ORACLE
INSURANCE

Oracle® Documaker

Introduction to Enterprise Web Processing Services

version 1.2

Part number: E14902-01

September 2009

THIRD PARTY SOFTWARE NOTICES

This product includes software developed by Apache Software Foundation (http://www.apache.org/).

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

This product includes software distributed via the Berkeley Software Distribution (BSD) and licensed for binary distribution under the Generic BSD license.

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.

This product includes software developed by the JDOM Project (http://www.jdom.org/).

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

This product includes software developed by the Massachusetts Institute of Technology (MIT).

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This product includes software developed by Jean-loup Gailly and Mark Adler. This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

This software is based in part on the work of the Independent JPEG Group (http://www.ijg.org/).

This product includes software developed by the Dojo Foundation (http://dojotoolkit.org).

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.

Copyright (c) 2005-2009, The Dojo Foundation. All rights reserved.

This product includes software developed by W3C.

Copyright © 2009 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. (http://www.w3.org/Consortium/Legal/)

This product includes software developed by Mathew R. Miller (http://www.bluecreststudios.com).

Copyright (c) 1999-2002 ComputerSmarts. All rights reserved.

This product includes software developed by Shaun Wilde and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Chris Maunder and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by PJ Arends and distributed via Code Project Open License (http://www.codeproject.com).

THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

This product includes software developed by Erwin Tratar. This source code and all accompanying material is copyright (c) 1998-1999 Erwin Tratar. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY. USE IT AT YOUR OWN RISK! THE AUTHOR ACCEPTS NO LIABILITY FOR ANY DAMAGE/LOSS OF BUSINESS THAT THIS PRODUCT MAY CAUSE.

This product includes software developed by Sam Leffler of Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE

Copyright (c) 1988-1997 Sam Leffler
Copyright (c) 1991-1997 Silicon Graphics, Inc.

This product includes software developed by Guy Eric Schalnat, Andreas Dilger, Glenn Randers-Pehrson (current maintainer), and others. (http://www.libpng.org)

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

This product includes software components distributed by the Cryptix Foundation.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED 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 CRYPTIX FOUNDATION LIMITED 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

Copyright © 1995-2005 The Cryptix Foundation Limited. All rights reserved.

This product includes software components distributed by Sun Microsystems.

This software is provided "AS IS," without a warranty of any kind. ALLEXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANYIMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BELIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. 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, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Copyright (c) 1998 Sun Microsystems, Inc. All Rights Reserved.

This product includes software components distributed by Dennis M. Sosnoski.

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.

Copyright © 2003-2007 Dennis M. Sosnoski. All Rights Reserved

It also includes materials licensed under Apache 1.1 and the following XPP3 license

THIS SOFTWARE IS PROVIDED "AS IS"' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE 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.

Copyright © 2002 Extreme! Lab, Indiana University. All Rights Reserved

---

This product includes software components distributed by CodeProject. This software contains material that is © 1994-2005 The Ultimate Toolbox, all rights reserved.

This product includes software components distributed by Geir Landro.

Copyright © 2001-2003 Geir Landro (drop@destroydrop.com) JavaScript Tree - www.destroydrop.com/hjavascripts/tree/version 0.96

Contents

Chapter 4, Available Web Services

Chapter 5, Additional Resources

Chapter 1

Overview

The need to produce customer information 24 hours a day, seven days a week has shifted a large percentage of publishing volume away from traditional batch processing to a real-time, customer-driven, business model.

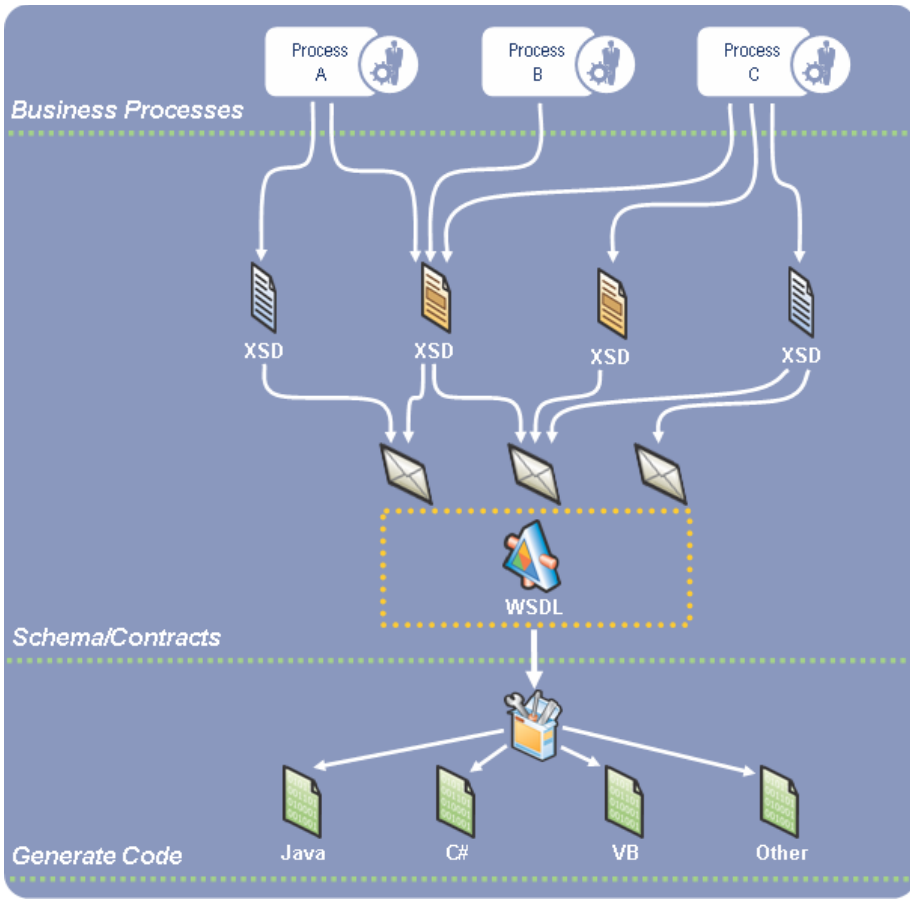
Moreover, companies increasingly want to leverage the web to reach their customers and prospects, resulting in new requirements for scalability and reliability.

At Oracle Insurance, we recognize that organizations are changing how they do business, and we have come to the marketplace with technology and architecture in keeping with this significant market shift.

This chapter discusses the following topics:

The Need for Web Services

The Enterprise Web Processing Services (EWPS) framework offers functionality via a set of established and interoperable standards such as XML and web services. This allows a multitude of enterprise applications — including policy production and claims correspondence — to be designed and developed around a core functional infrastructure.



Oracle Insurance's contract-first approach to design

So, what exactly is *WSDL*? WSDL stands for *Web Services Description Language*. WSDL is kind of an XML grammar for describing web services interfaces (available functions).

WSDL leverages XML schema to describe the basic types used by a web service and provides all sorts of additional information that frame the *contract* of the interface, including things like ports, bindings, and so on.

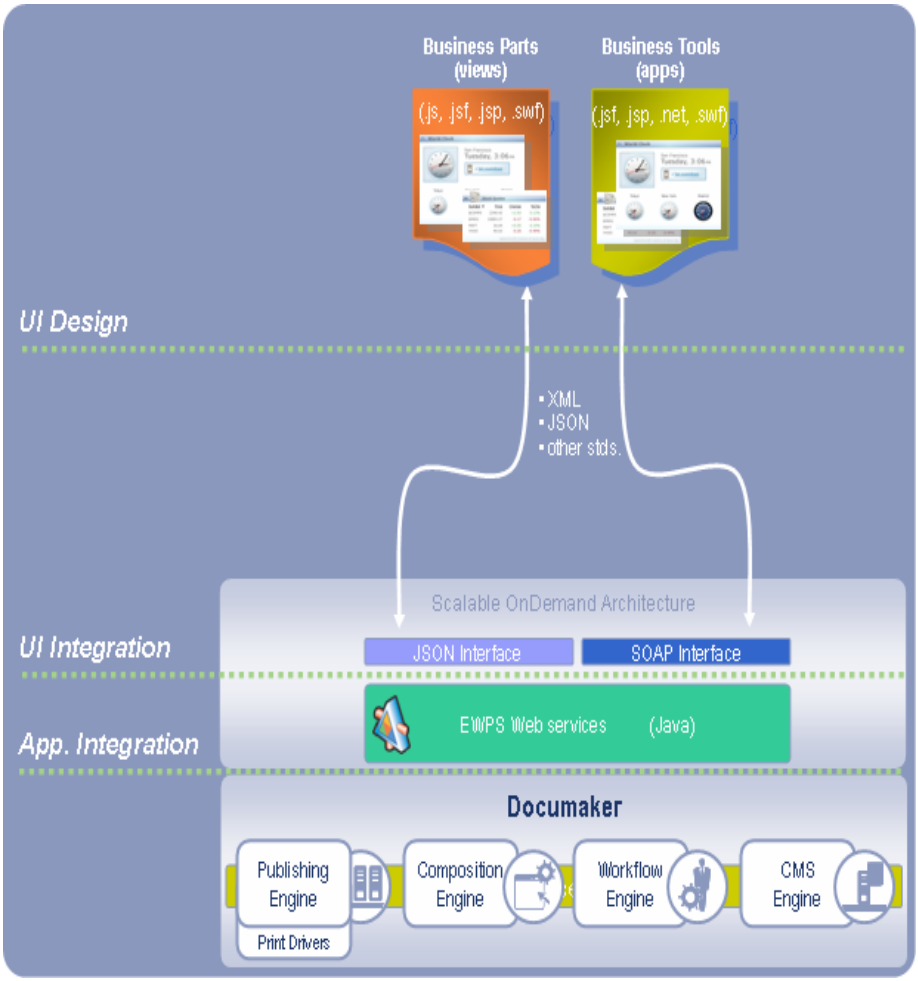
NOTE: For more information about WSDL, see Web Services Description Language on page 75.

Available Services and Methods of Use

EWPS provides access to the Oracle Insurance suite of publishing, composition, workflow, and content management engines. It enables third-party applications to build custom applications, tools, and services that leverage the full breadth of Oracle Insurance functionality.

EWPS emphasizes business value throughout the whole web services technology stack. This self-service model means you can use a multitude of essential mechanisms — WS-I SOAP interfaces for application integration, JSON for UI integration, or pre-packaged *business parts* for design-time integration — in any sort solution.

EWPS is available in an Apache Axis2 package for J2EE and designed to be hosted under most popular application servers such as Apache Tomcat (Java), IBM WebSphere (Java), and BEA WebLogic (Java).



Strategic opportunities for integration with EWPS

Typical EWPS-enabled applications include:

- Self-service publishing solutions
- Document search utilities
- Composition and workflow systems
- Systems that embed publishing artifacts in their web pages
- Applications that assist users in creating various types of documents and forms

An EWPS-enabled application can present data in ways that best meet the needs of a particular business scenario.

EWPS supports these protocols:

- SOAP (Simple Object Access Protocol). See Using SOAP on page 5 for more information.
- JSON (JavaScript Object Notation) See Using JSON on page 7 for more information.

USING SOAP

With the SOAP API, the request interface (called a proxy) contains business-object interfaces and stubs generated directly from a WSDL document that specifies the EWPS schema and service address.

The third-party application works with data in the form of object properties. It sends and receives the data by calling object methods. The auto-generated SOAP proxy handles the details of serializing/de-serializing the SOAP request from EWPS into objects that are easy to work with.

NOTE: The SOAP API is built on open standards like SOAP and WSDL. These standards are supported by a wide-range of development tools on a variety of platforms. For more information, see SOAP on page 72.

Request:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <doGetTemplateListRequest xmlns="http://
webservices.docucorp.com/ewps/schema/2005-12-01">
      <AuthUser>string</AuthUser>
      <LibraryId>string</LibraryId>
      <BusUnitsList>
        <Key1 id="string" package="string">
          <Key2 id="string" />
          <Key2 id="string" />
        </Key1>
        <Key1 id="string" package="string">
          <Key2 id="string" />
          <Key2 id="string" />
        </Key1>
      </BusUnitsList>
      <EffectiveDate>string</EffectiveDate>
      <Start>integer</Start>
      <MaxResults>integer</MaxResults>
      <NameQuery>string</NameQuery>
      <DescQuery>string</DescQuery>
      <ContentQuery>string</ContentQuery>
      <SortBy>string</SortBy>
      <ResponseGroup>
        <Response>string</Response>
        <Response>string</Response>
      </ResponseGroup>
    </doGetTemplateListRequest>
  </soap:Body>
</soap:Envelope>
```

Response:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <doGetTemplateListResponse xmlns="http://
webservices.docucorp.com/ewps/schema/2005-12-01">
         <Result>Success</Result>
<TemplateList>
   <Story StoryName="string" id="string">
      <Key1 id="string" package="string">
    <Key2 id="string"/>
      </Key1>
      <Description>string</Description>
          <Props>
                   <Prop name="string">string</Prop>
      </Props>
   </Story>
   <Story StoryName="string" id="string">
      <Key1 id="string" package="string">
    <Key2 id="string"/>
      </Key1>
      <Required>boolean</Required>
      <Description>string</Description>
      <Props>
         <Prop name="string">string</Prop>
      </Props>
   </Story>
</TemplateList>
<RecipientList>
   <Recipient name="string">
      <Copies>integer</Copies>
      <Story StoryName="string" id="string"/>
   </Recipient>
   <Recipient name="string">
      <Copies>integer</Copies>
      <Story StoryName="string" id="string"/>
     </Recipient>
</RecipientList>
<StartIndex>integer</StartIndex>
<EndIndex>integer</EndIndex>
<TotalResults>integer</TotalResults>
<SearchTime>string</SearchTime>
      </doGetTemplateListResponse>
   </soap:Body>
</soap:Envelope>
```

Sample SOAP request and response template

USING JSON

The JSON API works just like the SOAP API, except requests and responses are handled in JSON rather than XML. JSON is a lightweight data-interchange format based upon a subset of the JavaScript language.

NOTE: For an overview of JSON, including the various tools and techniques for working with JSON, go to this web site: http://www.json.org.

```
Request: (using JavaScript – actual request is an HTTP POST)
var request = {
    "LibraryId":"string",
    "Start":integer,
    "MaxResults":integer,
    "BusUnitsList":
    [{
        "Key2":
        [{
            "id":"string"
        }],
        "id":"string",
        "package":"string"
    }]
     };

http://server/EWPS/DocumentService/<request.toJSONString()>

Response:
{
"TemplateList":
[{
    "Key1":
    {
        "Key2":
        [{
            "id":"string"
        }],
        "id":"string",
        "package":"string"
    },
    "Required":boolean,"
    "Description":"string",
    "Props":
    [{
        "name":"string",
        "Value":"string"
    }],
    "StoryName":"string",
    "id":"string",
    "alias":"string"
},
{
    "Key1":
```

```
    {
        "Key2":
        [{
            "id":"string"
        }],
        "id":"string",
        "package":"string"
    },
    "Required":boolean,
    "Description":"string",
    "Props":
    [{
        "name":"string",
        "Value":"string"
    }],
    "StoryName":"string",
    "id":"string",
    "alias":string
},
"RecipientList":
[{
    "Copies":"string",
    "Story":
    [{
        "extracopies":integer,
        "StoryName":"string",
        "id":"string",
        "alias":"string"
    },
    {
        "extracopies":integer,
        "StoryName":"string",
        "id":"string",
        "alias":"string"
    }],
    "name":"string"
}],
"StartIndex":integer,
"EndIndex":integer,
"TotalResults":integer,
"SearchTime":"0.031",
"Result":integer
}
```

Sample JSON request and response.template

Choosing Between SOAP and JSON

Both the SOAP API and JSON API share the same underlying schema, so the basic format of the input and output data is the same regardless of the API being used — both access the same EWPS functionality and data. As a result, you can use one or the other or both. It just depends on whichever approach works best for your situation.

Here is a complete listing of available services and a general overview of what they are used for:

Service	An operation used to...	Expanded terminology
doGetLibraries	Get a list of the possible form libraries available for collaborative authoring, composition, or publishing services.	Form Library – Config, MRL
doGetBusUnits	Get a list of candidate business-unit selection criteria for a particular library. This helps refine the document-selection process.	
doGetTemplatelist	Get a list of candidate forms available for collaborative authoring, composition, or publishing services.	
doGetTemplateListData	Get the schema for a given template list, including details pertaining to the Story, StoryFragments and Field information for the template list selection.	For Documaker publishing, think of a *story* as a form and a *story fragment* as a section
doCreateFolder (ComposeData)	Create a remote folder for composition, collaborative authoring, or publishing from a list of story templates and field data (ComposeData).	For Documaker publishing, think of a *folder* as a WIP item, Future use for Documanage Archive Folders.
doCreateFolder (Import)	Create a remote folder for composition, collaborative authoring, or publishing from an extract file (Import) used with a set of pre-defined rules.	
doGetFolderList	Get a list of Folders for a specific owner, search criteria, or both.	Think of *owners* as a specific user
doGetFolder	Retrieve the contents of a folder.	
doModifyFolder	Add, remove, re-arrange, and generally modify the templates/data (ComposeData) and/or CoreProperties of a folder.	
doDeleteFolder	Delete a folder from the application.	

Service	An operation used to...	Expanded terminology
doPublish (FolderId)	Publish a document from a pre-existing folder (FolderId). See DistributionOptions on page 42 for information pertaining to the various publishing and distribution options.	
doPublish (ComposeData)	Publish a document from a list of story templates and field data (ComposeData). See DistributionOptions on page 42 for information pertaining to the various publishing and distribution options.	
doPublish (Import)	Publish a document from an extract file (Import) with a set of pre-defined rules. See DistributionOptions on page 42 for information pertaining to the various publishing and distribution options.	

EWPS Requirements

The EWPS Java web application deployment supports Java Runtime Environment (JRE) version 1.5 or higher. For best results, use JRE version 1.6. This table shows you the version of Java you need for each supported platform:

For this platform	You need this version of Java
Microsoft Windows (x86-32 and x86-64)	
XP (SP 2) Server 2003 (SP2)	Sun Java 5 or higher
Linux (x86-32 and x86-64) 32- and 64-bit kernels	
SuSE Linux Enterprise Server (SLES) version 9.4 or higher RedHat Enterprise Linux (RHEL) version 5.1 or higher	Sun Java 5 or higher or IBM Java 5 or higher
Sun Solaris (SPARC) 32-and 64-bit kernels	
Solaris 9/SunOS 5.9 or higher)	Sun Java 5 or higher
IBM AIX 5L pSeries (RISC) 32- and 64-bit kernels	
version 5.2 TL 5200-09	IBM Java 5 or higher
version 5.3 TL 5300-07	IBM Java 6 or higher

Web application server

Whether running on Windows, Linux, Solaris, or AIX, you can use either of the following Java Web Application Servers:

- IBM WebSphere AS, version 6.1 or higher
- Tomcat version 5.5 or higher

NOTE: We test with both WebSphere and Tomcat. Other Java application servers should also work.

An EWPS Java web application deployment requires the following version of Docupresentment to process web services requests:

- Docupresentment version 2.2, patch 04 or higher

Message bus systems

EWPS communicates to Docupresentment using a message bus provider such as JMS, WebSphere MQ, or MSMQ. This table shows the possible message bus systems. Tested systems are indicated with a single asterisk (*).

For this platform	You can use one of these message busses
Microsoft Windows (x86-32 and x86-64) • XP (SP 2 • Server 2003 (SP2)	HTTP/SOAP(*) (**) MSMQ (*) IBM WebSphere MQ, version 5.3 or higher (*) ActiveMQ JMS (*) (***) Other JMS providers
Linux (x86-32 and x86-64) **** • SuSE Linux Enterprise Server (SLES) version 9.4 or higher • RedHat Enterprise Linux (RHEL) version 5.1 or higher	HTTP/SOAP(*) (**) IBM WebSphere MQ, version 5.3 or higher (*) ActiveMQ JMS (*) (***) Other JMS providers
Sun Solaris (SPARC) • Solaris 9/SunOS 5.9 or higher	HTTP/SOAP(*) (**) IBM WebSphere MQ, version 5.3 or higher (*) ActiveMQ JMS (*) (***) Other JMS providers
IBM AIX 5L pSeries (RISC) 32-bit and 64-bit • version 5.2 TL 5200-09 • version 5.3 TL 5300-07	HTTP/SOAP(*) (**) IBM WebSphere MQ, version 5.3 or higher (*) ActiveMQ JMS (*) (***) Other JMS providers

(*) Tested providers
(**) This provider has not been tested under heavy concurrency and load.
(***) At the time of publication, we noticed problems with the ActiveMQ JMS during heavy load testing, We expect Apache to address these issues in the future. Contact Apache for more information.
(****) Other Linux distributions should work at kernel v2.6.11.4-21 or higher but have not been tested.

Docupresentment, JRE, and Documaker versions

This table shows, for each supported and tested platform, the version of Docupresentment, the corresponding Java Runtime Environment (JRE) for Docupresentment, and the version of Documaker necessary to support EWPS.

For this platform	Docupresentment	JRE	Documaker
Microsoft Windows (x86-32 and x86-64) • XP (SP 2) • Vista (SP 1) • Server 2003 (SP2)	Version 2.2, patch 04* or higher	Sun JRE v1.5** Sun JRE v1.6	Version 11.3, patch 05 or higher
Linux (x86-32 and x86-64)**** • SuSE Linux Enterprise Server (SLES) version 9.4 or higher • RedHat Enterprise Linux (RHEL) version 5.1 or higher	Version 2.2, patch 04* or higher	Sun JRE v1.5** Sun JRE v1.6 IBM JRE v1.5** IBM JRE v1.6***	Version 11.3, patch 05 or higher
Sun Solaris (SPARC) • Solaris 9/SunOS 5.9 or higher	Version 2.2, patch 04* or higher	Sun JRE v1.5** Sun JRE v1.6	Version 11.3, patch 05 or higher
IBM AIX 5L pSeries (RISC) 32-bit and 64-bit • version 5.2 TL 5200-09 • version 5.3 TL 5300-07	Version 2.2, patch 04* or higher	IBM JRE v1.5** IBM JRE v1.6***	Version 11.3, patch 05 or higher

* A Docupresentment Windows installation includes a bundled Sun J2SE JRE version 1.6 which is used by default by the Docupresentment server.
** Running the Docupresentment web services interface under JRE version 1.5 requires JAXB 2. See Downloading JAXB on page 14 for more information.
*** IBM's J2SE JRE version 1.6 is only supported in Docupresentment version 2.2, patch 05 or higher and Documaker Shared Objects version 11.3, patch 06 or higher.
**** Other Linux distributions should work at kernel version 2.6.11.4-21 or higher but have not been tested.

NOTE: To find the latest version of Documaker, Docupresentment, iDocumaker, iPPS, or EWPS, log into the following web site:

http://aru.us.oracle.com/

Downloading JAXB

JAXB provides a way to map XML and Java code and extend applications with XML and Web Services technologies. You must install and implement JAXB 2 to run the Docupresentment web services interface under JRE version 1.5.

Follow these steps to download the latest version of JAXB:

1. Go to the JAXB web site:

   https://jaxb.dev.java.net/

2. Click Download Now for the latest version of JAXB.
3. On the new page, click the Download the Binary link and save the ZIP file to your local machine.
4. Unzip the downloaded ZIP file into directories. There will be a \jaxb-ri directory with a \lib subdirectory.
5. From the \lib subdirectory, copy the jaxb-api.jar and jaxb-impl.jar files to the \lib directory of your Docupresentment installation.

Chapter 2

Business Scenarios

As mentioned earlier, Enterprise Web Processing Services (EWPS) provides set of well-defined services that have been designed from the standpoint of functionality and business-use.

Here are some typical business scenarios and how you can use Enterprise Web Processing Services to address them:

Publishing a Quote Form from a Rating Application

Suppose a carrier wants to use Oracle Insurance to produce a real-time quote form from their rating application. Some carriers may host an online rating application as an added benefit to their agents. Since an agent is providing sufficient data to get a quote, this data can be used to populate and publish the quote form.

How do you map the rating application data to the actual quote form? Should the customer be responsible for resolving all mapping prior to the web service call or should you use Oracle Insurance tools for mapping?

Using EWPS, you can accomplish this task using the doPublish web service two ways:

- Use Oracle Insurance tools to do the mapping — The carrier produces an Oracle Insurance standard XML or data extract file to be used as an import file.

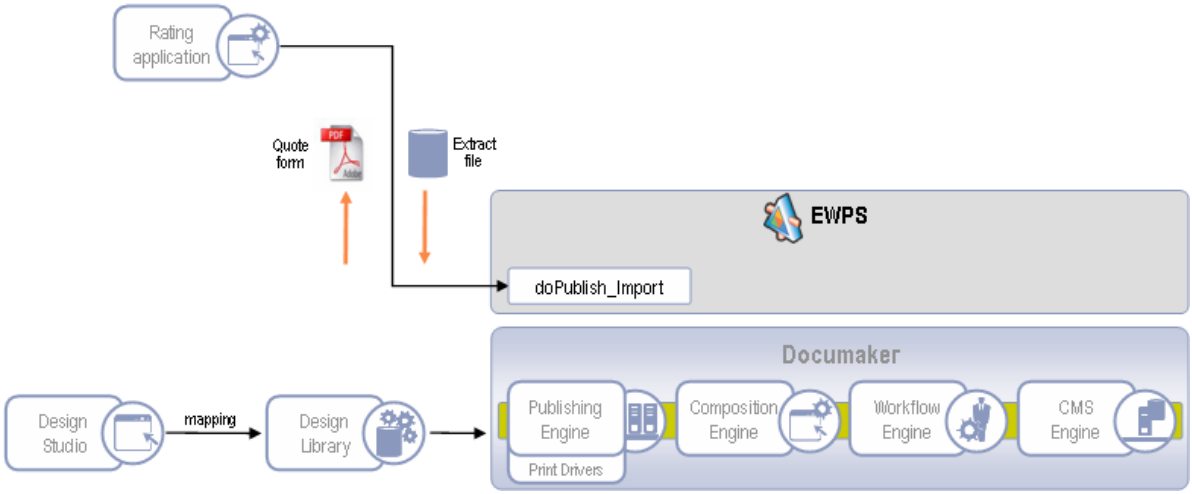
  See Option 1: Mapping the Data Using Oracle Insurance Tools on page 17 for more information.

- Mapping resolved prior to the doPublish request — The carrier can get schema for the quote form via the doGetTemplateListData web service and then provide a ComposeData structure populated with data.

  See Option 2: Resolving the Data Mapping Before the doPublish Request on page 19 for more information.

OPTION 1: MAPPING THE DATA USING ORACLE INSURANCE TOOLS

- Rating data is mapped to the quote form using Skywire Software tools.
- Rating engine generates a standardized extract with data.
- Rating engine calls EWPS doPublish Web service with extract.
- Carrier disburses the quote form as necessary.



Process flow diagram

Step 1 **MAPPING.** The rating data is mapped to the quote form using Oracle Insurance tools. A layout or copy-book of the extract feed is provided as a reference point.

Step 2 **EXTRACT.** The rating engine generates a standardized extract file with data for the quote form.

Step 3 **DOPUBLISH.** The rating engine calls the EWPS doPublish web service with the extract data. For example, here is a sample request:

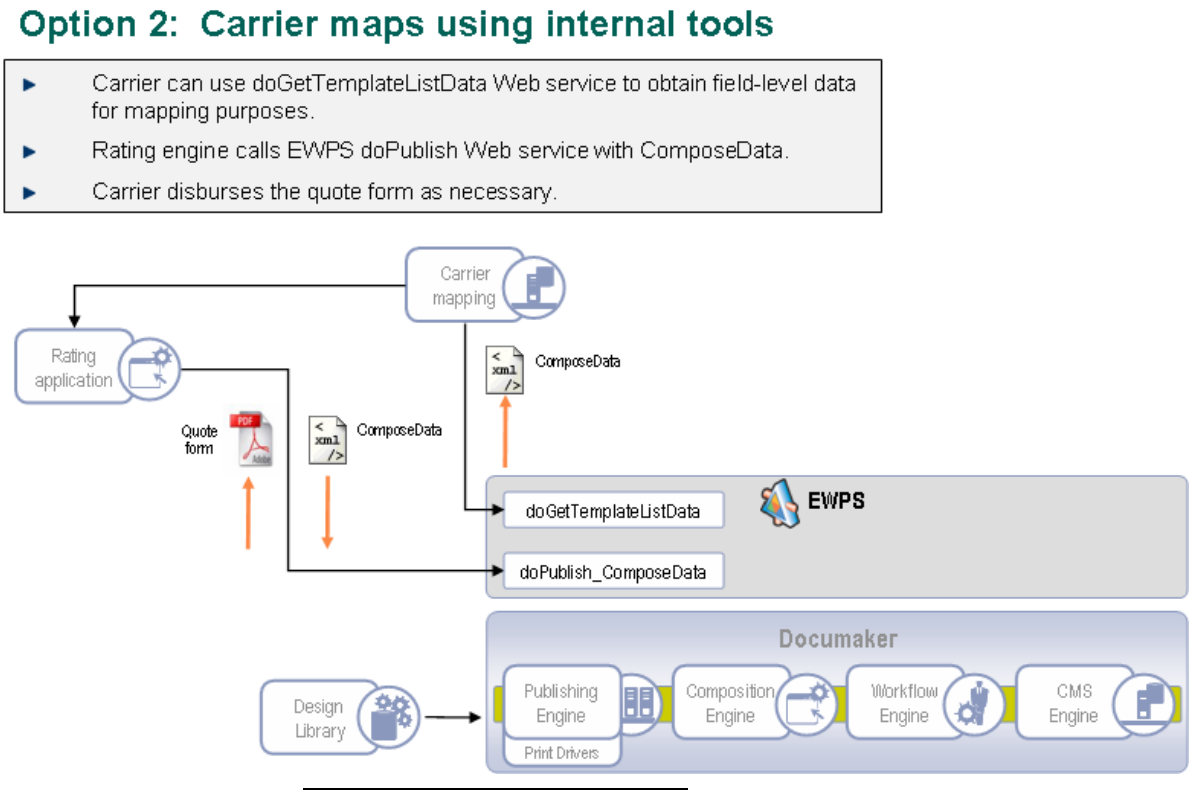
```
<doPublishRequest xsi:type="doPublishReq_Import"
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
       <LibraryId>100AIC</LibraryId>
       <DistributionOptions xsi:type="DistributionOptions_ADHOC"
source="ADHOC">
              <Priority>REALTIME</Priority>
              <Channel xsi:type="Channel_IMMEDIATE">
                     <PublishType>PDF</PublishType>
                     <DistributionType>IMMEDIATE</DistributionType>
                     <Disposition location="ATTACH"/>
                     <Recipient name="INSURED">
                            <Props/>
                            <Copies>1</Copies>
                     </Recipient>
              </Channel>
       </DistributionOptions>
       <SourceType>IMPORT</SourceType>
       <Import>
              <ImportFile xsi:type="ImportFile_ATTACH"
d5p1:contentType="" location=""
xmlns:d5p1="http://www.w3.org/2005/05/xmlmime">PD94bWwgdmVyc2lvbj0iM
S4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxETONVTUVOVCBUWVBFPSJSUFdJUCIgVkVSU
0lPTj0iMTEuMyI+DQo8RE9DUOVUIE5BTUU9IiI+DQo8RklFTEQgTkFNRTOiUE9MSUNZI
j5BSUM5MDkwQTwvRklFTEQ+DQo8RklFTEQgTkFNRTOiSU5TTkFNIj5CaWxsIFNhbXBsZ
TwvRklFTEQ+DQo8RklFTEQgTkFNRTOiSU5TQUQxIj4xMjMgTWFpbiBTdHJlZXQ8L0ZJR
UxEPg0KPEZJRUxEIE5BTUU9Ik10UONUWSI+QXRsYW50YTwvRklFTEQ+DQo8RklFTEQgT
kFNRTOiSU5TU1QiPkdBPC9GSUVMRD4NCjxGSUVMRCBOQU1FPSJJTlNaSVAiPj5UPgOK<
/ImportFile>
       </Import>
</doPublishRequest>
```

…and the response:

```
<doPublishResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <DistributionResults source="ADHOC">
    <Channel>
      <PublishType>PDF</PublishType>
      <DistributionType>IMMEDIATE</DistributionType>
      <Documents>
        <Document xsi:type="DocumentFile_ATTACH"
d7p1:contentType="application/pdf" status="Success"
location="ATTACH"
xmlns:d7p1="http://www.w3.org/2005/05/xmlmime">JVBERiOxLjMNJeLjz9MNC
jEOMiAwIG9iagO8PC9MaW5lYXJpemVkIDEuMDAvTCAxMjY4MDcgICAgL0ggWyAlMzEgI
CAgICAgIDMyMyAgICAgICBdL08gMTQOICAgICAgIC9FIDU4MTEgICAgICAvTiAzMSAgI
CAgICAgLlQgMTIzOTE5ICAgID4+DWVuZG9VPRgOK</Document>
      </Documents>
    </Channel>
  </DistributionResults>
</doPublishResponse>
```

Step 4 **DISTRIBUTION.** The carrier distributes the quote form as necessary.

OPTION 2: RESOLVING THE DATA MAPPING BEFORE THE DOPUBLISH REQUEST



Process flow diagram

Step 1 **MAPPING.** The carrier uses the doGetTemplateListData web service to get field-level information for mapping purposes. For example, here is a sample request:

```
<doGetTemplateListDataRequest
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <AuthUser />
  <LibraryId>100AIC</LibraryId>
  <TemplateList>
    <Story StoryName="Quote Form" id="35">
      <Key1 id="AMERGEN">
        <Key2 id="QUOTE" />
      </Key1>
      <Description>Quote form</Description>
      <Props>
        <Prop name="OPTIONS">N</Prop>
      </Props>
    </Story>
  </TemplateList>
  <EffectiveDate />
  <ResponseGroup>
    <Response />
  </ResponseGroup>
</doGetTemplateListDataRequest>
```

…and the response:

```
<doGetTemplateListDataResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <ComposeData>
    <Field name="QUOTATION" />
    <Field name="POLICY" />
    <Field name="TO:" />
    <Field name="FROM:" />
    <Field name="BIND" />
    <Field name="INSNAM" />
    <Field name="INSAD1" />
    <Field name="INSAD2" />
    <Field name="INSCTY" />
    <Field name="INSST" />
    <Field name="INSZIP" />
    <Field name="COMPANY LINE 1" />
    <Field name="EFFDTE" />
    <Field name="EXPDTE" />
    <!-- More fields ommitted for brevity-->
    <Story StoryName="Quote Form" id="35">
      <Key1 id="AMERGEN">
        <Key2 id="QUOTE" />
      </Key1>
      <Description>Quote form</Description>
      <Props>
        <Prop name="OPTIONS">N</Prop>
      </Props>
      <StoryFragments>
        <StoryFragment FragmentName="quote">
          <Field name="TODAYS DATE" />
          <Field name="VOICE #" />
          <Field name="FAX #" />
          <Field name="COVERAGE" />
          <Field name="NOTES" />
          <Field name="NOTES #002" />
        </StoryFragment>
      </StoryFragments>
    </Story>
  </ComposeData>
</doGetTemplateListDataResponse>
```

Step 2 **DOPUBLISH.** The rating engine calls the EWPS doPublish web service with ComposeData. For example, here is a sample request:

```
<doPublishRequest xsi:type="doPublishReq_ComposeData"
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <LibraryId>100AIC</LibraryId>
  <DistributionOptions xsi:type="DistributionOptions_ADHOC"
source="ADHOC">
    <Priority>REALTIME</Priority>
    <Channel xsi:type="Channel_IMMEDIATE">
      <PublishType>PDF</PublishType>
      <DistributionType>IMMEDIATE</DistributionType>
      <Disposition location="ATTACH"/>
      <Recipient name="INSURED">
        <Props/>
        <Copies>1</Copies>
      </Recipient>
    </Channel>
  </DistributionOptions>
  <SourceType>COMPOSEDATA</SourceType>
  <ComposeData>
    <Field name="QUOTATION">sample</Field>
    <Field name="POLICY">213422</Field>
    <Field name="TO:">A.D. Kent</Field>
    <Field name="FROM:">John Doe </Field>
    <Field name="BIND" />
    <Field name="INSNAM">Insured Name</Field>
    <Field name="INSAD1">Addr 1</Field>
    <Field name="INSAD2">Addr 2</Field>
    <Field name="INSAD3">Addr 3</Field>
    <Field name="INSCTY">City</Field>
    <Field name="INSST">State</Field>
    <Field name="INSZIP">00000</Field>
    <Field name="EFFDTE">07/04/2007</Field>
    <Field name="EXPDTE">07/03/2008</Field>
    <!-- More fields ommitted for brevity-->
    <Story StoryName="Quote Form" id="35">
      <Key1 id="AMERGEN">
        <Key2 id="QUOTE" />
      </Key1>
      <Description>Quote form</Description>
      <Props>
        <Prop name="OPTIONS">N</Prop>
      </Props>
      <StoryFragments>
        <StoryFragment FragmentName="quote">
          <Field name="TODAYS DATE">06/14/2007</Field>
          <Field name="VOICE #">770-555-5555</Field>
          <Field name="COVERAGE" />
          <Field name="NOTES">some notes here</Field>
        </StoryFragment>
      </StoryFragments>
    </Story>
  </ComposeData>
</doPublishRequest>
```

...and the response:

```
<doPublishResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <DistributionResults source="ADHOC">
    <Channel>
      <PublishType>PDF</PublishType>
      <DistributionType>IMMEDIATE</DistributionType>
      <Documents>
        <Document xsi:type="DocumentFile_ATTACH"
d7pl:contentType="application/pdf" status="Success"
location="ATTACH"
xmlns:d7pl="http://www.w3.org/2005/05/xmlmime">JVBERi0xLjMNJeLjz9MNC
jEOMiAwIG9iag08PC9MaW5lYXJpemVkIDEuMDAvTCAxMjY4MDcgICAgL0ggWyAlMzEgI
CAgICAgIDMyMyAgICAgICBdL08gMTQ0ICAgICAgIC9FIDU4MTEgICAgICAvTiAzMSAgI
CAgICAgLlQgMTIzOTE5ICAgID4+DWVuZG9VPRg0K</Document>
      </Documents>
    </Channel>
  </DistributionResults>
</doPublishResponse>
```

Step 3 **DISTRIBUTION.** The carrier distributes the quote form as necessary.

Initiating an Issuance Process from a Rating Application

Once a quote as been bound, an underwriter initiates the issuance process. Typically, the assigned underwriter produces the quote and will know when the time is right to issue the policy, which means some human intervention is required for issuance.

So how do you know what forms are needed to issue the policy and how do you know how to map the data? Can the carrier can manage the job of mapping/triggering the policy forms based on the type of quote and the data (using doGetTemplateListData) or do you use Oracle Insurance tools for mapping and form triggering?

You can accomplish this business scenario using multiple steps — use the EWPS doCreateFolder web service to create the transaction and use another application, such as a policy administration system or iDocumaker/iPPS to complete the issuance process.

Using EWPS, you can accomplish the task of creating the transaction using the doCreateFolder web service one of two ways:

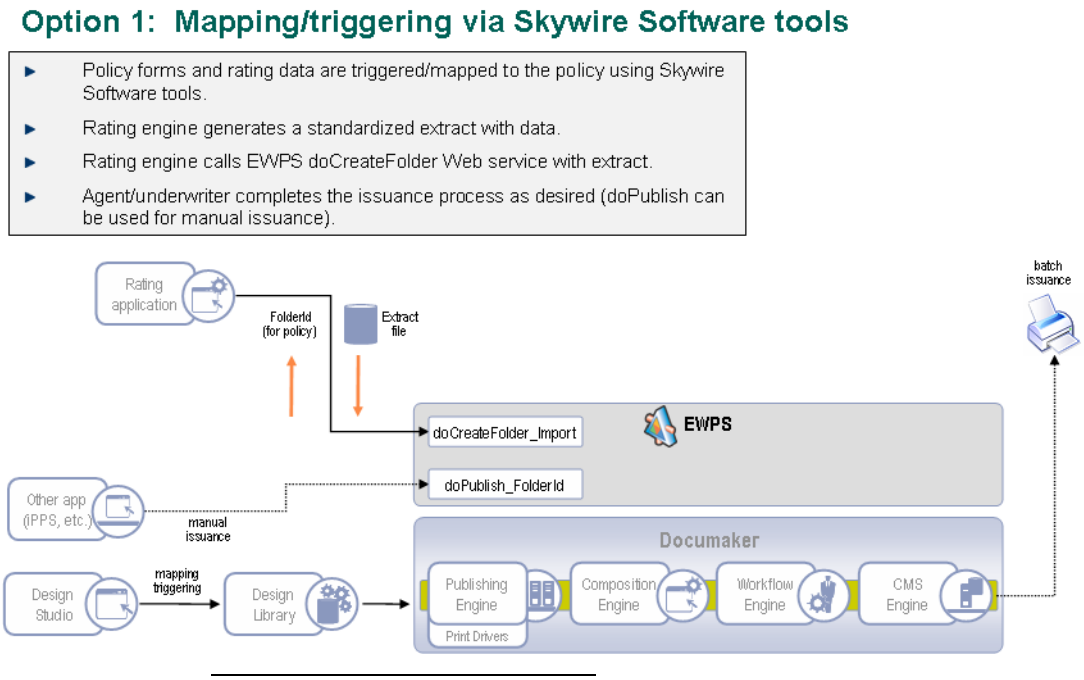
- Using Oracle Insurance tools to map the data — The carrier produces an Oracle Insurance standard XML or data extract file to be used as an import file.

  See Option 1: Mapping the Data using Oracle Insurance Tools on page 25 for more information.

- Mapping resolved prior to doCreateFolder request — The carrier can get schema for the quote form via the doGetTemplateListData web service and then provide a ComposeData structure populated with data.

  See Option 2: Resolving the Data Mapping Before the doPublish Request on page 27 for more information.

OPTION 1: MAPPING THE DATA USING ORACLE INSURANCE TOOLS



Process flow diagram

Step 1 **MAPPING.** The rating data is mapped to the policy forms using Oracle Insurance tools. A layout or copy-book of the extract feed is provided as a reference point for mapping and triggering.

Step 2 **EXTRACT.** The rating engine generates a standardized extract file with data for the quote form.

Step 3 **DOPUBLISH.** The rating engine calls the EWPS doCreateFolder web service with the extract data. For example, here is a sample request:

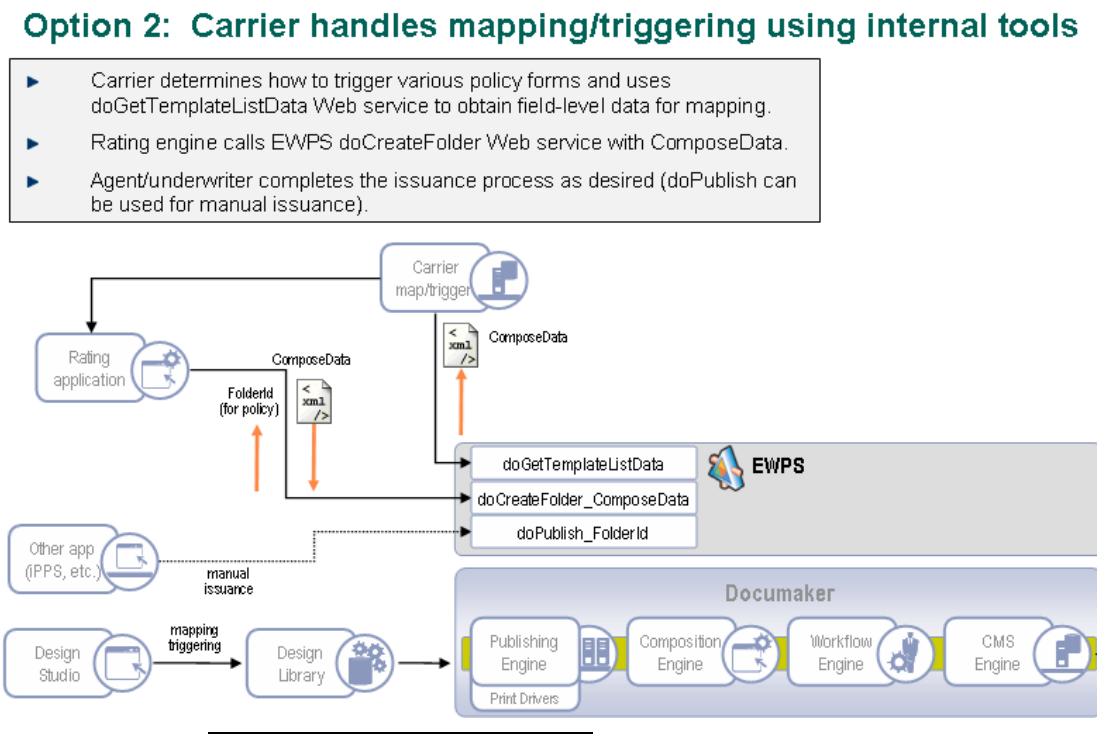
```
<doCreateFolderRequest xsi:type="doCreateFolder_Import"
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Owner>DEM01</Owner>
  <CoreProperties>
    <LibraryId>100AIC</LibraryId>
    <DocumentId>AIC9090M</DocumentId>
    <Key1 id="AMERGEN">
      <Key2 id="INTERLINE" />
    </Key1>
    <Description>Policy From WebService</Description>
    <Props>
      <Prop name="LOCID">100</Prop>
      <Prop name="SUBLOCID">0</Prop>
      <Prop name="JURISDICTN">AL</Prop>
      <Prop name="TRNNAME">Bill Sample</Prop>
    </Props>
  </CoreProperties>
              <ImportFile xsi:type="ImportFile_ATTACH"
d5p1:contentType="" location=""
xmlns:d5p1="http://www.w3.org/2005/05/xmlmime">PD94bWwgdmVyc2lvbj0iM
S4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxETONVTUVOVCBUWVBFPSJSUFdJUCIgVkVSU
0lPTj0iMTEuMyI+DQo8RE9DUOVUIE5BTUU9IiI+DQo8RklFTEQgTkFNRT0iUE9MSUNZI
j5BSUM5MDkwQTwvRklFTEQ+DQo8RklFTEQgTkFNRT0iSU5TTkFNIj5CaWxsIFNhbXBsZ
TwvRklFTEQ+DQo8RklFTEQgTkFNRT0iSU5TQUQxIj4xMjMgTWFpbiBTdHJlZXQ8L0ZJR
UxEPgOKPEZJRUxEIE5BTUU9Ikl0UONUWSI+QXRsYW50YTwvRklFTEQ+DQo8RklFTEQgT
kFNRT0iSU5TU1QiPkdBPC9GSUVMRD4NCjxGSUVMRCBOQUlFPSJJTlNaSVAiPj5UPgOK<
/ImportFile>
</doCreateFolderRequest>
```

…and the response:

```
<doCreateFolderResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <FolderId>72</FolderId>
</doCreateFolderResponse>
```

Step 4 **ISSUANCE.** The agent or underwriter completes the issuance as desired via their policy administration system or by using an application such as iPPS.

OPTION 2: RESOLVING THE DATA MAPPING BEFORE THE DOPUBLISH REQUEST



Process flow diagram

Step 1 **MAPPING.** The carrier uses the doGetTemplateListData web service to get field-level information for mapping and triggering purposes. For example, here is a sample request:

```
<doGetTemplateListDataRequest
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01"
schemaVersion="1.0">
  <AuthUser></AuthUser>
  <LibraryId>100AIC</LibraryId>
  <TemplateList>
    <Story StoryName="A100  03-1997" id="2870026170">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE" />
      </Key1>
      <Description>Common Policy Dec - DC</Description>
      <Props>
        <Prop name="OPTIONS">RMX</Prop>
      </Props>
    </Story>
    <Story StoryName="A101  03-1997" id="4">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE" />
      </Key1>
      <Description>Minimum Earned Premium Endt</Description>
      <Props>
        <Prop name="OPTIONS">RM</Prop>
      </Props>
    </Story>
  </TemplateList>
</doGetTemplateListDataRequest>
```

…and the response:

```
<doGetTemplateListDataResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <ComposeData>
    <Field name="POLICY"/>
    <Story StoryName="A100  03-1997" id="2870026170">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE"/>
      </Key1>
      <Description>Common Policy Dec - DC</Description>
      <Props>
        <Prop name="OPTIONS">RMX</Prop>
      </Props>
      <StoryFragments/>
    </Story>
    <Story StoryName="A101  03-1997" id="4">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE"/>
      </Key1>
      <Description>Minimum Earned Premium Endt</Description>
      <Props>
        <Prop name="OPTIONS">RM</Prop>
      </Props>
      <StoryFragments>
        <StoryFragment FragmentName="a101c">
          <Field name="ERNPRM"/>
          <Field name="PERCENT"/>
        </StoryFragment>
      </StoryFragments>
    </Story>
  </ComposeData>
</doGetTemplateListDataResponse>
```

Step 2 **DOCREATEFOLDER.** The rating engine calls the EWPS doCreateFolder web service with ComposeData. For example, here is a sample request:

```
<doCreateFolderRequest xsi:type="doCreateFolderReq_ComposeData"
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Owner>DEM01</Owner>
  <CoreProperties>
    <LibraryId>100AIC</LibraryId>
    <DocumentId>AIC9091M</DocumentId>
    <Key1 id="AMERGEN">
      <Key2 id="INTERLINE" />
    </Key1>
    <Description>Policy From WebService</Description>
    <Props>
      <Prop name="LOCID">100</Prop>
      <Prop name="SUBLOCID">0</Prop>
      <Prop name="JURISDICTN">AL</Prop>
      <Prop name="TRNNAME">Bill Sample</Prop>
    </Props>
  </CoreProperties>
  <ComposeData>
    <Field name="POLICY">AIC9091M</Field>
    <Story StoryName="A100  03-1997" id="2870026170">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE"/>
      </Key1>
      <Description>Common Policy Dec - DC</Description>
      <Props>
        <Prop name="OPTIONS">RMX</Prop>
      </Props>
      <StoryFragments/>
    </Story>
    <Story StoryName="A101  03-1997" id="4">
      <Key1 id="AMERGEN" package="INTERLINE">
        <Key2 id="INTERLINE"/>
      </Key1>
      <Description>Minimum Earned Premium Endt</Description>
      <Props>
        <Prop name="OPTIONS">RM</Prop>
      </Props>
      <StoryFragments>
        <StoryFragment FragmentName="a101c">
          <Field name="ERNPRM">$100,000</Field>
          <Field name="PERCENT">35%</Field>
        </StoryFragment>
      </StoryFragments>
    </Story>
  </ComposeData>
</doCreateFolderRequest>
```

…and the response:

```
<doCreateFolderResponse
xmlns="http://webservices.docucorp.com/ewps/schema/2005-12-01">
  <Result>Success</Result>
  <FolderId>73</FolderId>
</doCreateFolderResponse>
```

Step 3 **ISSUANCE.** The agent or underwriter completes the issuance as desired via their policy administration system or by using an application such as iPPS.

Chapter 3

Common Schema Types

Here are the common schema types used throughout Oracle Insurance's Enterprise Web Processing Solution (EWPS).

These schema may be part of a message contract in one or more instances:

LIBRARYLIST

The LibraryList group provides information about libraries and their respective publishing services.

Here is a sample group:

```
<LibraryList>
     <Library id="Amergen">
          <Service type="Entry" name="Entry"/>
          <Service type="WIP" name="Work in Process"/>
          <Service type="Archive" name="Archive"/>
     </Library>
     <Library id="DOCC">
      <Service type="Entry" name="Entry"/>
    </Library>
</LibraryList>
```

BUSUNITSLIST

The BusUnitsList group provides information about business units that you can use to refine and filter the document selection process. Additional business unit refinement appears as nested lists of *Key* criteria (Key1, Key2, Key3, and so on).

Here is a sample group:

```
<BusUnitsList>
     <Key1 id="AMERGEN PACKAGE" package="AMERGEN PACKAGE">
          <Key2 id="CRIME"/>
          <Key2 id="INLAND MARINE"/>
          <Key2 id="LIABILITY"/>
          <Key2 id="PROPERTY"/>
          <Key2 id="MOTOR TRUCK CARGO"/>
     </Key1>
     <Key1 id="AMERGEN GL" package="GENERAL LIABILITY">
          <Key2 id="LIABILITY"/>
     </Key1>
     <Key1 id="AMERGEN PROPERTY" package="COMM'L PROPERTY">
          <Key2 id="PROPERTY"/>
     </Key1>
     <Key1 id="AMERGEN IM">
          <Key2 id="INLAND MARINE"/>
     </Key1>
     <Key1 id="AMERGEN MTC">
          <Key2 id="MOTOR TRUCK CARGO"/>
     </Key1>
     <Key1 id="AMERGEN AUTO">
          <Key2 id="AUTO"/>
     </Key1>
     <Key1 id="AMERGEN IM">
          <Key2 id="INLAND MARINE">
               <Key3 id="PWC"/>
               <Key3 id="OTHER"/>
          </Key2>
     </Key1>
</BusUnitsList>
```

TEMPLATELIST

The TemplateList group provides information about candidate templates (story) returned from a query or when filtering requests for a library.

Here is a sample group:

```
<TemplateList>
     <Story StoryName="Letter" id="1" alias="">
          <Required>Yes</Required>
          <Description>Customer Letter</Description>
     </Story>
     <Story StoryName="Bill Letter" id="2" alias="">
          <Required>No</Required>
          <Description>Bill Letter</Description>
     </Story>
     <Story StoryName="Bill Letter" id="2.1" alias="">
          <Required>No</Required>
          <Description>Bill Letter Duplicate</Description>
     </Story>
</TemplateList>
```

NOTE: For a Documaker implementation, a *story* can be limited in scope and be considered to be roughly equivalent to a *form*. As part of a broader schema for future growth and functionality, a story can extend across multiple pages, and several stories can share a single page.

A story can encompass the entire contents of a document package, or it may include an individual block of content. Additionally, a story could be quite dynamic; appearing in blocks throughout a document. such as the first part on page 1, the second part on page 5, and so on.

RECIPIENTLIST

The RecipientList group provides a way to associate recipients with story templates. The RecipientList is exclusive to composition services such as doGetTemplateList and the folder-oriented services.

Here is a sample group:

```
<RecipientList>
     <Recipient name="AGENT">
         <Story StoryName="Letter" id="1" alias="" extracopies="1"/>
          <Story StoryName="Bill Letter" id="2" alias=""
extracopies="0"/>
     </Recipient>
     <Recipient name="HOME OFFICE">
         <Story StoryName="Letter" id="1" alias="" extracopies="0"/>
          <Story StoryName="Bill Letter" id="2" alias=""
extracopies="0"/>
     </Recipient>
     <Recipient name="INSURED">
         <Story StoryName="Letter" id="1" alias="" extracopies="0"/>
          <Story StoryName="Bill Letter" id="2" alias=""
extracopies="0"/>
     </Recipient>
</RecipientList>
```

COMPOSEDATA

The ComposeData group (requests only) provides information about the user-entered data on a particular page of a document composition to be saved for stateful requests.

NOTE: Schema for FIELD attributes are primarily driven by the type of View that is returned, mostly via the attributes found at the field (such as INPUT) level.

Here is a sample group:

```
<ComposeData>
     <Field name="GlobalField" type="" required="True">data</Field>
     <Story StoryName="Letter" id="1" alias="">
          <Field name="StoryFieldField" type="">data</Field>
          <StoryFragments>
               <StoryFragment FragmentName="CPADR">
                    <Field name="StoryFragmentField1" type="">Bob</
Field>
                   <Field name=" StoryFragmentField2" type="">Main</
Field>
                    <Remark datestamp="06/06/2005, 11:44">Review</
Remark>
               </StoryFragment>
               <StoryFragment FragmentName="CPBODY">
                    <Field name="StoryFragmentField3"
required="True">text here</Field>
               </StoryFragment>
          </StoryFragments>
     </Story>
</ComposeData>
```

NOTE: For a Documaker implementation, a *StoryFragment* can be considered to be similar to a section or image.

PROPS

The Props group provides a generic structure for extended properties that are not native to base schema objects.

For example, a recipient might have extended property information for distribution addresses or a folder could have extended property information with its CoreProperties to handle application-specific attributes.

The following schema objects can have extended properties:

- Story
- Recipient
- Folder (CoreProperties)

Here is a sample group:

```
<Props>
  <Prop name="propertyname1">propertyvalue1</Prop>
  <Prop name="propertyname2">propertyvalue2</Prop>
  <Prop name="propertyname3">propertyvalue3</Prop>
  <Prop name="propertyname4">propertyvalue4</Prop>
  ...
</Props>
```

COREPROPERTIES

The CoreProperties group provides information about the core properties of a document or folder.

Here is a sample group:

```
<CoreProperties>
     <Library id="DOCUCORP"/>
     <Description>Past Due Notification</Description>
     <DocumentId>90125</DocumentId>
     <DocumentType>NB</DocumentType>
     <StatusCode>N</StatusCode>
     <EffectiveDate>2005-12-01</EffectiveDate>
     <Key1 id="AMERGEN PACKAGE" package="AMERGEN PACKAGE">
          <Key2 id="CRIME"/>
          <Key2 id="INLAND MARINE"/>
          <Key2 id="LIABILITY"/>
          <Key2 id="PROPERTY"/>
          <Key2 id="MOTOR TRUCK CARGO"/>
     </Key1>
     <Props>
          <Prop name="RECNUM">66421AER7</Prop>
     </Props>
</CoreProperties>
```

IMPORTFILETYPE

The ImportFileType provides a generic structure for passing a chunk (file) of opaque data to a service as a base64Binary element. Note that data can be referenced as in-line data (location="ATTACH") or via a URL (location="URL").

Here is a sample group:

```
<Import>
     <ImportFile location="ATTACH" p5:contentType="text/xml"
xmlns:p5="http://www.w3.org/2005/05/
xmlmime">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLT…
     </ImportFile>
     <ImportFile location="URL" p5:contentType="text/xml"
xmlns:p5="http://www.w3.org/2005/05/xmlmime"> file:////1.1.1.1/
38ED0A22842449A49D921B7542D09EC0.XML
     </ImportFile>
</Import>
```

Oracle Insurance supports different contentType definitions, which you can use to provide EWPS with information about the type of file being sent. Here are the supported contentTypes and their meaning:

contentTypes	Tells the system to treat the referenced file as
application/vnd.docucorp+xml	Oracle Insurance XML format
application/vnd.docucorp+v2	Oracle Insurance V2 format (PPS import)
application/vnd.docucorp+extract	A raw extract file.

ERRORS

The Errors group provides information about any errors or problems that occurred during a request.

Here is a sample group:

```
<Errors>
    <Error>
        <ErrorCode>String</ErrorCode>
        <DetailedMessage>String</DetailedMessage>
        <ErrorSource>String</ErrorSource>
        <Severity>Warning</Severity>
        <Remedy>String</Remedy>
        <Trace>String</Trace>
    </Error>
    <Error>
        <ErrorCode>String</ErrorCode>
        <DetailedMessage>String</DetailedMessage>
        <ErrorSource>String</ErrorSource>
        <Severity>Warning</Severity>
        <Remedy>String</Remedy>
        <Trace>String</Trace>
    </Error>
</Errors>
```

RESPONSEGROUP

The ResponseGroup group, only used for web service requests, provides a way to specify one or more optional response groups as part of an overall web service response.

NOTE: This group is reserved for future use.

Here is a sample group:

```
<ResponseGroup>
      <Response>Group1</Response>
      <Response>Group2</Response>
      <Response>Group3</Response>
</ResponseGroup>
```

DISTRIBUTIONOPTIONS

The DistributionOptions group provides information about publishing channels and the recipient-specific distribution options contained therein. You can handle each recipient differently with specific options (sample A) or bundled together as a group (sample B).

Here is a discussion of the parameters you can use:

Parameter	Description
Copies	(Optional) Determines the number of copies desired for each recipient type. The default is one copy.
DocucorpArchive	(Optional) Determines if the transaction should be archived into the Docucorp Smart Archive.
Distribution Source	(Optional) Determines the source of the information driving the distribution. You can choose from Ad Hoc or Predefined. The default is Ad Hoc.
Priority	(Optional) Determines the publishing priority of the document. DEFERRED means that a true batch system handles all distribution and publish type specifications. Basically, the system saves the transaction to WIP with a *Batch* status code for the nightly process to pick it up and print it. For DEFERRED, a generic DistributionResults is returned stating the status was *Sent*. REALTIME means the publishing system will be executed immediately and will tell you what happened during the print process. If REALTIME is used, a detailed listing of the output is returned via the DistributionResults complex type. The default is REALTIME.
PublishType	(Optional) Determines the type of document. • XER (Metacode) • AFP • PCL • PXL (PCL6) • PDF • RTF • HTML • PST (PostScript) • BPD (TIFF or other bitmap) • TXT (line print) • GDI (Windows print) • XMP (XML output) • VIPP (Xerox flavor of PostScript) • V2 (Standard Export File) The default is PDF.

Parameter	Description
DistributionType	(Optional) Determines channel of distribution. The default is Immediate Print. Not valid if the Distribution Source is Predefined. • Immediate Print • Deferred Print (a scheduled or nightly batch)
Disposition	(Optional) Determines how the document should be returned. You can choose from: • URL (for a file location) • ATTACH (for an attachment) See table below for defaults for each distribution type.
Preview	(Optional) Determines if the document should be generated as a *Print* or *Template* preview.
Storys	(Optional) Used to determine which story parts and extra copies are published for each recipient. Not valid with a distribution type of *Batch*. The default is all story parts.
Recipient	(Optional) Describes the recipients requested for print. To let the form set/extract and publishing system handle all default recipients, do not specify any recipients in this structure. When you use a predefined distribution source, you can use a specific recipient to override the disposition generated from the predefined system.

Here is sample group A — a simple example with Predefined source:

```
<DistributionOptions source="PREDEFINED">
   <Priority>REALTIME</Priority>
</DistributionOptions>
```

Here is sample group B — a simple example with Ad Hoc source:

```
<DistributionOptions source="ADHOC">
   <Channel>
    <Recipient name="ALLRECIPS"/>
   </Channel>
</DistributionOptions>
```

Here is sample group C — a simple example for Preview:

```
<DistributionOptions source="ADHOC">
  <Channel>
    <Recipient name="RECIP"/>
    <Preview>True</Preview>
  </Channel>
</DistributionOptions>
```

DISTRIBUTIONRESULTS

The DistributionResults group provides read-only information about the recipient-specific published document results. You can handle each recipient differently with specific options (sample A), bundle them as a group (sample B), or group them in any combination.

This group includes these parameters:

Parameter	Description
Story	This is the list of story templates that contains data (fields and StoryFragments) that comprise the forms.
PublishType	(Optional) Determines type of document. The default is PDF.
Documents	A listing of published documents as base64Binary elements or URL references to external files.
DocumentStatus	This indicates the status of the resulting document: • Failure - The document had some failure and detailed information can be found in the Error object. • Sent - Indicates the Document was sent to the DistributionType of Deferred Print, FAX, or Email. • URL - A URL reference to the document. • ATTACH - Indicates that the published document will be sent back as a 64-bit code encryption. These are the same encryptions found in Docupresentment send/receive message files. You must use a base-64 decoder to view the attachment.

Here is sample group A — a simple example with a pre-defined source:

```
<DistributionResults source="PREDEFINED">
   <Channel>
    <Recipient name="RECIP" id="1">
        <Props>
            <Prop name="RECIP_NAME1">Andy Jones</Prop>
        </Props>
    </Recipient>
    <PublishType>PDF</PublishType>
    <DistributionType>Immediate Print</DistributionType>
    <Documents>
       <Document status="URL">file://
\\myserver\documents\23480283423408098.pdf</Document>
    </Documents>
   </Channel>
   <Channel>
    <Recipient name="RECIP" id="2">
        <Props>
            <Prop name="RECIP_NAME1">Don Rogers</Prop>
        </Props>
    </Recipient>
    <PublishType>PDF</PublishType>
    <DistributionType>Immediate Print</DistributionType>
    <Documents>
```

```
      <Document status="URL">file://
\\myserver\documents\8798uoi79iuyiuyi.pdf</Document>
    </Documents>
   </Channel>
   <Channel>
    <Recipient name="RECIP" id="3">
        <Props>
            <Prop name="RECIP_NAME1">Don Abbot</Prop>
        </Props>
    </Recipient>
    <DistributionType>Deferred Print</DistributionType>
    <Documents>
       <Document status="sent"/>
    </Documents>
   </Channel>
   <Channel>
    <Recipient name="RECIP" id="4">
        <Props>
            <Prop name="RECIP_NAME1">Andy Jones</Prop>
        </Props>
    </Recipient>
    <PublishType>TIFF</PublishType>
    <DistributionType>Immediate Print</DistributionType>
    <Documents>
       <Document status="URL">ftp://client.docucorp.com/ewps/spool/
8205243jlkj345903823.tiff</Document>
    </Documents>
   </Channel>
</DistributionResults>
```

Here is sample group B — a simple example with Ad Hoc source:

```
<DistributionResults source="ADHOC">
      <Channel>
    <Recipient name="ALLRECIPS"/>
    <PublishType>PDF</PublishType>
    <DistributionType>Immediate Print</DistributionType>
    <Documents>
       <Document status="ATTACH" p5:contentType="application/pdf"
xmlns:p5="http://www.w3.org/2005/05/
xmlmime">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLT…</Document>
    </Documents>
      </Channel>
</DistributionResults>
```

Here is sample group C — a simple example for Preview:

```
<DistributionResults ="ADHOC">
     <Channel>
    <Recipient name="RECIP" id="1"/>
    <PublishType>PDF</PublishType>
    <Documents>
       <Document status="ATTACH" p5:contentType="application/pdf"
xmlns:p5="http://www.w3.org/2005/05/
xmlmime">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLT…</Document>
    </Documents>
    </Channel>
</DistributionResults>
```

NOTE: Distribution options must be configured in the Documaker system to properly product the distribution channels provided in the request. See the Documaker System Reference for more information.

Chapter 4

Available Web Services

There are several web services available with EWPS. This chapter describes these services.

You can use the following composition services:

You can use this service for composition or publishing:

You can use this service for publishing:

doGetLibraries

Use this service to get a simple list of candidate document libraries available for publishing services.

This web service is non-stateful in nature and accepts optional user identification for library profiling purposes.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetLibrariesRequest	AuthUser	Optional user identification.	String
	ResponseGroup	Optionally return certain response groups, including: • LibraryList	ResponseGroup
Response			
doGetLibrariesResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	LibraryList	A list of the available libraries.	LibraryList
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doGetBusUnits

Use this service to get a list of candidate business unit (BU) selection criteria for a particular library that helps refine the document selection process.

This web service is non-stateful in nature and accepts optional user identification for library profiling purposes.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetBusUnitsRequest	AuthUser	Optional user identification.	String
	LibraryId	Required library selection (ID).	String
	EffectiveDate	Optional date qualifier.	String
	ResponseGroup	Optionally return certain response groups, including: • BusUnitsList	ResponseGroup
Response			
doGetBusUnitsResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	BusUnitsList	A list of available business-unit groupings to refine transaction selection.	BusUnitsList
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doGetTemplateList

Use this service recursively to get a list of the candidate forms available for publishing services. Use the Start and MaxResults parameters to specify where the template listing should start and how many records are returned. The NameQuery, DescQuery, and ContentQuery parameters filter the results by their form name, description, in-line contents, or any combination thereof.

This web service is non-stateful in nature and accepts optional user identification for library profiling purposes.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetTemplateListRequest	AuthUser	Optional user identification.	String
	LibraryId	Required library selection (ID).	String
	BusUnitsList	A list of selected business-unit groupings to refine transaction selection.	BusUnitsList
	EffectiveDate	Optional date qualifier.	String
	Start	One-based index of the first desired result.	Integer
	MaxResults	Number of results desired per query.	Integer
	NameQuery	Use to refine the search by template name. You can use all or part of this parameter in the query.	String
	DescQuery	Use to refine the search by template description. You can use all or part of this parameter in the query.	String
	ContentQuery	Search string for content matches within a template.	String
	SortBy	Optional parameter to specify how the list you retrieve is ordered. You can sort the list by: • NAME • DESCRIPTION	String
	ResponseGroup	Optionally return certain response groups, including • TemplateList:	ResponseGroup
Response			
doGetTemplateListResponse	Result	Returns *Success* or an error message	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	TemplateList	A list of the available story templates.	TemplateList
	StartIndex	The index (1-based) of the first search result in TemplateList.	Integer
	EndIndex	The index (1-based) of the last search result in TemplateList	Integer

	TotalResults	The total number of results that exist for the search request.	Integer
	SearchTime	The total amount of time the service took to complete the search in seconds.	String
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doGetTemplateListData

Use this service to get schema for a given TemplateList, including details about the Story, StoryFragments, and Fields information for the TemplateList. This service is useful if you want to map field-level data to a document package.

The ComposeData type in the response contains a full aggregate of schema for each story in the TemplateList, which lets you interrogate any portion of the schema for varying types and scope of elements.

This web service is non-stateful in nature and accepts optional user identification for library profiling purposes.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetTemplateListDataRequest	AuthUser	Optional user identification.	String
	LibraryId	Required library selection (ID).	String
	TemplateList	A list of the available story templates.	TemplateList
	EffectiveDate	Optional date qualifier.	String
	ResponseGroup	Optionally return certain response groups, including: • ComposeData	ResponseGroup
Response			
doGetTemplateListDataResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	ComposeData	A fragment of the schema for the selected list of templates – use to map data for downstream composition and publishing services.	ComposeData
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doCreateFolder

Use this service to create a remote *folder* of selected story templates you want to work on. A remote *folder* works somewhat like an e-Commerce shopping cart, except it is designed for managing an active document *package* with stateful composition.

The doCreateFolderRequest service is considered *abstract* in nature, which means it cannot be implemented. Instead, doCreateFolder supports the implementation of these underlying *concrete* types:

Type	Use this type if you...
doCreateFolder_Import	Want Oracle Insurance rules to handle the dynamic triggering of Story and StoryFragment types and mapping of data to the document. (<doCreateFolderRequest xsi:type="doCreateFolderReq_Import"…)
doCreateFolder_ComposeData	Know which story templates you need and (optionally) want to map specific data elements to the document. (<doCreateFolderRequest xsi:type="doCreateFolderReq_ComposeData"…)

This web service is stateful in nature and returns a unique FolderId to be used in subsequent requests.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Abstract Request			
doCreateFolderRequest	Owner	Identity of the document or folder owner, for identification purposes.	Owner
	CoreProperties	Core properties of the folder.	CoreProperties
	ResponseGroup	Optionally return certain response groups, including: • TemplateList • CoreProperties	ResponseGroup
Typed Request			
doCreateFolderReq_Import	ImportFile	Attachment data for the import files that drives the publishing request	ImportFileList
	TemplateList	Optional listing of selected story templates.	TemplateList
Typed Request			
doCreateFolderReq_ComposeData	ComposeData	A list of selected story templates with composition data to be merged with the active document for composition or publishing.	ComposeData
Response			

doCreateFolderResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	FolderId	Unique folder identifier.	String
	TemplateList	A list of all story templates currently in the remote folder.	TemplateList
	CoreProperties	The core properties of the folder.	CoreProperties
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doGetFolderList

Use this service to get a list of folders.

This web service is stateful in nature and requires a FolderId to maintain the state of the request.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetFolderListRequest	Owner	Identity of the document or folder owner, for identification purposes.	Owner
	LibraryId	Required library selection (ID).	String
	Start	One-based index of the first desired result.	Integer
	MaxResults	Number of results desired per query.	Integer
	DocumentIdQuery	Use to refine the search by DocumentId. You can use all or part of this parameter in the query.	String
	DescQuery	Use to refine the search by Description. You can use all or part of this parameter in the query.	String
	PropQuery	Use to refine the search by one or more custom properties. You can use all or part of this parameter in the query.	PropQueryInfo
	SortBy	Optional parameter to specify how to order the list you retrieve. You can sort the list by: • DOCUMENTID • DESCRIPTION • <PROPERTY>	String
	ResponseGroup	Optionally return certain response groups, including: • TemplateList • ComposeData	ResponseGroup
Response			
doGetFolderListResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	FolderList	A list of folders returned from the search query.	FolderListType
	StartIndex	The index (1-based) of the first search result in FolderList.	Integer
	EndIndex	The index (1-based) of the last search result in FolderList	Integer
	TotalResults	The total number of results that exist for the search request.	Integer
	SearchTime	The total amount of time the service took to complete the search in seconds.	String
Fault			

	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doGetFolder

Use this service to get the contents of a pre-existing folder.

This web service is stateful in nature and requires a FolderId to maintain the state of the request.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doGetFolderRequest	LibraryId	Required library selection (ID).	String
	FolderId	Unique folder identifier.	String
	ResponseGroup	Optionally return certain response groups, including: • TemplateList • CoreProperties • ComposeData	ResponseGroup
Response			
doGetFolderResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	Owner	Identity of the document or folder owner, for identification purposes.	Owner
	FolderId	Unique folder identifier.	String
	TemplateList	A list of all story templates currently in the remote folder.	TemplateList
	CoreProperties	The core properties of the folder.	CoreProperties
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doModifyFolder

Use this service to add, remove, re-arrange, and generally modify the contents, XML data, and/or general information of a folder.

Here are the rules that apply to the use of doModifyFolder:

Rule	Description
CoreProperties	Any property item omitted means the pre-existing property in the folder remains intact. Additionally, any property passed as a blank/empty value indicates the pre-existing property in the folder should be cleared — subject to certain underlying publishing rules, wherein the clearing of a property would effectively invalidate the folder in the system.
ComposeData	If ComposeData is included in the request, it means there is an intention to update/modify the document itself. If ComposeData is included and there is a mismatch between Story items, ComposeData determines the new document packaging (overwrites the current document). Additionally, if a field item is included in the request, it means there is an intention to update/modify the same field in the folder. Conversely, omitting field items in the request indicates there is an intention to preserve the current content of the field in the folder.

This web service is stateful in nature and requires a FolderId to maintain the state of the request.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doModifyFolderRequest	Owner	Identity of the document or folder owner, for identification purposes.	Owner
	LibraryId	Required library selection (ID).	String
	FolderId	Unique folder identifier.	String
	ComposeData	Optional fragment of composition data to be injected/merged with the active document for composition or publishing	ComposeData
	CoreProperties	Modified core properties of the folder.	CoreProperties
	ResponseGroup	Optionally return certain response groups, including: • TemplateList • CoreProperties	ResponseGroup
Response			
doModifyFolderResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList

	FolderId	Unique folder identifier.	String
	TemplateList	A list of all story templates currently in the remote folder.	TemplateList
	CoreProperties	The core properties of the folder.	CoreProperties
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doDeleteFolder

Use this service to delete a folder.

This web service is stateful in nature and requires a FolderId to make the request.

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Request			
doDeleteFolderRequest	LibraryId	Required library selection (ID).	String
	FolderId	Unique folder identifier.	String
Response			
doDeleteFolderResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

doCallAPI

Use this service operation to submit any request type to a provider such as Docupresentment. This service operation allows more flexibility than the other service operations discussed in this document by providing more abstraction of a request payload that can be submitted to a provider. For instance, you can submit and return any number of name/value pairs, collections, and attachments. This also means that unlike the other service operations discussed in this document, doCallAPI does not lend itself well to the definition of a well-defined service. Only use this service when one of the other service operations does not provide the necessary functionality.

NOTE: This service operation is only supported in Java.

Parameter	Description	Type	Occurrence
Request payload (doCallAPIRequest) elements			
schemaVersion	The schema version to use. The default is 1.0, but version 1.1 can also be used.	schemaVersion Enum	0…1
timeOut	The timeout value in seconds for the service operation to receive a response from the provider. The default is 30 seconds.	int	0…1
ProviderName	The provider name for the operation that should be invoked. The default is *IDSProvider*, which represents a Docupresentment server as a provider.	string	1
Operation	The provider operation that should be invoked. The default is *processRequest* but you can also provide a value of *Discovery* to ask the doCallAPI service operation to return a list of supported operations for the provider.	string	1
Props	A list of *Prop*, *FileProp* and *Collection* elements to send to the provider.See the definition of each for more details.	PropertyList	1
Prop	A name/value pair to send to the provider for a specific operation. You can submit a *Prop* name/value pair of name *Discovery* and value *true* to ask doCallAPI service operation to return additional information about the expected *Prop* elements for the *Operation* value provided. Here are the attributes: • name – The name of the name/value pair. • Text – The value of the name/value pair. Here is an example: `<Prop name="Reqtype">SSS</Prop>`	PropertyInfo	0…many

Parameter	Description	Type	Occurrence
FileProp	A file attachment to send to the provider. The attachment can be inline base64 content or a valid HTTP or File URL to a file accessible by the doCallAPI service operation. Here are the attributes: • name – The name of the attachment. • *location* – The location of the attachment. Use *ATTACH* if the content is provided as base64 inline text. Use *URL* if the content will be provided by an HTTP or File URL. • URLLocation – only present when the type is PropFile_URL and should contain the value of the HTTP or File URL. • contentType – You can omit this value. It is only present because the base class used for PropFile_ATTACH and PropFile_URL types contains this attribute. • Text – None (blank) when *location* is set equal to *URL*, otherwise, the base64 inline text content of the attachment when *FileProp* element is a child of *Props* element and none (blank) when it is a child of *ResponseProps* element. Here is an example: `<FileProp xsi:type="contract:PropFile_URL" location="URL" URLLocation="file:///c:/test.xml"  name="MyAttachment"/>`	PropFile_ATTACH or PropFile_URL	0…many
Collection	A collection of items to send to the provider. Each item can contain one or more columns with a name and value. You can also think of a collection as a row set with one or more rows, each row containing one or more name/value pairs. Here are the attributes: • name – The name of the collection • Text – None (blank) Here is an example: `<Collection name="My Collection"><Item name="My Item 1"><Column name="name1">value1</Column><Column name="name2">value2</Column></Item><Item name="My Item 2"><Column name="name1">value1</Column><Column name="name2">value2</Column></Item></Collection>`	CollectionType	0…many

Parameter	Description	Type	Occurrence
ResponseProps	An element that defines how attachments should be returned by the provider. As such, it contains one or more *FileProp* elements. See the definition of *FileProp* for more information. Here is an example: `< ResponseProps> <FileProp xsi:type="contract:PropFile_ATTACH" location="ATTACH" name="ATC1"/> <FileProp xsi:type="contract:PropFile_URL" location="URL" URLLocation="file:///c:/ test.xml" name="ATC2"/> </ResponseProps>`	ResponsePropertyList	0…many
Response payload (doCallAPIResponse) elements			
Props	A list of *Prop*, *FileProp* and *Collection* elements returned by the provider. See the definition of each for more details.	PropertyList	1
Prop	A name/value pair returned by the provider. Here are the attributes: • name – The name of the name/value pair. • Text – The value of the name/value pair. Here is an example: `<Prop name="Reqtype">SSS</Prop>`	PropertyInfo	0…many
FileProp	A file attachment returned by the provider. The attachment can be inline base64 content or a valid HTTP or file URL. Here are the attributes: • name – The name of the attachment. • location – The location of the attachment. Will be *ATTACH* if the content is returned as base64 inline text, otherwise *URL* if the content is returned as an HTTP or file URL. • URLLocation – Only present when the type is PropFile_URL and should contain the value of the HTTP or file URL. • contentType – You can omit this value. It is only present because the base class used for PropFile_ATTACH and PropFile_URL types contains this attribute. • Text – None (blank) when *location* is set equal to *URL*, otherwise, the base64 inline text content of the attachment returned by the provider. Here is an example: `<FileProp xsi:type="contract:PropFile_URL" location="URL" URLLocation="file:///c:/ test.xml"  name="MyAttachment"/>`	PropFile_ATTACH or PropFile_URL	0…many

Parameter	Description	Type	Occurrence
Collection	A collection of items returned by the provider. Each item can contain one or more columns with a name and value. You can also think of a collection as a row set with one or more rows, each row containing one or more name/value pairs. Here are the attributes: • name – The name of the collection • Text – None (blank) Here is an example: <Collection name="My Collection"> <Item name="My Item 1"> <Column name="name1"> value1 </Column> <Column name="name2"> value2 </Column> </Item> <Item name="My Item 2"> <Column name="name1"> value1 </Column> <Column name="name2"> value2 </Column> </Item> </Collection>	CollectionType	0…many

Here is an example request payload:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns="http://
www.w3.org/2005/05/xmlmime" xmlns:contract="http://
webservices.docucorp.com/ewps/schema/2005-12-01">
  <soap:Body>
    <contract:doCallAPIRequest schemaVersion="1.0">
      <contract:timeOut>30</contract:timeOut>
      <contract:ProviderName>IDSProvider</contract:ProviderName>
      <contract:Operation>processRequest</contract:Operation>
      <contract:Props>
        <contract:Prop name="REQTYPE">ECH</contract:Prop>
        <contract:Prop name="Foo">Foo</contract:Prop>
        <contract:FileProp xsi:type="contract:PropFile_ATTACH"
location="ATTACH" name="ATTACHMENT1">
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxtZXNzYWdlPg0
KCTxkYXRh
Pg0KCSAgPHZhciBuYW1lPSJDT05GSUciPkFNRVJHRU48L3Zhcj4NCgkgIDx2YXIgbmF
tZT0iVVNF
UklEIj5GT1JNQUtFUjwvdmFyPg0KCSAgPHZhciBuYW1lPSJQQVNTV09SRCI+Rk9STUF
LRVI8L3Zh
cj4NCgkgIDx2YXIgbmFtZT0iUkVRVFlQRSI+VEVTVDwvdmFyPg0KCSAgPHZhciBuYW1
lPSJEWU5B
TUlDLUNPTkZJR1VSQVRJT04tRklMRSI+QzpcZG9jc2Vydlx0ZXN0ZmlsZXNcdGVzdC1
jb25maWct
```

```
ZmlsZS54bWw8L3Zhcj4NCgk8L2RhdGE+DQoJPGF0dGFjaG1lbnRzPg0KCQk8ZmlsZSB
uYW1lPSJB
VFRBQ0hNRU5UMSI+QzpcZG9jc2Vydlx0ZXN0ZmlsZXNcaW1wb3J0LnhtbDwvZmlsZT4
NCgk8L2F0
dGFjaG1lbnRzPgkNCjwvbWVzc2FnZT4NCg==</contract:FileProp>
        <contract:FileProp xsi:type="contract:PropFile_URL"
location="URL" URLLocation="http://localhost:8080/ewps-axis2/cache/
test.xml" name="ATTACHMENT2"/>
        <contract:FileProp xsi:type="contract:PropFile_URL"
location="URL" URLLocation="file:///c:/java/tomcat/webapps/ewps-
axis2/cache/test.xml" name="ATTACHMENT3"/>
        <contract:Collection name="Collection1">
          <contract:Item name="Item1">
            <contract:Column name="column1">value1</contract:Column>
            <contract:Column name="column2">value2</contract:Column>
          </contract:Item>
          <contract:Item name="Item2">
            <contract:Column name="name1">string1</contract:Column>
            <contract:Column name="name2">string2</contract:Column>
          </contract:Item>
        </contract:Collection>
        <contract:Collection name="Collection2">
          <contract:Item name="Coll1">
            <contract:Column name="testname1">testvalue1</
contract:Column>
            <contract:Column name="testname2">testvalue2</
contract:Column>
          </contract:Item>
          <contract:Item name="Coll2">
            <contract:Column name="myname1">stringvalue1</
contract:Column>
            <contract:Column name="myname2">stringvalue2</
contract:Column>
          </contract:Item>
        </contract:Collection>
      </contract:Props>
      <contract:ResponseProps>
          <contract:FileProp name="ATTACHMENT1" location="URL"
URLLocation="file:///c:/temp/test1.xml"/>
          <contract:FileProp name="ATTACHMENT2" location="ATTACH"/>
          <contract:FileProp name="ATTACHMENT3" location="URL"
URLLocation="file:///c:/temp/test2.xml"/>
      </contract:ResponseProps>
    </contract:doCallAPIRequest>
  </soap:Body>
</soap:Envelope>
```

Here is the corresponding response payload example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/">
   <soapenv:Body>
     <doCallAPIResponse xmlns="http://webservices.docucorp.com/ewps/
schema/2005-12-01">
         <Props>
            <Prop name="Foo">Foo</Prop>
            <Prop name="IDSGUID">ef87ae86d4d53eb5010b6791190894f1</
Prop>
```

```
            <Prop name="IDSHOSTNAME">jrobertsnb1</Prop>
            <Prop name="REQTYPE">ECH</Prop>
            <Prop name="SERVERTIMESPENT">0.000</Prop>
            <Prop name="SERVERTIMESPENTMS">0</Prop>
            <Prop name="Timeout">30</Prop>
            <Prop name="ServiceResults">SUCCESS</Prop>
            <Prop name="ServiceTimeMillis">125</Prop>
            <FileProp name="ATTACHMENT1" location="URL"
URLLocation="file:///c:/temp/test1.xml"/>
            <FileProp name="ATTACHMENT2"
location="ATTACH">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiP
z4NCjxtZXNzYWdlPg0KCTxkYXRh
Pg0KCSAgPHZhciBuYW1lPSJDT05GSUciPkFNRVJHRU48L3Zhcj4NCgkgIDx2YXIgbmF
tZT0iVVNF
UklEIj5GT1JNQUtFUjwvdmFyPg0KCSAgPHZhciBuYW1lPSJQQVNTV09SRCI+Rk9STUF
LRVI8L3Zh
cj4NCgkgIDx2YXIgbmFtZT0iUkVRVFlQRSI+VEVTVDwvdmFyPg0KCSAgPHZhciBuYW1
lPSJEWU5B
TUlDLUNPTkZJR1VSQVRJT04tRklMRSI+QzpcZG9jc2Vydlx0ZXN0ZmlsZXNcdGVzdC1
jb25maWct
ZmlsZS54bWw8L3Zhcj4NCgk8L2RhdGE+DQoJPGF0dGFjaG1lbnRzPg0KCQk8ZmlsZSB
uYW1lPSJB
VFRBQ0hNRU5UMSI+QzpcZG9jc2Vydlx0ZXN0ZmlsZXNcaW1wb3J0LnhtbDwvZmlsZT4
NCgk8L2F0
dGFjaG1lbnRzPgkNCjwvbWVzc2FnZT4NCg==</FileProp>
            <FileProp name="ATTACHMENT3" location="URL"
URLLocation="file:///c:/temp/test2.xml"/>
            <Collection name="Collection1">
               <Item name="1">
                  <Column name="column1">value1</Column>
                  <Column name="column2">value2</Column>
               </Item>
               <Item name="2">
                  <Column name="name1">string1</Column>
                  <Column name="name2">string2</Column>
               </Item>
            </Collection>
            <Collection name="Collection2">
               <Item name="1">
                  <Column name="testname1">testvalue1</Column>
                  <Column name="testname2">testvalue2</Column>
               </Item>
               <Item name="2">
                  <Column name="myname1">stringvalue1</Column>
                  <Column name="myname2">stringvalue2</Column>
               </Item>
            </Collection>
         </Props>
      </doCallAPIResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Configuring the Provider

Each provider supported by the doCallAPI service operation contains a configuration section with the same name as that of *ProviderName* inside the ewps.config.xml configuration file. You can use this section to configure connection properties for each provider. For example, the IDSProvider section for Docupresentment contains the same configuration properties as those of the queue configuration properties for Docupresentment. Here is an example:

```
<IDSProvider>
   <entry
name="marshaller.class">com.docucorp.messaging.data.marshaller.SOAP
MIMEDSIMessageMarshaller</entry>
   <entry
name="queuefactory.class">com.docucorp.messaging.http.DSIHTTPMessag
eQueueFactory</entry>
   <entry name="http.url">http://localhost:49152</entry>
   <entry name="http.reuse.ports">15</entry>
   <entry name="http.putmessage.tries">15</entry>
   <entry name="timeout">30</entry>
</IDSProvider>
```

The request payload can provide the configuration properties for the provider, overriding the properties defined in the ewps.config.xml configuration file. The configuration properties should be provided as a collection with the same name as that of *ProviderName*. Here is an example of a request payload that does that:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns="http://
www.w3.org/2005/05/xmlmime" xmlns:contract="http://
webservices.docucorp.com/ewps/schema/2005-12-01">
  <soap:Body>
    <contract:doCallAPIRequest schemaVersion="1.0">
      <contract:ProviderName>IDSProvider</contract:ProviderName>
      <contract:Operation>processRequest</contract:Operation>
      <contract:Props>
        <contract:Prop name="REQTYPE">SSS</contract:Prop>
        <contract:Prop name="CONFIG">DOCCDEMO</contract:Prop>
        <contract:Collection name="IDSProvider">
          <contract:Item name="Properties">
              <contract:Column
name="marshaller.class">com.docucorp.messaging.data.marshaller.SOAP
MIMEDSIMessageMarshaller</contract:Column>
              <contract:Column
name="queuefactory.class">com.docucorp.messaging.http.DSIHTTPMessag
eQueueFactory</contract:Column>
           <contract:Column name="http.url">http://127.0.0.1:49152</
contract:Column>
          </contract:Item>
        </contract:Collection>
      </contract:Props>
      <contract:ResponseProps>
      </contract:ResponseProps>
    </contract:doCallAPIRequest>
  </soap:Body>
</soap:Envelope>
```

doPublish

Use this service to publish a composed document, either from a stateful transaction or via an imported transaction.

A stateful request simply requires a FolderId, which is the identifier to handle a folder-based publishing request with pre-selected story templates in a ComposeData structure. Stateless requests can be driven by an import file (such as XML or a pre-defined extract) or the selection of desired story templates in a folder.

The doPublishRequest is considered *abstract* in nature, which means it cannot be implemented. Instead, doPublish supports the implementation of two underlying *concrete* types, as follows:

Type	Publishes a document from...
doPublish_Import	An import file (<doPublishRequest xsi:type="doPublishReq_Import"…)
doPublish_FolderId	A FolderId (<doPublishRequest xsi:type="doPublishReq_FolderId"…)

Scenario	Synchronous Request/Response using SOAP over HTTP
Message style	doc/literal

Operation/Message types

The following operation/message types should be supported and follow the synchronous request/response scenario:

Message	Parameter	Description	Type
Abstract Request			
doPublishRequest	LibraryId	Optional library selection identifier (required for SourceType=IMPORT or FOLDERID).	String
	DistributionOptions	Required grouping that specifies various publishing and distribution options.	DistributionOptions
Typed Request			
doPublishRequest_Import	SourceType	Fixed identifier for the type of publishing request (IMPORT).	String
	Import	Attachment data for the import files that drives the publishing request.	ImportFileList
Typed Request			
doPublishRequest_FolderId	SourceType	Fixed identifier for the type of publishing request (FOLDERID).	String
	FolderId	Unique identifier for the remote Folder	String
Typed Request			
doPublishRequest_ComposeData	SourceType	Fixed identifier for the type of publishing request (COMPOSEDATA).	String

	ComposeData	A list of selected story templates with composition data to be merged with the active document for composition or publishing.	ComposeData
Response			
doPublishResponse	Result	Returns *Success* or an error message.	String
	Errors	A list of the errors returned if the request completed, but not 100% successfully.	ErrorList
	DistributionResults	The DistributionResults group contains information pertaining to recipients and their published document results.	DistributionResults
Fault			
	BadRequest	An exception because of a bad request or malformed parameters.	Client
	ServiceException	An exception because of server problem or configuration.	Server

Chapter 5

Additional Resources

The following resources provide information about SOAP, JSON, and web services in general, as well as other useful topics:

The definitions within various sections of this document are taken from several of these resources.

SOAP

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.

The framework has been designed to be independent of any particular programming model and other implementation specific semantics. SOAP supports message security, attachment, routing, reliability, and choreography.

W3C: http://www.w3.org/TR/soap12-part1/

SOAP Messaging Framework:

W3C: http://www.w3.org/TR/SOAP

W3Schools SOAP Tutorial:

W3Schools: http://www.w3schools.com/soap/default.asp

Web Services

Web services is a technology that lets applications communicate with each other in a platform- and programming language-independent manner.

A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another web service.

Web services promise to increase interoperability and lower the costs of software integration and data-sharing with partners. As they are based on simple and non-proprietary standards, web services make it possible for computer programs to communicate directly with one another and exchange data regardless of location, operating systems, or languages.

IBM: http://www-106.ibm.com/developerworks/webservices/newto/

References and Projects

IBM developerWorks Web Services
IBM Corporation

http://www-136.ibm.com/developerworks/webservices

O'Reilly Web Services
O'Reilly & Associates, Inc.

http://webservices.oreilly.com

Microsoft Web Services
Microsoft Corporation

http://msdn.microsoft.com/webservices

XML and Web Services
Microsoft Corporation

http://msdn2.microsoft.com/en-us/library/ms950421.aspx

Java Technology and Web Services
Sun Microsystems, Inc.

http://java.sun.com/webservices

Apache Web Services Project
The Apache Software Foundation

http://ws.apache.org

JSON
JSON.org

http://www.json.org

Web Services Standards and Specifications

Web Services Interoperability Organization
WS-I

http://www.ws-i.org

Web Services Activity
W3C

http://www.w3.org/2002/ws

OASIS
OASIS

http://www.oasis-open.org/home/index.php

Other Resources

Web Services Architect

http://www.webservicesarchitect.com

SOA World Magazine

http://webservices.sys-con.com

WebServices.org

http://www.webservices.org

Dr. Dobbs Journal – SOA, Web services, and XML

http://www.ddj.com/dept/webservices

Web Services Description Language

Web Services Description Language (WSDL) is an XML-based service description on how to communicate using web services. The WSDL defines services as collections of network endpoints, or ports. WSDL specification provides an XML format for documents for this purpose.

The abstract definition of ports and messages is separated from their concrete use or instance. This allows the reuse of these definitions. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Messages are abstract descriptions of the data being exchanged, and port types are abstract collections of supported operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding, where the messages and operations are then bound to a concrete network protocol and message format. In this way, WSDL describes the public interface to the web service.

WSDL is often used with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

Wikipedia: http://en.wikipedia.org/wiki/Web_Services_Description_Language

Using WSDL in SOAP applications:

IBM: http://www-128.ibm.com/developerworks/library/ws-soap/?dwzone=ws

Understanding WSDL:

Microsoft: http://msdn2.microsoft.com/en-us/library/ms996486.aspx

An overview of WSDL:

Sun: http://developers.sun.com/sw/building/tech_articles/overview_wsdl.html

Apache Axis2 User Guide:

Apache: http://ws.apache.org/axis2/1_1_1/userguide.html

Using the XML Configuration File

EWPS uses a file to set up configuration options, including how it communicates with Docupresentment. The default behavior is to communicate with Docupresentment over HTTP, on port 49152, with EWPS and Docupresentment on the same machine.

There are several ways to have EWPS change where it looks for the configuration file. Based on your application server and clustering setup, you should select the approach that best fits your needs.

EWPS first looks for the following JVM system property:

```
ewps.config.url
```

If this is set, EWPS looks for the configuration file at the URL specified in the setting. If the configuration file is in a file on the local machine, use the *file*: url naming scheme. Here is an example:

```
file:///c:/configurations/ewps.config.xml
```

If the system property is not set or if there was an error retrieving the configuration file, EWPS searches its Java classpath for a file named:

```
ewps.config.xml
```

With some application servers, such as Tomcat, the file can be located in one of these EWPS directories...

- /WEB-INF/classes directory
- /WEB-INF/lib directory (if packaged in a JAR file)

Other application servers, such as WebSphere, have options to set up shared libraries that can be added to a web application's classpath but still remain external to the web application's deployment.

If none of these options find a configuration file, EWPS looks at the *init.file* context parameter in the web application's web.xml file. Some application servers allow access and the editing of this file directly after web application deployment, while others have administration consoles to allow the editing of parameters in the web application.

If the parameter is in the form of a URL, EWPS looks for the configuration file at the URL specified in the setting. If the parameter is not a URL, it is assumed to be a file in the context and file structure of the web application. The default value for the *init.file* context parameter is shown here:

```
/WEB-INF/xml/ewps.config.xml
```

ewps.config.xml file

The ewps.config.xml file contains the configuration for the message bus provider. This file is located in the ewps-axis2.war's WEB-INF/services/EWPSService.aar (Axis2 Archive – Java Archive format) file, under its root.

You can modify this file to define the different message bus providers that Docupresentment (IDS) is configured to listen on. The ewps.config.xml file contains configuration examples for the following:

- JMS/ActiveMQ
- WebSphere MQ
- MSMQ (Windows only)
- IDS HTTP

You can find these message bus configuration examples under the <EWPS><Core><queuemanager> section of the XML file. Here is an example of the ewps.config.xml file:

```
<EWPS>
 <Core>
  <queuemanager>
    <!-- MESSAGING and QUEUE nodes are used for setting communication
to Docupresentment. Refer to Docupresentment documentation for
possible values -->
    <!-- ***Settings for ActiveMQ JMS setup *** -->
    <entry name="queuefactory.class">com.docucorp.messaging.jms.
DSIJMSJNDIMessageQueueFactory</entry>
    <entry name="jms.inputqueue.connectstring">resultq</entry>
    <entry name="jms.outputqueue.connectstring">requestq</entry>
    <entry name="jms.qcf.name">queueConnectionFactory</entry>
    <entry name="jms.initial.context.factory">org.apache.activemq.
jndi.ActiveMQInitialContextFactory</entry>
    <!-- ***Settings for IDS http connection*** -->
    <!--
    <entry name="queuefactory.class">com.docucorp.messaging.
http.DSIHTTPMessageQueueFactory</entry>
    <entry name="http.url">http://localhost:49152</entry>
      -->
    <!-- ***Settings for WebSphereMQ connection*** -->
    <!--
    <entry name="queuefactory.class">com.docucorp.messaging.
mqseries.DSIMQMessageQueueFactory</entry>
    <entry name="mq.queue.manager">QM.server1</entry>
    <entry name="mq.inputqueue.name">resultq</entry>
    <entry name="mq.inputqueue.maxwaitseconds">5</entry>
    <entry name="mq.outputqueue.name">requestq</entry>
    <entry name="mq.tcpip.host">10.1.10.159</entry>
    <entry name="mq.queue.channel">SCC1.server1</entry>
    <entry name="mq.tcpip.port">1415</entry>
      -->
    <!-- ***Setings for MSMQ connection*** -->
    <!--
    <entry name="queuefactory.class">com.docucorp.messaging.msmq.
DSIMSMQMessageQueueFactory</entry>
    <entry name="msmq.server.name">localhost</entry>
    <entry name="msmq.inputqueue.name">DIRECT=TCP:10.1.10.178\
private$\resultq</entry>
```

```
    <entry name="msmq.outputqueue.name">DIRECT=TCP:10.1.10.178
\private$\requestq</entry>
    <entry name="msmq.timeout">30000</entry>
    <entry name="msmq.expiry">1800000</entry>
    <entry name="msmq.debuglevel">2</entry>
      -->
    </queuemanager>
  </Core>
</EWPS>
```