table 5–2, "Output Template Order of Precedence" shows the sequence of precedence for determining which output template is used by an API. Events use a similar sequence of precedence, using the directories described in Section 5.6.2, "Event Templates" on page 5-42. Note that templates are not supported for user exits.

Table 5–2 Output Template Order of Precedence

Priority	Output Template Path and File Name
1	setApiTemplate(<file>|<xmlDocument>) to YFSEnvironment When a file is specified, it is picked up from the <INSTALL_DIR>/extensions/global/template/api directory.
2	<INSTALL_DIR>/extensions/global/template/api/apiName.docType.xml
3	<INSTALL_DIR>/extensions/global/template/api/apiName.docType.xml (Selling and Fulfillment Foundation sample template; not for use in production.)
4	<INSTALL_DIR>/extensions/global/template/api/apiName.xml
5	<INSTALL_DIR>/extensions/global/template/api/apiName.xml (Selling and Fulfillment Foundation sample template; not for use in production.)

5.6.2 Event Templates

Event templates help determine what elements and attributes should be present in the output XML of an event. For events raised, see the relevant transactions in the *Selling and Fulfillment Foundation: Javadocs*.

To see which events take output templates, see the files in the `<INSTALL_DIR>/repository/xapi/template/merged/event/` directory. These templates can be overridden by files you place in the `<INSTALL_DIR>/extensions/global/template/event` directory.

The naming convention for templates is BaseTxnName.eventName.xml. For example, the on_success event of the `createOrder()` API uses the ORDER_CREATE.ON_SUCCESS.xml event template.

Note that templates are not supported for user exits.

6

DTDs, XSDs, and Complex Queries

6.1 DTD and XSD Generator

Every Selling and Fulfillment Foundation API uses standard input, output, and error XMLs. These XMLs conform to the related Document Type Definition (DTD). For example, consider the following XML:

```
<?xml version="1.0" encoding="UTF-8">
<Order EnterpriseCode="DEFAULT" OrderNo="S100" />
```

The corresponding DTD for this XML is:

```
<!ELEMENT Order>
<!ATTLIST Order OrderNo CDATA #IMPLIED>
<!ATTLIST Order EnterpriseCode CDATA #REQUIRED>
```

To create such DTDs for the extended Selling and Fulfillment Foundation XML, a tool called the `xsdGenerator.xml` is provided in the `<INSTALL_DIR>/bin` directory. This tool converts a specially-formatted XML file into a DTD and XML schema definition (XSD). The command for running the tool is:

```
sci_ant.sh -f xsdGenerator.xml generate
```

You can also configure the following properties in the `<INSTALL_DIR>/properties/customer_overrides.properties` file:

- `xsdgen.use.targetnamespace`
- `xsdgen.use.datatypeimport`

For additional information about modifying properties and the `customer_overrides.properties` file, see the *Selling and Fulfillment Foundation: Properties Guide*.

Table 6–1 XSD Generator Properties

Fields	Description
xsdgen.use.targetnamespace	Optional. The default value is Y. If set to Y, the XSD files are generated with a defined target namespace.
xsdgen.use.datatypeimport	Optional. The default value is Y. If set to Y, all the XSD files reference a single common XSD file containing all the common data type definitions. If set to N, each XSD file is created with a copy of the database definitions embedded within it.

The input XML files should be placed in the <INSTALL_DIR>/xapidocs/extn/input directory. The resulting DTD and XSD files are placed in the <INSTALL_DIR>/xapidocs/extn/output/dtd and <INSTALL_DIR>/xapidocs/extn/output/xsd directories respectively.

Consider the following sample XML that can be placed in the input directory:

Example 6–1 Sample XML for Converting to an XSD and DTD

```
<Item yfc:DTDOccurrence="REQUIRED" ItemKey="" ItemID="REQUIRED"
OrganizationCode="REQUIRED" UnitOfMeasure="">
    <PrimaryInformation Description="" ItemType="" />
    <AdditionalAttributeList>
        <AdditionalAttribute Name="" Value=""/>
    </AdditionalAttributeList>
    <Extn ExtnAttr1="" ExtnRefId="">
        <CSTItemDataList yfc:DTDOccurrence="ZeroOrOne">
            <CSTItemData yfc:DTDOccurrence="ZeroOrMany" ItemDataKey=""
            Description="">
            <CSTItemExtraData yfc:DTDOccurrence="ZeroOrOne" CodeType=""
            DataType="" />
            <YFSCommonCode yfc:DTDOccurrence="REQUIRED" CodeName="" CodeType=""
            CodeValue="" />
            </CSTItemData>
        </CSTItemDataList>
    </Extn>
</Item>
```

Table 6–2 Special Attributes in your XML

Fields	Description
yfc:QryTypeSupported	This attribute determines whether or not the query type functionality is supported for the attributes in this element. If set to Y, it takes effect for all the elements.
yfc:ComplexQuerySupported	This attribute specifies whether or not a complex query type is supported. This attribute can only be present in the root element.
yfc:XSDType	The name of the type to use for the root element schema definition.
yfc:DTDOccurrence	This attribute can contain any of the following values: • REQUIRED - This element must be present if the parent element is present. • ZeroOrOne - This element is optional, but may occur only once. • ZeroOrMany - This element is optional, but may occur multiple times. • OneOrMany - This element is required, and may occur multiple times.
yfc:UseEntityOrdering	This attribute determines whether or not all the first-level children of an element are ordered in the sequence they are found in the entity xmls. This attribute can contain any of the following values: • true - All the first-level children of an element are ordered in the sequence they are found in the entity xmls. • false - The first-level children of an element are not ordered in the sequence they are found in the entity xmls.
xmlns	The namespace to use for the targetNameSpace in the output XSD. This attribute takes effect only if it is present in the root element.

The attributes with values of REQUIRED are generated as required attributes in the DTD and XSD. However, an existing required attribute cannot be marked as optional.

The attribute values can also be specified to supply additional constraints. A list of options is separated by a vertical bar (|). The value of the attribute must be one of the given options. This is only supported

for data types based on the strings. The values are trimmed of the whitespace character if the value itself is entirely spaces, in which case the enumerated option remains unchanged.

For example, SomeAttr="A | B | C | |" results in valid options of "A", "B", "C", " ", and "".

> **Note:** The DTDs do not support enumerated values containing only whitespace characters. Therefore, restrictions of this type cannot be represented in the DTD.

The default input and output XMLs that can act as a base for your custom XML are located in the <INSTALL_DIR>/xapidocs/xmlstruct/ directory. Also note that the DTDOccurrence and REQUIRED data provided for the standard tables are inferred from the base file in the xmlstruct directory and do not need to be supplied. If they are provided, the existing information is overridden by any new information present in the custom XMLs. Any required datatype and relationship information are obtained from the entity XMLs.

> **Note:** Do not put your custom XMLs in the xmlstruct directory.

Therefore, when the tool is run these base XML files serve as a default to your custom XML files, which need only contain the changes made by you such as the extended elements and attributes. This allows future upgrades to safely modify the XML files in the xmlstruct directory. Re-running the XSD generation tool automatically picks up these updates.

The appropriate XML file in the xmlstruct directory associated with your custom XML is identified by the file name. Your custom XML may start with an optional prefix followed by an under-score and the base file name. For example, a custom XML file named Custom_File_YFS_getOrderDetails_input.xml refers to the YFS_getOrderDetails_input.xml file in the xmlstruct directory.

However, the naming convention is optional. For example, you can also name your custom XML sampleCustomApi.xml but no base file is used.

In this case, the tool outputs an informational message to indicate that no base XML is found.

> **Note:** If you want to use our base XML file for conversion, the naming convention of your custom XML must be suffixed appropriately. For example, `Custom_File_YFS_getOrderDetails_input.xml` would use the base file named `YFS_getOrderDetails_input.xml`.

The generated XSD specifies the target namespace as shown below:

```
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.yantra.com/documentation"
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:yfc="http://www.yantra.com/documentation">
```

This namespace is picked up from the `xmlns` attribute on the root element of the input XML and defaults to `http://www.yantra.com/documentation`

The XSD and DTD files contain query type attributes used in list APIs when `QryTypeSupported="Y"` is set in the root element of the input XML. Similarly, the complex query types defined for `getItemList()` and `getOrganizationList()` APIs are represented in the XSD and DTD files when `ComplexQuerySupported="Y"` is set.

However, in APIs the following exceptions are exhibited in the DTDs since these constraints cannot be represented in a pure DTD, XSD or both:

- If an XML contains multiple `Extn` attributes, the generated DTD-only (not generated XSD) defines a single `Extn` element which appears as the union of all possible `Extn` elements.
- Conditionally required attributes. For example, you need to specify a group of attributes or another group of attributes such as `OrderHeaderKey` or `EnterpriseCode/OrderNo`.
- Mandatory condition of a node depends on some attribute value. For example in the `createOrder()` API, the `OrderLine` node is required if the `DraftOrderFlag="N"`.

6.2 Defining Complex Queries

Complex queries help to narrow a detailed listing obtained as output from an API. To generate the desired output, you can pass queries using And or Or operators in the input XML of an API.

For example, you can query the getItemList API based on the unit of measure, item group code or any parameters provided in the API definition, using the complex query operators, And or Or.

Complex queries are supported for the following APIs:

- deletePricelistAssignmentList
- deletePricingRuleAssignmentList
- getClassificationPurposeList
- getCustomerContactList
- getInventoryReservationList
- getItemList
- getOrderLineList
- getOrderList
- getOrganizationList
- getSearchIndexTriggerList
- getShipmentList

> **Note:** Only item, organization, order, order line, shipment and shipment line entities are supported for performing complex queries. The attributes for complex query must map directly to valid database columns of these entities and should be within the same XML element.
>
> For more information about these APIs, see the *Selling and Fulfillment Foundation: Javadocs*. For more information on valid database columns, see the Selling and Fulfillment Foundation ERDs.

Consider the following scenario for adding complex queries to the getItemList API.

The getItemList API returns a list of items based on the selection criteria specified in the input XML such as item attributes, aliases, category, and so on. You can create complex queries in the getItemList input XML as shown in Example 6–2.

Example 6–2 Adding Complex Queries in getItemList API

```
<Item OrganizationCode="DEFAULT" ItemGroupCode="PS" >
   <PrimaryInformation PricingQuantityStrategy="IQTY">
   <ComplexQuery Operator="OR">
      <And>
         <Or>
         <Exp Name="UnitOfMeasure" QryType="ISNULL"/>
         <Exp Name="UnitOfMeasure" Value="HR"
         QryType="FLIKE"/>
         </Or>
         <And>
         <Exp Name="ManufacturerName" Value="STERLING"/>
         </And>
      </And>
   </ComplexQuery>
   </PrimaryInformation>
</Item>
```

The OrganizationCode and ItemGroupCode are the two attributes of the <Item> element and PricingQuantityStrategy is the attribute of the <PrimaryInformation> element considered in this example. However you can include any or all of the attributes in the getItemList API. All the attributes in the API are interpreted with an implied And along with the complex query operator.

Apply the following rules when including complex queries:

- You can define only one ComplexQuery under a single element. For example, you cannot have two ComplexQuery operator under an Item element.
- You cannot add a single complex query against two different tables. For example, in getShipmentList API you cannot use ChainedFromOrderHeaderKey and ShipmentLineNo in the same query, since the former belongs to YFS_ORDER_LINE table and the latter is an attribute of the YFS_SHIPMENT_LINE table.

- The attribute with no value is not considered in the complex query, like `Attribute=""`.
- For attributes appended with `QryType`, specify a query type value from Table 6–3. This is case sensitive.

The values for the `QryType` attributes vary depending on the datatype of the field. Table 6–3 lists the supported query type values for each datatype.

Table 6–3 Query Type Values Used by List APIs

Field DataType	Supported Query Type Values
Char/VarChar2	• EQ - Equal to • FLIKE - Starts with • LIKE - Contains • GT - Greater than • LT - Less than
Number	• BETWEEN - Range of values • EQ - Equal to • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to
Date	• DATERANGE - Range of dates • EQ - Equals • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to

Table 6–3 Query Type Values Used by List APIs

Field DataType	Supported Query Type Values
Date-Time	• BETWEEN - Range of dates • EQ - Equals • GE - Greater than or equal to • GT - Greater than • LE - Less than or equal to • LT - Less than • NE - Not equal to
Null	• ISNULL - Return records that are null. • NOTNULL - Return records that are not null. **Note:** These two query types are used when the column or attribute is set to Nullable in the entity XML.

- There can be only one element under the `ComplexQuery` namely, `And` or `Or`.
- `And` or `Or` elements can have one or many child elements as required.
- `And` or `Or` elements can have other `And` or `Or` expression elements as child elements.

This example can be interpreted as the following logical expression:

```
(OrganizationCode="DEFAULT" AND ItemGroupCode="PS" ) AND
((PricingQuantityStrategy="IQTY") OR ( ( UnitOfMeasure =
"EACH" OR UnitOfMeasure="HR" ) AND ( ManufacturerName =
"STERLING") ))
```

Thus by following the above example you can include complex queries to achieve desired results from your database using the above mentioned APIs.

7

Creating Extended APIs

7.1 Invoking Extended APIs

Extended APIs are APIs that you provide; they are sometimes called custom APIs. You can use an extended API to invoke a Selling and Fulfillment Foundation API or third-party API, as well as to perform custom processing through the Service Definition Framework.

To invoke an extended API:

1. Code a class.
2. Code a function that has exactly two parameters of types YFSEnvironment and Document and ensure that the function returns a document.

   ```
   public Document <method-name> (YFSEnvironment env, Document doc)
   ```

3. Configure a service that contains an API node. When configuring an API node, use the properties described in Table 7–1.

Table 7–1 API Node Configuration Properties

Property	Description
General Tab	
Extended API	Select this option if a custom API is to be invoked.
API Name	Select or enter the API to be called. **Note:** This field is for integration purposes only.
Class Name	Specifies the class you coded in Step 1.
Method Name	Specifies the function to be called as coded in Step 2.

Table 7–1 API Node Configuration Properties

Property	Description
Arguments Tab	
Argument Name	You can pass name/value pairs to the API by entering the values in the Arguments Tab. In order for custom APIs to access custom values, the API should implement the interface `com.yantra.interop.japi.YIFCustomApi`. If entered, these name/value pairs are passed to the custom API as a properties object.
Argument Value	Enter the argument value.
Template Tab	
XML Template	Select this radio button to construct the XML to be used for the API output. Enter the template root element name and click OK. You can then construct the XML.
File Name	Select this radio button to enter the filename of the XML file to be used as the API output template. This file should also exist in your CLASSPATH.
Facts Tab	
You can configure the Fact Lookup for Database and custom APIs by using the Facts tab. You can define Name-Value pairs for Fact lookup. The Value can be an XML Path.	
Fact Name	Enter the fact name.
Fact Value	Enter the fact value.

When connecting the nodes within a service, keep in mind the API node connection properties as listed in Table 7–2:

Table 7–2 API Node Connection Properties

Connection	Node Connection Rules
Can be the first node after the start node	Only for services invoked synchronously
Can be placed before	• Any transport node except FTP or File IO • Any other component node

Table 7–2 API Node Connection Properties

Connection	Node Connection Rules
Can be placed after	• Start node • Any transport node except FTP or File IO • Any other component node
Passes data unchanged	Yes

4. Make sure the class is in the CLASSPATH of the Service Definition Framework.
5. Make sure that the class implements a method with a signature that takes in exactly two parameters, a YFSEnvironment and a Document.

   The following example shows how to implement a class:

```
import com.yantra.yfs.japi.YFSEnvironment;
import org.w3c.dom.Document;
public class Bar {
   public Bar () {
      }
   public Document foo(YFSEnvironment env, Document doc)
      {
      //write your implementation code here
      }
}
```

6. To access the extended API you created, invoke the service containing your extended API.

   For details and sample code that show how to access properties specified when the custom API is configured, see the `YIFCustomAPI` interface in the *Selling and Fulfillment Foundation: Javadocs*.

7.2 Implementing the Error Sequence User Exit

You can configure the Service Definition Framework to call a user exit that checks for prior errors for the exception group to which the API belongs. This user exit is called before any processing of the message starts. A Java interface is supplied for its implementation. This interface definition is in the `com.yantra.interop.japi.YIFErrorSequenceUE class`. The user exit computes the Message Key based on user defined custom code.

YIFErrorSequenceUE defines two functions. The function definitions are:

```
1) public Document getExceptionGroupReference(Document document, String apiName)
throws Exception
2) public void setExceptionGroupFinder (YIFExceptionGroupFinder finder)
```

The getExceptionGroupReference() function takes two parameters:

- Document - The input XML document retrieved by the Integration Adapter
- String - The API for which the Integration Adapter retrieved the XML

The setExceptionGroupFinder() function sets the YIFExceptionGroupFinder() interface. Use the implementation of this interface to retrieve the exceptionGroupId if prior errors exist.

An example implementation of this function is:

```
public void setExceptionGroupFinder (YIFExceptionGroupFinder finder){
 this.finder = finder;
}
```

7.3 Implementing the YIFExceptionGroupFinder Interface

This interface defines the findExistingError() function that takes in Document as the input parameter.

For example, the input XML document that the user exit passes to the findExistingError() function would contain:

```
<?xml version="1.0"?>
<ExceptionGroupReference messageKey="xyz"/>
```

7.4 Exception Handling in Extended APIs

The client always has the option of throwing an exception to the Service Definition Framework instead of handling it when it occurs. Depending on the configuration, the Service Definition Framework either sends the exception to the Alert Console or logs the exception.

7.5 Locking Records in Extended APIs

You can lock a record in a custom table when executing a custom entity API in Service Definition Framework (SDF). To lock a record, you must pass the `SelectMethod` attribute as part of the input XML to the custom entity API. The locking happens within the transaction boundary of the custom API call.

The value of the `SelectMethod` attribute will determine what (if any) type of locking to be used. You can pass one of the following values for the `SelectMethod` attribute:

- WAIT—The record is locked for SELECT FOR UPDATE operation.
- NO_WAIT—The record is locked for SELECT FOR UPDATE NOWAIT operation.
- NONE—Locking mechanism is not used.

Note: If you pass any other value for the `SelectMethod` attribute, an error is thrown indicating that the "SelectMethod" attribute value is not valid.

Note: If the `SelectMethod` attribute does not exist or if it is set to NONE in the input XML, the locking mechanism is not used.

8

Invoking APIs and Services

8.1 Invoking APIs from the Client Environment

In order to call standard APIs from the client, ensure that the client environment is set up correctly. The client environment must have appropriate CLASSPATH settings and the JAR files as described in this section.

> **Note:** It is recommended that you do not invoke a local API in the application JVM before the server initialization. Also, if you are making a local API call, you must add the following code after the local API invocation in the block that has YIFClientFactoryImpl.getLocalApi and api.invoke:
>
> `YFCRemoteManager.setIsLocalInvocation(false);`

The `<INSTALL_DIR>/resources/` directory must contain the `yifclient.properties` file.

If you are calling in local mode, the client CLASSPATH must contain all JAR files referred to in the `<INSTALL_DIR>/properties/dynamicclasspath.cfg` Selling and Fulfillment Foundation file

When invoking APIs through EJB or HTTP in IBM WebSphere, the client CLASSPATH must contain the following files in `<WAS_HOME>/AppClient/properties` directory:

- `xapi.jar`
- `log4j-1.2.15.jar`
- `platform_afc.jar`

- xercesImpl.jar
- xml-apis.jar
- j2ee.jar
- com.ibm.ws.ejb.thinclient_<version>.jar
- com.ibm.ws.orb_7.0.0.jar
- com.ibm.ws.sib.client.thin.jms_<version>.jar
- ejbstubs.jar

The client CLASSPATH must also contain the following files from <INSTALL_DIR>/jar/:

- install_foundation.jar
- smcfs/8.5/smcfsshared.jar

Use *ejb.jar available in the EAR file to retrieve the ejbstubs.jar.

When invoking APIs through EJB in BEA Weblogic, consult your Application server documentation for CLASSPATH requirements:

- wlclient.jar
- wljmsclient.jar

8.2 Invoking Services and Standard APIs Programmatically

Selling and Fulfillment Foundation provides sample code that demonstrates how the Selling and Fulfillment Foundation standard APIs and services can be invoked programmatically. See the sample files in the <INSTALL_DIR>/xapidocs/code_examples/ directory.

> **Note:** Use the executeFlow() method of the YIFApi interface to run a service defined within the Service Definition Framework.

API and service transactions that are outbound from Selling and Fulfillment Foundation can be configured through the Service Builder, as described in the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

API and service transactions that are inbound to Selling and Fulfillment Foundation can be invoked through the following protocols:

- EJB
- HTTP and HTTPS
- LOCAL
- Web Services
- COM+

8.2.1 EJB

Use EJB for server-side execution of the code. Java call. All the methods in Selling and Fulfillment Foundation take a `YFSEnvironment` and a `document`, and return a `document`. Since EJBs are designed to be called remotely, each of these documents is serialized on one end and unserialized on the other. However, Selling and Fulfillment Foundation uses an EJB, where each API takes two string parameters and returns a string. Thereby, forcing any document implementation to serialize and unserialize using a standard well-defined interface.

For example, a new EJB is created with method signatures like:

```
String createOrder(String env, String inputXML) throws
YFSException, RemoteException;
```

where env is an XML that should be a valid input to `createEnvironment` variable. The return value is the output XML.

When calling an API using `YIFClientFactory.getInstance().getApi("EJB")` the call is made using this String-based EJB. With this type of call you can pass a `YFSEnvironment` and `document`, and get a `document` in return. The Selling and Fulfillment Foundation code performs the conversion transparently.

Note: The DOM-based EJB is deprecated. Hence, moving forward you need to use the String-based EJB for server-side execution.

8.2.2 HTTP

Use HTTP for server side execution of code. Java call.

8.2.3 LOCAL

Use Local for client side execution of code. COM **or** Java call.

8.2.4 Web Services

Use Web Services for client side execution of code. COM **or** Java call.

8.2.5 COM+

Use COM for client side execution of VB or C++ code. COM **or** Java call.

Using COM requires setting up your server and runtime clients.

Note: Exceptions encountered when making synchronous API calls through EJB, COM, or HTTP transport protocols are not queued for reprocessing.

8.2.6 Configuring Service Invocation

To configure service invocation:

1. Rename the `<INSTALL_DIR>/resources/yifclient.properties.in` file to `<INSTALL_DIR>/resources/yifclient.properties`.
2. Ensure that the CLASSPATH contains the following:
   - `log4j-1.2.15.jar`
   - `xercesImpl.jar`
   - `install_foundation.jar`
   - `platform_afc.jar`

- resources.jar
- entities.jar
- xapi.jar
- JARs requried by your Application server
- JARs required by user exits and custom APIs

3. Set your java command line property to `-Dlog4j.configuration=resources/log4jconfig.xml`.
4. Make sure that the <INSTALL_DIR> directory is in your CLASSPATH.
5. Set the log4j properties in the `log4jconfig.xml` file to the appropriate values for your environment. If these properties are not specified correctly, the Service Definition Framework does not initialize correctly.
   - If you are using the EJB protocol and BEA WebLogic, make sure that `weblogic.jar` is in your CLASSPATH environment variable. In addition, `xercesImpl.jar` and `xalan.jar` must precede `weblogic.jar` in your CLASSPATH.
   - If you are using the EJB protocol and *JBoss*, make sure that <JBOSS_HOME>/client/jbossall-client.jar is in your CLASSPATH environment variable.
   - If you are using the EJB protocol and IBM WebSphere, make sure that the CLASSPATH environment variable contains the necessary JAR files. For information about the WebSphere JAR files, see IBM documentation. Make sure that the CLASSPATH environment variable contains the appropriate `properties` directory.

Note: If you are invoking the service or API from the machine on which the server is running, make sure that the CLASSPATH environment variable contains the <WAS_HOME>/AppServer/properties/ directory.

If you are invoking the service or API from a different machine, make sure that the CLASSPATH environment variable contains the <WAS_HOME>/AppClient/properties/ directory.

- If you are configuring a COM+ protocol call, use one of the following COM signatures that you need:

```
createEnvironment(VARIANT *lEnvHandle, BSTR sProgID, BSTR sUserID, int *iRetval)
```

Signature for calling standard APIs:

```
<SterlingAPI>(VARIANT *lEnvHandle, BSTR inXML, VARIANT *outXML, VARIANT *errXML, int *retval)
```

Signature for calling services:

```
executeFlow(VARIANT *lEnvHandle, BSTR flowName, BSTR flowMsg, VARIANT *outXML, VARIANT *errXML, int *retval)
```

For examples of VB code, see the samples in the <INSTALL_DIR>/xapidocs/code_examples/complus directory.

8.3 Directing API Calls to Specific Servers

The Application provides the ability to route custom API calls to a particular server or servers when these APIs are invoked either remotely or locally.

To enable this, the server and the protocol must be specified in the *yifclient.properties file* and in the required APIs.

To direct API calls to a specific server, perform the following steps:

1. Specify the attribute, "endpoint" in the *yifclient.properties* file under the directory /<INSTALL_DIR>/resources/. The "endpoint" is a configured server or a protocol which is used for routing the API calls. Modify the *yifclient.properties* file to include the declaration and usage of endpoint in the following format:

```
endpoint.<Server_Name>.apifactory.protocol=HTTP
endpoint.<Server_Name>.httpapi.url=http://<server>:<port>/<context_
root>/interop/InteropHttpServlet
```

endpoint.<Server_Name> specifies a server with the name <Server_Name>. For example, endpoint.INBOXSERVER creates an endpoint with the name, "INBOXSERVER". You can assign properties to the endpoint name, which takes precedence over any other assigned property.

The protocol configured for the endpoint is HTTP as specified in the line endpoint.<Server_Name>.apifactory.protocol=HTTP

The settings specified in the second line, endpoint.<Server_Name>.httpapi.url are used to connect to the server specified in the endpoint.

2. Configure a protocol to be used for connecting to the specified server. The protocols such as HTTP, HTTPS, EJB, LOCAL, AUTO are reserved endpoint names. If any of these are configured for the endpoint, the system uses the default connection settings (as applicable) for routing the API calls.
3. For each API, specify the endpoint (server) to be used as shown below:

```
yfs.api.<apiname>.endpoint=<ENDPOINT>
```

The API then calls the server specified in the endpoint attribute.

Note: If an endpoint is not configured for an API, then it uses the default (local) server or a general one configured for all APIs. The property set at the API level takes precedence over other common properties.

Index

F

G

I

L

N

O

P

Q

S

U

X

Y

Z