ORACLE®

Oracle® Documaker

Documaker Web-Enabled
Solutions

Part number: E51709-01

Version 12.3

March 2014

ORACLE®

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.

Copyright © 2000-2009 The Apache Software Foundation. All rights reserved.

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.

Copyright © 2009, Berkeley Software Distribution (BSD)

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.

Copyright (C) 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved.

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.

Copyright © 2009 MIT

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.

Copyright (c) 1995-2005 Jean-loup Gailly and Mark Adler

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 product includes software developed by W3C.

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

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

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

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.

Contents

Advanced Topics

Appendix A, Setting Up Print Preview with Tomcat as the Mid-tier

Appendix B, Setting Up Print Preview with WebSphere 5.1 as the Mid-tier

Appendix C, Modifying the GLOBAL.XML File

Appendix D, Optimizing Performance

Chapter 1

Web-Enabled Solutions

This document discusses Oracle Documaker web-enabled solutions that use WIP Edit, the web-enabled version of Documaker Workstation, such as iPPS and iDocumaker Workstation.

The iPPS and iDocumaker Workstation solutions let you import and export data, create, modify, print, and archive transactions via the Internet.

WIP Edit is a browser-based application for the Windows workstation. It lets you edit transactions via the internet in a *what you see is what you get* (WYSIWYG) format.

In this document you will find details about the iPPS and iDocumaker components, features, and a brief definition of the markets served by these products.

This chapter includes these topics:

BUSINESS SCENARIOS

There are several configuration scenarios that are designed to meet your needs. This document discusses those that use WIP Edit.

Previously, the iPPS solution was sold to the MGA (Managing General Agents) market. Oracle Documaker now typically sells the iDocumaker Workstation solution to insurance carriers and other large enterprises.

Standard IPPS - What your Agents, Customers, or End Users See

With this configuration your users access a Windows based web site, log-on, access HTML forms to select lines of businesses, and so on, that determine which forms to fill in. The forms are presented to the user in HTML format in a browser. The end user can save the transactions to WIP for future use and also print and archive the transactions.

Typically, an end user enters or imports data from an XML file onto the forms, then reviews the information on screen, and prints the transaction to a local printer for further review. The end user may want to print the transaction as a PDF file and send this file via email to a customer or co-worker for review and approval. After approval, the end user can finalize the transaction and send it to archive.

The archived data can be accessed via the end user for endorsements, renewals, or correspondence. Additionally, customer service people or the end user can help with any customer questions by accessing the archived data on-line in PDF WYSIWYG format to see exactly what a customer has received.

This approach uses HTML forms throughout the process. One benefit to using HTML forms is that your end users may be able to perform rapid data entry tasks where collecting data quickly is more important than seeing exactly how it will be presented on a form.

Prior to the 11.0 Documaker Bridge, with this configuration you would maintain a master library of resources (MRL) and convert those forms into a set of HTML files. Now, the 11.x and higher Documaker Bridge generates HTML files dynamically from the form set in the MRL.

Standard iDocumaker Workstation - What your Agents, Customers, or End Users See

With this configuration your users access a Windows or UNIX-based web site, log on, access HTML forms to select lines of businesses, and so on, that determine which forms to fill in. The forms are presented in HTML format via a browser. The end user can save the transactions to WIP for future use and also print and archive the transactions.

Typically, an end user may want to enter or import data from an XML file onto the forms, then preview the information on screen, and print the transaction to a local printer. The end user may then want to create a PDF file of the transaction and send the PDF file via email to a customer or co-worker for review and approval. After approval, the end user can then finalize the transaction and send it to archive.

iDocumaker Workstation users who have licensed Documaker Server can also submit a job to a batch process to print the data for multiple recipients on high speed Xerox or AFP printers. The final copies are ready to be mailed to the customers, agents, and so on.

Finally the archived data can be accessed via the end user for endorsements, renewals, or correspondence. Additionally, customer service people or the end user can help with any customer questions by accessing the archived data on-line in PDF format to see exactly what a customer has received.

This approach uses HTML forms throughout the process. One benefit to using HTML forms is that your end users may be able to perform rapid data entry tasks where collecting data quickly is more important than seeing exactly how it will be presented on a form.

With this configuration you maintain a master library of forms (MRL) and convert those forms into a set of HTML files.

WHAT YOUR AGENTS, CUSTOMERS, OR END USERS SEE WHEN YOU ADD WIP EDIT

- Install the plug-in application from the edelivery.oracle.com.With the installed Configurations, log on to the website and enter appropriate values in the Customer Transaction Information screen.Open the Form Selection screen to select various forms associated with the MRL, Lines of Business etc and choose the forms you want to complete and then select the Work On Forms option to edit the documents.

- Within this view using the WIP Edit Plug-In 12.0 and higher

- The navigation tree will show the list of forms within the document, but not the recipient associated with them.

- The navigation tree also shows the required field status of the document and allows users to activate a check required field function via right click menu option.

- WIP Edit plugin supports the Can Split section level rule. When defined on a section in the library, the plug-in will allow the section to dynamically split across pages as its pushed down by additional the content added above it. When content above the section is removed, the split section will also merge back into one section on the first page.

- By default, users can only view forms with editable fields. To allow users to view all forms, both variable and static/print only, enable the Control group option ViewPrintOnly within the WIPEdit.ini and set the value to Yes. In versions 12.0 and 12.1 you will also need to set the LoadPrintOnly option to Yes. Starting in 12.2, all print only forms are loaded by default so the second option is no longer needed.

- On completion of updating/editing the fields, the user is directed to Processing Options page to complete the printing process.

IMPROVING THE PROCESS

- To take full advantage of iPPS or iDocumaker Workstation with WIP Edit you need version 3.11 or higher. This version and all subsequent patch levels of the web-based product set let you use new Documaker 11.x shared objects, WIP Edit, a single MRL, and 11.0 and higher Studio format files.

- This means is less resource maintenance for you while letting your end users work with the familiar interface and form sets you provide. Your end users install WIP Edit, then use the application each time they work with transactions.

- This document will help you with the setup process.

NOTE: Version 3.1 will continue to be supported.

Operating system support

- Documaker iPPS and iDocumaker Workstation systems let you install and configure the COM+ system for Windows operating systems.

- iDocumaker Workstation system lets you install and configure the Java (J2EE) system for UNIX operating systems as well as Microsoft Windows.

- This document reviews the server side and client side setup tasks you have to perform. Assumptions about the software applications and versions are also discussed. Parts of this document describe how to install and configure the various components, how to run a simple test, and how to troubleshoot the system to make sure the test works accurately.

- Product upgrades

- Oracle Documaker has a product upgrade path that usually consists of applying patches to existing installations. Steps guide you through the process necessary to apply base product patches from Oracle Documaker, modify your system, and reapply custom file settings accordingly.

ARCHITECTURE OVERVIEW

- This topic provides an architectural overview of Documaker web-enabled solutions. This discussion focuses on these areas:

- iPPS core system

- iPPS for managing general agents

- iDocumaker Workstation

iPPS Core System

- Oracle Documaker is a provider of web-enabled electronic document correspondence and policy production solutions. At the center of these solutions is a set of core technology called iPPS — Internet Policy Processing System.

- The *iPPS Core System* is a web application with these key features:

- Leverages a common Documaker/PPS forms electronic library

- Enables an operator to browse a list of candidate forms in the forms library

- Enables an operator to select a set of forms for data entry

- Enables an operator to enter data into variable field areas on the forms in the form set

- Enables data to be imported from an external system into the form set

- Enables the operator to create, view, and print PDF proofs of the form set

- Enables the operator to save a form set in WIP (Work-in-process)

- Enables the operator to assign form sets in WIP to other operators

- Enables the operator to complete and publish a final output document from the form set in multiple output formats and delivery methods

- Enables data to be exported from the form set to an external system

- Offers exits and hooks for integrating with an archive system

- Enables completed form sets to be archived into an archive system

- Enables the operator to browse the index of previously archived form sets

- Enables the operator to start new documents using data from previously archived form sets

- The iPPS Core System is able to display the selected forms in the form set, and accept user data entry into the forms in the form set, using either of two ways:

- Using dynamic HTML forms presented in a frame in the browser within the application

- Using Documaker *WIP Edit* component (an ActiveX control) hosted within a frame in the browser within the application

- The modules of the iPPS Core System are part of the packaging of two products:

- iPPS for the MGA market

- iDocumaker Workstation

iPPS for the MGA market

- MGAs are Managing General Agents. MGAs are independent insurance companies that act as agents for other companies and are empowered to rate, quote, and bind insurance policies on behalf of carriers, typically in the domestic excess and surplus insurance market. The excess and surplus insurance market is typically dominated by specialty or unusual-risk commercial insurance policies.

- Oracle Documaker sells a product in the MGA market called PPS — Policy Processing System. iPPS for the MGA market is a web-enabled version of PPS.

- iPPS requires a license to PPS, and is licensed by host server site, and also by each end-user client location. PPS and iPPS are designed for use with electronic libraries provided by licensed carriers. Additional licensing of development tools may be required.

- iPPS for the MGA market consists of these components:

- *iPPS Core System* for one of the following platforms:

- Windows 2000 Server (COM+ technology)

- Windows 2003 Server (COM+ technology)

- Documaker WIP using a standard xBase index

- Documaker archive system using a standard xBase index and flat CAR repository

- A subset of Docupresentment (for Windows only) consisting of these components:

- Internet Document Server (IDS)

- Bridge to Documaker WIP and Archive

- HTML conversion tools

  - PDF driver

iDocumaker Workstation

- Insurance carriers and other large enterprises can license an expanded version of the same core technology found in PPS in a product called Documaker Workstation. Similar to how iPPS is a web-enabled version of PPS, iDocumaker Workstation is a web-enabled version of Documaker Workstation.

- iDocumaker Workstation is sold as component of the following solutions:

- Policy Generation web-enabled

- Policy Generation web-enabled with Archive

- Correspondence web-enabled

- Correspondence web-enabled with Archive

- iDocumaker Workstation consists of the following:

- iPPS Core System for one of the following platforms:

- Windows 2000 Server (COM+ technology)

- Windows 2003 Server (COM+ technology)

- AIX (J2EE technology)

- Solaris (J2EE technology)

- HP-UX (PA-RISC) (J2EE technology)

- Intel PC-based Linux (J2EE technology)

- Documaker WIP using one of these index methods:

- Standard XBASE index

- SQL database index

- Interface to archive system (Documanage sold separately)

- The full Docupresentment system (Docupresentment sold separately)

USING WIP EDIT VS. HTML FORMS

Once the mid-tier and back end servers are configured, you decide which client best suites your organizational requirements. You can choose between an HTML view of your forms and data or a WYSIWYG view. This topic provides guidelines to help you decide which approach better suits your needs.

Use HTML when:

Pure thin-client is a requirement (other than Acrobat)

Users that are not known to the enterprise may need to enter onto the forms

Forms involve relatively standard forms-fill and not a lot of field-level editing

A reasonably accurate rendition of the form is required, but not an exact WYSIWYG representation

Use WIP Edit when:

You have an existing Documaker or PPS library that is rich with user edits and scripts already exists

The only users are internal users known to the enterprise and fidelity is more important than being thin-client

The forms are highly dynamic, word-processor type forms, or have barcodes, charts, or other features not available in HTML

HOW WIP EDIT, IDS, AND DOCUMAKER WORKSTATION INTERACT

Figure 1 shows Oracle Documaker Thin Client architecture from a logical view. This represents a fully configured system in terms of the thin client, mid-tier, and back-end servers in a physical implementation. It also references the Master Resource Library (MRL) you create to use with the system.

Prior to the 11.0 Documaker Bridge, if you created an iPPS application you had to create your MRL and then generate HTML files that corresponded to your forms, correspondence, and so on. If you changed the forms in the MRL, you also had to change the HTML files. For large applications, keeping these entities in sync could be a problem.

The 11.x and higher Documaker Bridge solves this problem by automating the process. It now queries the MRL through IDS, for a list of forms and then dynamically generates the HTML. Now you have a choice about whether you want to use HTML based forms or true WYSIWYG forms. WIP Edit provides the latter functionality by rendering the forms directly from the MRL. The previous topic provided some guidelines on when to choose HTML based forms versus WYSIWYG forms.
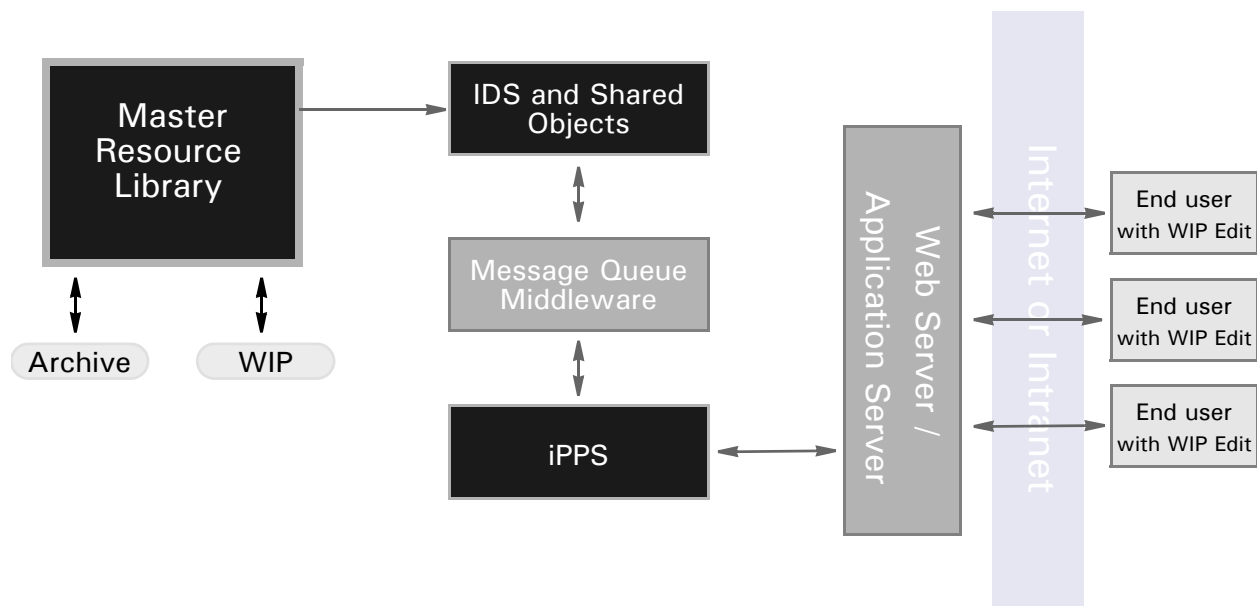
Figure 1: Oracle Documaker Thin Client Logical Architecture

Specific hardware and software platform requirements for the thin client architecture and the various software components are described elsewhere in this document.

In Figure 1, the client side consists of a workstation running Microsoft Windows, Internet Explorer and Documaker WIP Edit ActiveX component. When a user logs into the system and makes a request, it is sent to iPPS.

iPPS acts as a front end to all of the back end server components IDS (Internet Document Server) and Shared Objects. iPPS looks at the request made by the client, encodes that into a message and places it on the message queue.

IDS checks the message queue for incoming requests. When it sees a new message, it pulls it off the queue. It determines the action the client wants to perform and initiates the action. The action could be retrieving documents from archive storage, printing documents, retrieving documents from WIP, and so on. All of these actions are performed against the MRL that is in use and rule sets that constitute the business logic of the application. When the request has completed, IDS sends a result (which depends on the request type) back to iPPS. iPPS forwards the result back to the client browser. If the request is to retrieve a form set and display a particular form, WIP Edit renders a form in the user's browser.

When configuring IDS and iPPS, you typically have to define things like web server settings, queue mangers, and any custom rule sets that required for specific request types. Later in this document you will see how to configure these items.

Central to the execution of the system is an item referred to as a request type. A request type is an action the user wants to perform. A request type triggers an action like saving a document or running a spell check on this document. On the back-end server (IDS), the request types are identified in identified by files accessible to the IDS. The request types for the web application or plug-in could be put in the docserv.xml but request types are usually configured in the INI file that is defined in the docserv.xml. The file contains all of the configuration information for the IDS server. iPPS has its own configuration file called GLOBAL.XML. As the file extension implies, this is an XML file, whereas the IDS configuration file is an INI file.

Here is an example of a request type.

```
< ReqType:i_PluginInit >
;Plugin initialization. Get DPW file
 function= atcw32->ATCLoadAttachment
function= atcw32->ATCUnloadAttachment
function= atcw32->ATCSendFile,RF_POSTFILE,PRINTFILE,BINARY
function= dprw32->DPRSetConfig
 function= dprw32->DPRInitLby
function= dprw32->DPRGetWipFormset
 function= dprw32->DPRPrintDpw
```

Listing 1: Listing 1: Typical Request Type

This listing shows a typical request type This is one of the request types supported by WIP Edit. The general format of a request type is the keyword ReqType: followed by the name (with no spaces). The type is placed in square brackets.

Under the request type, a series of entries that begin with function= is found. These are the specific rules that are run by the Rules Processor component of IDS. The order of the listing is the order in which the rules are run. For a complete list of the rules, see the SDK Reference. This book contains all of the information needed to write custom rules. The document explains the purpose of each function and identifies the required input and output variables (attachment variables) associated with each function

The specific modifications you must make to the configuration files for IDS and iPPS recovered in separate topics. However, the request types listed in the configuration files will match each other. The IDS configuration files contain request types and rules that are run for each request. The GLOBAL.XML file contains request types and attachment variables. Attachment variables are input and output variables described in a name/value pair, such as USERID=JOHNSMITH.

When iPPS sends a message to IDS via the message queue middleware (such as WebSphere MQ or MSMQ) it typically packs the request type up with the attachment variables, and posts the message on the queue. When IDS is notified that a message is waiting, it retrieves the message from the queue, identifies the request type, and then begins to run the rule set identified in the IDS configuration files.

In most cases, the first rule that is run is

atcw32->ATCLoadAttachment

This rule takes the attachment variables sent by iPPS and stores them in an internal data structure. Additional code is run on the back end that corresponds to the request type.

If IDS needs to send data back to the client, it transforms the data from the internal data structures to a message for the middleware. This rule set builds up the message and places

it in the message queue:

atcw32->ATCUnloadAttachment

When IDS needs to send a file to the client, this rule is run:

function = atcw32->ATCSendFile, RF_POSTFILE, PRINTFILE, BINARY

This says IDS will post a file of type PRINTFILE that is a binary file, back to the client. The parameters for the ATCSendFile rule may vary dependent on the web page that makes the request.

This rule:

function = dprw32->DPRSetConfig" and "function= dprw32->DPRInitLby

Gets the configuration information for the print functionally and initializes a library, while this rule:

function = dprw32->DPRGetWipFormset

Retrieves the form set that was request (identified by one of the input attachment variables) by the client. IDS writes the information to a file (the extension is DPW).

Finally, this rule:

function= dprw32->DPRPrintDpw

Generates a DPW file from a WIP record and sends the file back to iPPS, which in turn forwards it on to the client. WIP Edit takes the DPW file, unpacks it, and displays information to the user. The information displayed depends on the request type.

The following chapter on Setting Up IDS and Setting Up iPPS goes into additional detail on the request types, attachment variables and typical patterns that are seen in most requests. Please refer to those topics for more information.

Chapter 2

Installation and Configuration

This chapter discusses how you install and configure iDocumaker Workstation and iPPS and includes information on these topics:

HOW TO CONFIGURE INI REQUEST TYPES

There should be an INI file defined in the docserv.xml. Request types for web application or plug-in could be put in the docserv.xml but it is usually INI file that is defined in the docserv.xml section.This is the section in the docserv.xml that defines the request types for plug-in/web app.

- <section name="INIFiles">

<entry name="File">[INIFiles:~Platform] File =</entry>

<entry name="File">idmk_java_requests.ini</entry>

</section>

Format of request type in docserv.xml

<section name="ReqType:GETRESOURCE">

<entry name="function">atcw32->ATCLogTransaction</entry>

<entry name="function">atcw32->ATCLoadAttachment</entry>

<entry name="function">atcw32->ATCUnloadAttachment</entry>

<entry name="function">dprw32->DPRSetConfig</entry>

<entry name="function">dprw32->DPRDecryptLogin</entry>

<entry name="function">dprw32->DPRDefaultLogin</entry>

<entry name="function">dprw32->DPRCheckLogin</entry>

<entry name="function">atcw32-

>ATCSendFile,RETURNFILE,RETURNFILE,Binary</entry>

<entry name="function">dprw32->DPRGetResource,RETURNFILE</entry>

<!-- -->

</section>

INI file format

; Gets a resource for the Plugin

[ReqType:iDM_PluginGetResource] function = atcw32-ATCLoadAttachment function = atcw32ATCUnloadAttachment function = dprw32->DPRSetConfig

function = atcw32->ATCSendFile,DOCUMENTSTREAM,RETURNFILE,Binary function = dprw32->DPRGetResource,RETURNFILE

function = java;com.docucorp.ids.rules.CopyDataRule;copyit;transaction;copyMessageVariables;TAG_AND_FOLLOW,CONFIG

; Returns entries from standard RP table

OVERVIEW

The installation and configuration steps are covered in detail throughout this chapter. Here is a brief overview of the steps you will perform to get your system running:

Install IDS 1.8 Patch 42 or higher and Shared Objects 12.2 or higher. See Setting Up IDS on page 34 for more information.

Install and configure MQSeries and modified the INI files to reflect the MQSeries settings.

Install WIP Edit 12.3. See Setting Up the WIP Edit Client on page 28 for more information.

If you are running iPPS Com+, make sure your Windows server has IIS installed.

Install MDAC and MSXML. See iPPS COM+ Software Prerequisites on page 50 for more information.

Install the iPPS 3.11 resources. See Setting Up iPPS on page 50 for more information.

Install the iPPS 3.11 DLL files. See Setting Up iPPS on page 50 for more information.

Install the AMERGEN resources. See MRL section on page 56 for more information.

Configure the IDS INI files for your queue manager and verify the settings identified in this document. See Setting Up IDS on page 34 for more information.

Make sure the Amergen and WIP Edit INI files are configured as shown throughout this document. See MRL section on page 56 and Changing the WIPEDIT.INI File on page 124 for more information.

Modify the GLOBAL.XML file as shown in Modifying the GLOBAL.XML File on page 115

SETTING UP THE WIP EDIT CLIENT

This topic discusses how to install and configure a WIP Edit client.

To use Oracle iDocumaker Workstation or iPPS/MGA document management solution, you need several components on the client machine. The basic components are:

Component	Description
Operating System	This must be a 32-bit Microsoft Windows-based system such as Windows 2000 Professional or Windows XP Professional.
Browser	The default output of the iPPS core system is HTML. You must have Microsoft's Internet Explorer version 6.0 and above.Documaker Interactive will support Firefox.
Adobe Acrobat Reader	WIP Edit supports version 6.0 and above. WIP Edit can generate PDF files that can be printed on the client machine through Adobe Acrobat.
WIP Edit	WIP Edit is an optional Oracle Documaker component that provides a WYSIWYG (what you see is what you get) form view from within Microsoft's Internet Explorer. This control renders forms produced by the iPPS core system directly from your MRL.

You can not have a mixed environment of WIP Edit and non-WIP Edit users. The server configuration is not dynamic depending on the client machine. All your users must either run in HTML mode or WIP Edit mode.

The WIP Edit plug-in must be at version 12.3 or higher. Oracle Insurance provides a setup program to install all of the necessary WIP Edit components, including the program files, fonts, INI files, and so on.

The installation of the third party client side basic components is documented by the vendors that supply the software. The WIP Edit component installation is described in Setting Up WIP Edit on page 29.

SETTING UP WIP EDIT

The version of WIP Edit you install should be compatible with the version of Shared Objects used on the back-end IDS server. This means that if the IDS server Shared Objects is running at version 12.3, it would be a good idea to patch the corresponding WIP Edit installation to the same revision level.

The WIP Edit installation program is identified by revision level on the Oracle Insurance Support software download web site. For example, a version of the WIP Edit setup program may be identified as ODWE12.3.00.22498W32, indicating version 12.3, and patch 00. So make sure you download the compatible version for your IDS Shared Objects installation.

The client side configuration for WIP Edit consists of these steps:

Wipedit now uses install4j program for install.

Configuring Internet Explorer for ActiveX controls.

NOTE: You must have Administrator rights to install WIP Edit. You can install and run the WIP Edit from either a local or network drive.

Installing WIP Edit

The installation program automatically installs and sets up WIP Edit. To run the installation program:

Double click on the file you downloaded from the Oracle Insurance web site, such as ODWE12.3.00.22498W32.

Follow the instructions that appear on your screen.

At the end of the installation process on a Windows 7 -64bit environment, system may display the following:

Microsoft Visual C++ 2008 Re distributable Setup dialog for Maintenance mode.

Select one of the options below:

  Repair Microsoft Visual C++ 2008 Re distributable to its original state.

  Uninstall - Uninstall Microsoft Visual C++2008 Re distributable from this computer.

Choose Repair to go ahead with the installation process.

In a scenario when the re distributable is still installed post Documaker Uninstallation, Maintenance: Repair/uninstall screen appears when you re-install Documaker.

NOTE: At the end of the installation, you do not have to reboot your computer

---

12.0 install and higher doesn't use TrueType fonts

The installation program installs and registers several TrueType fonts which are then available to any Windows program. In addition, it sets a registry value to point to the WIP Edit installation path. You must have Administrator rights on the machine when you run the install program. The entire footprint of the install, including the TrueType fonts, is approximately 25MB.

The registry value is set as shown here:

```
HKEY_LOCAL_MACHINE\\SOFTWARE\\DocuCorp International\\Docucorp
WIPEdit\\12.3\\InstallP\[install location]
```

Controlling when a font file is created

You can use the DownloadDPWFonts INI option to control whether the system creates a font file when it creates the DPW file. The default is No, which means the font file is not created.

```
< WIP2DPW >
    DownloadDPWFonts = Yes
```

If you set this option to Yes, the system includes any font used in the document in a temporary FXR file that is downloaded to WIP Edit. This may affect text edit mode if the user wants a font that is not used in the document.

If a font is needed but is not used in the document, use the File2DPW control group to download an existing FXR file to WIP Edit and then set the DownloadDPWFonts option to No, as shown here:

```
< File2DPW >
    XRFToken        = yourfxrfile.fxr
< WIP2DPW >
    DownloadDPWFonts= No
```

Configuring Internet Explorer for ActiveX controls

Once the installation program completes, you need to configure Internet Explorer to set up the permission levels for ActiveX controls.

NOTE: No ActiveX controls are downloaded when you perform this task. In this task, you are only enabling your browser to run the ActiveX control that was installed when you ran the WIP Edit installation program.

---

1 Start Internet Explorer.

2 Select Tools, Internet Options from the main menu and click the Security tab. A window similar to the one shown here appears:
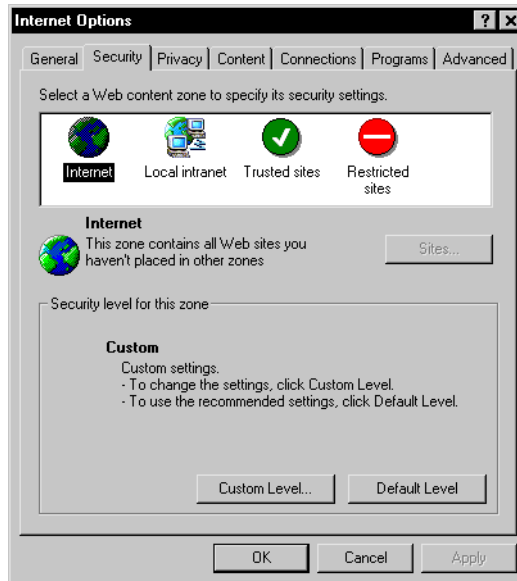
Figure 2: Internet Explorer's Internet Options window

If you are running only on an intranet, skip to step 5. Otherwise, make sure the Internet icon near the top of the window is selected and click on Custom Level to change the security levels for this zone. A window similar to the one shown here appears:
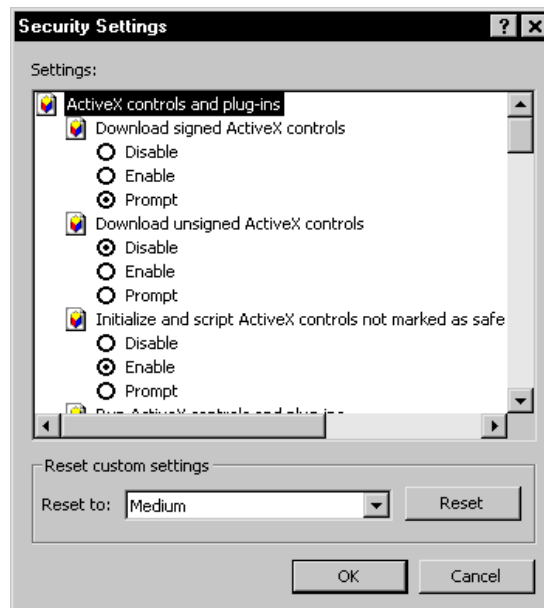
Figure 3: Internet Explorer's Security Settings window

3 Enable these ActiveX control settings in the list:

- Run ActiveX controls and plug-ins

- Script ActiveX controls marked safe for scripting

4   Skip this step if you are only running on the internet, not a local intranet. Otherwise, click on the Local Intranet icon near the top of the window to select it. Then repeat step 4. Once you've enabled the ActiveX control settings. Click Ok to close the Internet Options window.

5   Close your browser and restart it for the changes to take effect.

Setting up a secure web server (SSL/HTTPS)

If you are using SSL for your server connections, there is one setting that can affect WIP Edit. In the Internet Explorer Options window, choose Advanced Tab, Security List and make sure the Do not save encrypted pages to disk option *is not* checked.

If you are using SSL/HTTPS and this item is checked, when you click on the link to download the DPW file the setting prohibits the file from being stored on disk. WIP Edit will try to open the file from the disk location. If it cannot locate the file, this message appears:

Transaction Error (transaction invalid)

If the web site where you access the WIP documents is secured using SSL, you must use *https://* as a prefix in the URL to the web server you use to save data, which is called the *PUTURL*. Here is an example:

<https://www.docucorp.com>

PUTURL is in the global.xml for iDocumaker and IPPS COM. DI 12.0 and 12.3 uses Documaker Administrator to setup PUTURL.12.3 it is no longer required.

When formatting the PUTURL variable, specify the URL the same way you would using a web browser. Here are some examples:

Example	Description
Localhost	
MACHINENAME	Net bios machine name if using an intranet
www.docucorp.com	Normal HTTP
http://www.docucorp.com	Normal HTTP
https://www.docucorp.com	Secure web site using default port 443
https://www.docucorp.com:51000	Secure web site using port 51000
http://somemachine:8080	An intranet with a port specified
http://www.docucorp.com:8080	An intranet with a port specified

Upgrading WIP Edit

To upgrade the WIP Edit plugin, follow the below steps:

1 In the Control Panel, click Programs and Features.

2 Select Wip Edit plugin from the Programs list and remove the older version of the plugin.

3 Install the new version of WIP Edit Plugin, see Installing WIP Edit on page 29 for installation process.

SETTING UP IDS

You must make some modifications to the IDS configuration files request types files, and WIPEDIT.INI) for WIP Edit, iPPS, and IDS to work together.

If this is a new installation the configuration files are written to the appropriate directory. The new request types will be in the configuration files.

If you have an existing installation of IDS or iPPS COM+, then sample configuration files will be written to the installation directories. No existing configuration files will be over-written. You will, however, see configuration files installed that have the word Sample in their name, such as Sample (*REQUEST TYPE INI FILE) and so on. Using a merge tool or text editor, you can copy and paste the new configuration items into an existing installation.

> **NOTE:** Make sure you have the latest version, including patches of IDS and Shared Objects installed before you begin to modify the configuration files.

Most of the changes to the request types are adding new request types that are supported by WIP Edit. Also, you must configure a queuing service such as MQ Series, MSMQ, or the HTTP queues. iPPS does not work without a queue system in place.

A small change to the WIPEDIT.INI file is also required. By making these changes, the back-end server is set to handle requests from WIP Edit.

The following table shows a list of the request types that are required for WIP Edit functionality. A high level description of the function each type performs is also provided.

Request Type	Description
New Request Types	
SPELL*1	Checks spelling using IDS
RETUSERDICT*1	Returns a user dictionary as XML
EDTUSERDICT*1	Edits a user dictionary
i_GetMRLResource*2	Obtains various MRL resource data
i_PluginInit	Initializes WIP Edit and returns a DPW file
i_PluginSave	Saves the DPW to WIP
i_PluginGetResource	Gets a resource and returns it to the client browser
i_Tbllkup*2	Returns entries from standard Documaker table
Modified Request Types	
i_DPRArcWIP	Archives a WIP entry

*1- This request type is only used for the HTML version of iPPS

*2- The i_GetMRLResource and iTbllkup request types are new to version 3.11 of iPPS. They are not required for WIP Edit. However, they are required to get resources real time from IDS.

Request Type	Description
i_ArcWIP	Archives a WIP entry for the Java version
i_DPRModifyWIPData	Modifies and updates the form set data in WIP
i_ModifyWIPData	Modifies and updates the form set data in WIP for the Java version

*1- This request type is only used for the HTML version of iPPS

*2- The i_GetMRLResource and iTbllkup request types are new to version 3.11 of iPPS. They are not required for WIP Edit. However, they are required to get resources real time from IDS.

REQUEST TYPE

If you are installing a new copy of IDS, you do not have to make the following changes. If, however, you have an existing copy of IDS and you are modifying it to supportWIP.Edit, you must add the following request types.

The following listing shows the request types you have to modify to work with WIP Edit. Some of the rules methods in the request types have been deleted and replaced with new entries to accommodate the WIP Edit functionality. The next topic explains what happens in IDS when these rules are executed

```
; Archive WIP Entry

< ReqType:i_DPRArcWIP >

    function = atcw32->ATCLogTransaction

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRGetWipFormset

    function = dprw32->DPRArchiveFormset

; Archive WIP Entry for Java version

[ReqType:i_ArcWIP]

    function = atcw32->ATCLogTransaction

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRGetWipFormset

    function = dprw32->DPRArchiveFormset

; Modify/Update the form set data in WIP

[ReqType:i_DPRModifyWIPData]

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRInitLby

    function = dprw32->DPRGetWipFormset

    function = dprw32->DPRLoadXMLAttachment

    function = dprW32->DPRUpdateFormsetFromXML

    function = dprW32->DPRModifyWipData

; Modify/Update the form set data in WIP for Java Version

[ReqType:i_ModifyWIPData]

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRInitLby

    function = dprw32->DPRGetWipFormset
```

```
function = dprw32->DPRLoadXMLAttachment
function = dprW32->DPRUpdateFormsetFromXML
function = dprW32->DPRModifyWipData
```
Listing 2: Request Types modified for WIP Edit

The following request types file comparison shows the changes you need to make to an existing request types. This example shows the (*REQUEST TYPES INI FILE) file that ships with Shared Objects version 11.0, patch 17 (on the left side) and Shared Objects version 11.0, patch 23 (on the right). The ~~strike through~~ on the left indicates that you delete that rule.

The request type from...	
Shared Objects version 11.0, patch 17	**Shared Objects version 11.0, patch 23**
; Archive WIP Entry and then delete it from wip	; Archive WIP Entry and then delete it from wip
< ReqType:i_ArcWIP >	< ReqType:i_ArcWIP >
function = atcw32->ATCLogTransaction	function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment	function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment	function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig	function = dprw32->DPRSetConfig
function = dprw32->DPRGetWipFormset	function = dprw32->DPRGetWipFormset
function = dprw32->DPRArchiveFormset	function = dprw32->DPRArchiveFormset
function = dprw32->DPRDeleteWipRecord	
; Archive WIP Entry and then delete it from wip ;	; Archive WIP Entry and then delete it from wip
< ReqType:i_DPRArcWIP >	< ReqType:i_DPRArcWIP >
function = atcw32->ATCLogTransaction	function = atcw32->ATCLogTransaction
function = atcw32->ATCLoadAttachment	function = atcw32->ATCLoadAttachment
function = atcw32->ATCUnloadAttachment	function = atcw32->ATCUnloadAttachment
function = dprw32->DPRSetConfig	function = dprw32->DPRSetConfig
function = dprw32->DPRGetWipFormset	function = dprw32->DPRGetWipFormset
function = dprw32->DPRArchiveFormset	function = dprw32->DPRArchiveFormset
function = dprw32-vDPRDeleteWipRecord	
; Modify/Update the formset data in wip ;	; Modify/Update the formset data in wip

The request type from...	
Shared Objects version 11.0, patch 17	**Shared Objects version 11.0, patch 23**
< ReqType:i_DPRModifyWIPData >	< ReqType:i_DPRModifyWIPData >
function = atcw32->ATCLogTransaction	
function = atcw32->ATCLoadAttachment	function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig	
function = atcw32->ATCUnloadAttachment	function = atcw32->ATCUnloadAttachment
function = dprw32->DPRLoadImportFile	function = dprw32->DPRSetConfig
function = dprw32->DPRInitLby	
~~function = dprw32->DPRGetWipFormset~~	
function = dprw32->DPRLoadXMLAttachment	
function = dprw32->DPRUpdateFormsetFromXML	
function = dprw32->DPRModifyWipData	function = dprw32->DPRModifyWipData
; Modify/Update the formset data in wip ;	; Modify/Update the formset data in wip
< ReqType:i_ModifyWIPData >	< ReqType:i_ModifyWIPData >
function = atcw32->ATCLogTransaction	
function = atcw32->ATCLoadAttachment	function = atcw32->ATCLoadAttachment
function = dprw32->DPRSetConfig	
function = atcw32->ATCReceiveFile,XMLIMPORT,IMPORTFILE,*.XML	
function = atcw32->ATCUnloadAttachment	function = atcw32->ATCUnloadAttachment
function = dprw32->DPRLoadImportFile	function = dprw32->DPRSetConfig
; Need to add this rule to existing request types if using DMSudio.	
function = dprw32->DPRInitLby	
function = dprw32->DPRGetWipFormset	
function = dprw32->DPRLoadXMLAttachment	
function = dprw32->DPRUpdateFormsetFromXML	
function = dprw32->DPRModifyWipData	function = dprw32->DPRModifyWipData

The following listing shows the new request types you have to add to the (*REQUEST TYPE INI FILE) file to support the WIP Edit requests:

```
[ReqType:SPELL ]
; Spell check utilizing IDS
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCReceiveFile,XMLIMPORT,IMPORTFILE,*.XML
    function = atcw32->ATCSendFile,DOCUMENTSTREAM,EXPORT,BINARY
    function = dprW32->DPRSpellCheck
[ReqType:RETUSERDICT ]
; Returns a user dictionary as XML
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCSendFile,DOCUMENTSTREAM,RETFILE,BINARY
    function = dprW32->DPRRetFromUserDict
[ReqType:EDTUSERDICT ]
; Edits a user dictionary
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = atcw32->ATCReceiveFile,XMLIMPORT,EDITFILE,*.XML
    function = dprW32->DPREditUserDict
[ ReqType:i_GetMRLResource ]
; Obtains various MRL resource data
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRInitLby
    function = dprw32->DPRGetFormList
[ReqType:i_PluginInit]
; Plugin initialization. Get a DPW file
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = atcw32->ATCSendFile,RF_POSTFILE,PRINTFILE,Binary
    function = dprw32->DPRSetConfig
    function = dprw32->DPRInitLby
```

```
            function = dprw32->DPRGetWipFormset

            function = dprw32->DPRPrintDpw

        [ReqType:i_PluginSave]

        ; Saves the DPW to WIP

            function = atcw32->ATCLoadAttachment

            function = dprw32->DPRSetConfig

            function = dprw32->DPRInitLby

            function = Atcw32-
        >ATCReceiveFile,RF_POSTFILE,RF_POSTFILE,carfile.*

            function = atcw32->ATCUnloadAttachment

            function = dprw32->DPRDpw2Wip

        [ReqType:i_PluginGetResource]

        ; Gets a resource for the Plugin

            function = atcw32->ATCLoadAttachment

            function = atcw32->ATCUnloadAttachment

            function = dprw32->DPRSetConfig

            function = dprw32->DPRInitLby

            function = atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary

            function = dprw32->DPRGetResource,RETURNFILE

        [ReqType:i_Tbllkup]

        ; Returns entries from standard Documaker RP table

            function = atcw32->ATCLoadAttachment

            function = atcw32->ATCUnloadAttachment

            function = dprw32->DPRSetConfig

            function = dprw32->DPRInitLby

            function = dprw32->DPRTblLookUp
```

Listing 3: Request types you must add

REQUESTING A DICTIONARY FROM IDS

You can set up a separate dictionary under IDS for each user ID. WIP Edit determines whether it needs to get another copy of the dictionary. It also updates the spelling dictionary on the server if you make changes.

Follow these steps to configure separate dictionaries for each user:

1   Since IDS must know where the spell dictionaries are kept for each user, you need to define the location in the configuration-specific INI file located in the \Docserv directory (such as AMERGEN.INI). Open this INI file and use the UserDictPath option to establish a directory called *Spell* under \Docserv, as shown here:

```
< Spell >
    UserDictPath = Spell
```

2   Next, add the following to the configuration-specific INI file:

```
< INI2XML >
    CalcCRC = #USERID.tlx!TLX
```

3   This entry lets the WIP Edit get the CRC from IDS and determines whether it needs to download another copy. This example assumes each user's spell dictionary is kept in a separate file based on his or her login user ID. This is the default behavior. The software automatically substitutes the user's ID for *#USERID*.

```
Next, make sure you have the following request types defined

< ReqType:PUTRESOURCE >

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRDecryptLogin

    function = dprw32->DPRDefaultLogin

    function = dprw32->DPRCheckLogin

    function = dprw32->DPRPutResource

    function = atcw32->ATCReceiveFile,RF_POSTFILE,RF_POSTFILE
,.\recv.txt


< ReqType:GETRESOURCE >

    function = atcw32->ATCLogTransaction

    function = atcw32->ATCLoadAttachment

    function = atcw32->ATCUnloadAttachment

    function = dprw32->DPRSetConfig

    function = dprw32->DPRDecryptLogin

    function = dprw32->DPRDefaultLogin

    function = dprw32->DPRCheckLogin

    function = atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary

    function = dprw32->DPRGetResource,RETURNFILE
```

Listing 4: Get and Put Resource Request Types

CUSTOMIZING HOW USER DICTIONARIES ARE STORED

You can create an INI file WIP Edit will use to override the default file name where user spell dictionaries are stored. This example uses an INI file called WIPEDIT.INI. The configuration-specific INI contains these options:

```
< File2DPW >
    INIToken = wipedit.ini
< INI2XML >
    CalcCRC = C:\docserv\Spell\#ORIGUSER.tlx!TLX
```

NOTE: The variable #ORIGUSER can be set to another variable in the WIP index (such as a custom field). See the Documaker Workstation Administration Guide for additional information.

The WIPEDIT.INI file contains this option:

```
< Spell >
    UserDict = ~WIPFIELD ORIGUSER
```

STORING DICTIONARIES ON THE CLIENT

If you want to store the user spell dictionary on a user's local PC instead of the server, remove the CalcCRC option from the INI2XML control group. You may still need the WIPEDIT.INI file to identify the spell dictionary. In this case, however, the user's dictionary will not be available to that user if that user uses a different computer to access the web.

If the spell dictionary is at c:\spell\userspell.tlx, then in the configuration-specific INI file enter:

```
< File2DPW >
    INIToken = wipedit.ini
```

In the WIPEDIT.INI file enter:

```
< Spell >
    UserDict = c:\spell\userspell.tlx
```

SENDING PASSWORDS TO WIP EDIT

If you are using WIP Edit with a web site that requires basic authentication for browsers, you must provide a password and user ID to save a document. You can use the same user ID and password you entered when you logged onto the browser.

To do this, set the following INI options in the configuration-specific INI file.

```
< INI2XML >
    HTTPUserID    = userID
    HTTPPassword  = password
```

The DPRIni2Xml rule must be in the request type that contains the DPRWip2Dpw rule.

```
;Edit record
< ReqType:WEDIT >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRGetWipFormset
    function = dprw32->DPRLockWip
    function = atcw32->ATCSendFile,RF_POSTFILE,RF_POSTFILE,Binary
    function = dprw32->DPRWipIndex2XML
    function = dprw32->DPRFile2Dpw,RF_POSTFILE
    function = dprw32->DPRWip2Dpw,RF_POSTFILE
    function = dprw32->DPRIni2XML
```

Encrypting IDs and passwords

If you do not want the user ID or password viewed in the INI file, use the cryruw32.exe program to encrypt them. The encrypted value can be put into the INI file.

To encrypt both the user ID and password, follow these steps:

1 To encrypt the user ID, from the command line enter:

cyruw32 myuserid

You will get the following output:

Encrypted string (2W9M-z_QGlBZd-5XS4nmB2000)

The parentheses are not part of the encrypted string.

2 To encrypt the password, enter:

cryruw32 password

The program will output:

Encrypted string (2XAUnkxUYlx7i5AnQ4m4E1m00)

3 Make these changes in the configuration-specific INI file:

```
< INI2XML >
    HTTPUserID      = 2W9M-z_QGlBZd-5XS4nmB2000
    HTTPPassword    = 2XAUnkxUYlx7i5AnQ4m4E1m00
```

UNDERSTANDING RULE SETS

The topic, Architecture Overview on page 19, explains how a client request is processed through iPPS and IDS. The key to understanding what is happening in the system is to understand request types and their associated rules.

This topic describes each WIP Edit request type and provides a walk through of what happens on the IDS side when the request and rule sets are processed. For an explanation of the attachment variables and return values for each request type, refer to Setting Global XML Options on page 51.

One thing that will become apparent when looking at the request types/rule sets is that there are patterns to the execution of the requests. Oracle Insurance lets you write your own rule sets that can be folded into the requests. This provides a high degree of flexibility and customization of the server side software to meet changing business needs.

NOTE: Writing custom rules is beyond the scope of this document.

The rest of this topic presents each request type and describes the rules executed when IDS is presented with a client request. Because you can customize IDS by adding your own rules, this topic describes the rule sets defined for the software as it is originally installed.

Request Type	Rule	Description
i_DPRArcWIP		Archives WIP entry.
	atcw32->ATCLogTransaction	Writes transaction information to log file.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32-vATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRGetWipFormset	Retrieves a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files.
	dprw32->DPRArchiveFormset	Sends a form set to DAP archive.
i_ArcWIP		Archives the WIP entry for the Java version.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.

Request Type	Rule	Description
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRGetWipFormset	Retrieves a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files.
	dprw32->DPRArchiveFormset	Sends a form set to DAP archive.
i_DPRModifyWIPData		Modifies and updates the form set data in WIP.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	dprw32->DPRGetWipFormset	Retrieves a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files.
	dprw32->DPRLoadXMLAttachment	Loads the XML attachment into memory.
	dprW32->DPRUpdateFormsetFromXML	Compares the existing and incoming form sets in WIP and synchronizes the changes.
	dprW32->DPRModifyWipData	Updates the form set information when the user changes the form data.
i_ModifyWIPData		Modifies and updates the form set data in WIP for the Java version.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	dprw32->DPRGetWipFormset	Retrieves a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files.

Request Type	Rule	Description
	dprw32->DPRLoadXMLAttachment	Loads the XML attachment into memory.
	dprW32->DPRUpdateFormsetFromXML	Compares the existing and incoming form sets in WIP and synchronizes the changes.
	dprW32->DPRModifyWipData	Updates the form set information when the user changes the form data.
SPELL		Checks spelling using IDS.
	atcw32->ATCLogTransaction	Writes transaction information to log file.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	atcw32->ATCReceiveFile,XMLIMPORT,IMPORTFILE,*.XML	Merges a series of attachment variables into a file and writes that file to disk. Generally, this rule is used to re-assemble a file that has been posted in segments to an IDS queue by the ATCSendFile rule. The file that is received can be either a binary or text file.
	atcw32->ATCSendFile,DOCUMENTSTREAM,EXPORT,BINARY	Posts a file in segments to the output attachment and sends it over the IDS queue. The ATCReceiveFile rule or the DSIReceiveFile API can then re-assemble the file from the input attachment and save it. The file can be binary or text.
	dprW32->DPRSpellCheck	Initializes the spell check.
RETUSERDICT		Returns a user dictionary as XML.
	atcw32->ATCLogTransaction	Writes transaction information to log file.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.

Request Type	Rule	Description
	atcw32->ATCSendFile,DOCUMENTSTREAM, RETFILE,BINARY	Posts a file in segments to the output attachment and sends it over the IDS queue. The ATCReceiveFile rule or the DSIReceiveFile API can then re-assemble the file from the input attachment and save it. The file can be binary or text.
	dprW32->DPRRetFromUserDict	Returns the user dictionary.
EDTUSERDICT		Edits a user dictionary.
	atcw32->ATCLogTransaction	Writes transaction information to a log file.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	atcw32->ATCReceiveFile,XMLIMPORT,EDITFILE, *.XML	Merges a series of attachment variables into a file and writes that file to disk. Generally, this rule is used to re-assemble a file that has been posted in segments to an IDS queue by the ATCSendFile rule. The file that is received can be either a binary or text file.
	dprW32->DPREditUserDict	Posts the changes for the user dictionary to the back end server.
i_GetMRLResource		Gets MRL resource data.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	dprw32->DPRGetFormList	Returns a set of information that is based on the XML sent as an input variable.
i_PluginInit		Initializes the plug-in and gets a DPW file.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.

Request Type	Rule	Description
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	atcw32->ATCSendFile,RF_POSTFILE,PRINTFILE, Binary	Posts a file in segments to the output attachment and sends it over the IDS queue. The ATCReceiveFile rule or the DSIReceiveFile API can then re-assemble the file from the input attachment and save it. The file can be binary or text.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	dprw32->DPRGetWipFormset	Retrieves a form set from the WIP record. If the record exists, it loads the WIP form set by loading POL and NA files.
	dprw32->DPRPrintDpw	Writes the WIP record to a DPW file and sends the results back to the caller.
i_PluginSave		Saves the DPW to WIP.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	Atcw32->ATCReceiveFile, RF_POSTFILE,RF_POSTFILE, C:\docserv\data\carfile.*	Merges a series of attachment variables into a file and writes that file to disk. Generally, this rule is used to re-assemble a file that has been posted in segments to an IDS queue by the ATCSendFile rule. The file that is received can be either a binary or text file.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRDpw2Wip	WIP Edit uses this to save to WIP.
i_PluginGetResource		Gets a resource for the plug-in.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.

Request Type	Rule	Description
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary	Posts a file in segments to the output attachment and sends it over the IDS queue. The DSIReceiveFile API can then re-assemble the file from the input attachment and save it. The file can be binary or text.
	dprw32->DPRGetResource,RETURNFILE	Returns an item from IDS. The result depends on the input variable request.
i_Tbllkup		Returns entries from a standard Documaker table.
	atcw32->ATCLoadAttachment	Parses the input attachment variables from the input queue and stores the data in an internal data structure.
	atcw32->ATCUnloadAttachment	Converts the attachment variables from internal format into the queue attachment format in the output queue.
	dprw32->DPRSetConfig	Sets the current INI file context based on the CONFIG value. The CONFIG value is passed from the client in the attachment. If this value does not exist, the rule does nothing and returns.
	dprw32->DPRInitLby	Initializes Library Manager.
	dprw32->DPRTblLookUp	Returns the name value pairs from IDS that are stored in the Workstation table data files.

SETTING UP iPPS

Follow these instructions to set up iPPS COM+.

iPPS COM+ SOFTWARE PREREQUISITES

The COM+ version of iPPS operates on the Microsoft Windows Server platform. As such, there are certain software components available from Microsoft you must install before you install the Oracle Insurance product. You can get the following software at...

www.microsoft.com

- Windows Installer 2.0 or higher

- Data Access Components 2.6 or higher

- MSXML Parser 4 or 6 (see note below)

NOTE: iPPS versions 3.1 and 3.11 work with MSXML Parser 4 Service Pack 2 or higher.

iPPS versions 3.12 and higher require MSXML Parser 6.0 Service Pack 1 or higher.

Be aware that Microsoft has announced plans to end support for MSXML Parser 4 on 12/31/2008. Oracle Insurance recommends that you move to version 3.12 at your convenience. Moving to version 3.12 consists of upgrading to iPPS COM version 3.12 base objects and modifying your custom ASP pages to start using MSXML Parser 6.

For more information on Microsoft XML Parser (MSXML) versions, see Microsoft's Help and Support web site (http://support.microsoft.com/).

Once the prerequisite software has been installed, you can install iPPS. There is a specific installation order that must be followed:

1 Make sure Docupresentment (IDS) version 1.8 and the Documaker Bridge are installed first on your back-end server (or on this server if you want IDS and iPPS to reside on the same computer).

2 iPPS 3.1x Web Resources COM+

3 iPPS 3.1x COM+

4 (Optional) The Amergen sample MRL. This can be used to test the system.

iPPS COM+ ON MICROSOFT WINDOWS SERVER 2003

Microsoft made changes to the default settings on IIS between Windows 2000 Server and Windows 2003 Server. With Windows 2000 Server, almost everything was turned on. They went in the opposite direction for Windows 2003 Server.

If you install iPPS COM+ on Windows 2003 Server, there are three things that you must do to get the system to work:

1 In Control Panel, Administrative Tools, Component Services, Computers, My Computer, COM+ Applications, Docucorp – iPPS311, right click the Docucorp item and select Properties from the menu. Then select the Security tab and uncheck the Enforce access checks for this application field. Click Ok to save the change.

2 In the Control Panel, Administrative Tools, IIS Manager select the Default Web Site (or the one you selected during your installation). Right click on this and open the properties menu. Go to the Home Directory tab and click Configuration.

3 Select the Options tab and check the Enable Parent Paths field. Click Ok. Then click Ok again to save the changes.

4 While still in the IIS Manager window, select the Web Service Extensions field. On the right side, select Active Server Pages and click Allow. Then select Server Side Includes and click Allow again.

5 Restart the web server to make sure the changes take effect.

SETTING GLOBAL XML OPTIONS

Both the COM+ and Java versions of iPPS ship with a GLOBAL.XML file that controls the configuration of the iPPS application. The system administrator modifies this file to establish settings for the server environment. This topic describes the structure of this file to clarify some aspects of what is required to properly configure the iPPS software.

Keep in mind that all entries in the GLOBAL.XML file are case sensitive. While discussing various settings in the text portion of this document, an entry may be referred to in mixed case for easier reading. However, the examples shown in the listings will have the correct case utilization for the GLOBAL.XML file.

Global XML File Structure

As shipped from Oracle Insurance, the Global XML has two main sections as shown here:

```
<?xml version="1.0" ?>

<XMLINI TYPE="IPPS" VERSION="1.0">

<CONFIG NAME="SYSTEM">

<CONFIG NAME="AMERGEN" LIBRARY="Amergen Insurance">

</XMLINI>
```

Listing 5: GLOBAL.XML Basic Structure

The root element name is *XMLINI*. Within that structure there are configuration elements, called *CONFIG*. The first configuration is labeled *System*. This configuration is a global type of setting that all other configurations are derived from. All of the elements that appear under the System configuration are available to any other configuration. However, each configuration (excluding the System configuration) is independent of any other configuration. So your individual configuration can access the System configuration, but not the Amergen configuration.

When you create an MRL for your application, you will add an element to the GLOBAL.XML file for that configuration. In general, you can copy the Amergen configuration and edit it to reflect the settings for your MRL.

System Element

Listing 6 shows the basic elements that are under the System configuration. The Encoding element establishes the encoding mechanism for the iPPS system. In this case, the default is the ISO-8859-1 standard.

The ApplicationData element contains settings that affect the web server iPPS uses. The DBASE element contains the user ID and password information, along with other system connection information, for using a database for storing the Users database. The ReqTypes element holds all of the request types for the iPPS system. Specific requests can also be defined within a configuration. For example, the Amergen configuration can have custom request types that are not available to other configurations.

The Services element applies to the Java version of iPPS. For each service, this setting lets the system know which of the presentation layers should have XSL templates applied for the default system settings.

The Logging element applies to the Java version of iPPS only and controls whether the system is set in debug mode. If debugging is turned on, then this defines where the debug log files are written and stored on the server.

The Mail element defines parameters for your mail server. If this is set up, then when a user completes a transaction, they can mail a form to someone else.

When configuring iPPS, the system administrator has to edit the GLOBAL.XML file to set various parameters. In general, anywhere a variable is shown with a dollar sign ($) in front of the name, the administrator must edit the value with information for his particular system.

```
<?xml version="1.0" ?>

<XMLINI TYPE="IPPS" VERSION="1.0">

<CONFIG NAME="SYSTEM">

<ENCODING>ISO-8859-1</ENCODING>

<APPLICATIONDATA>

<DBASE>

<REQTYPES>

<SERVICES>

<LOGGING NAME="" LEVEL="DEBUG"/>

<MAIL TYPE="SMTP" HOST="$EMAILHOST" DOMAIN="$EMAILDOMAIN"
PASSWORD=""  USERID="" PORT="25" />

</CONFIG>

<CONFIG NAME="AMERGEN" LIBRARY="Amergen Insurance">

</XMLINI>
```

Listing 6: Elements under *System*

Listing 7 shows the XML file with the Application Data element expanded. This element contains information about the web server location and how the administrator wants to do session management.

```
<?xml version="1.0" ?>
```

```
<XMLINI TYPE="IPPS" VERSION="1.0">
<CONFIG NAME="SYSTEM">
<ENCODING>ISO-8859-1</ENCODING>
<APPLICATIONDATA>
<ACTIVESERVER PORT="80" HOST="$HOST" DOMAIN="$DOMAIN"
ROOT="$IPPSROOT"  TRANSLATED_PATH="$WWWPATH">
<STATEMGT DOMAIN="$DOMAIN" HOST="$HOST" ROOT="$ROOT" PORT="21"
PASSWORD="guest" USERID="anonymous">$STATEMGT</STATEMGT>
</APPLICATIONDATA>
<DBASE>
<REQTYPES>
<SERVICES>
<LOGGING NAME="" LEVEL="DEBUG"/>
<MAIL TYPE="SMTP" HOST="$EMAILHOST" DOMAIN="$EMAILDOMAIN"
PASSWORD=""  USERID="" PORT="25" />
</CONFIG>
<CONFIG NAME="AMERGEN" LIBRARY="Amergen Insurance">
</XMLINI>
```

Listing 7: Application Data section information

Listing 8 shows an Application Data element section configured for an iPPS COM+ server. It illustrates the values that an administrator puts into the GLOBAL.XML file.

One key setting is the Translated_Path. On a Microsoft Windows server running iPPS COM+, this points to the location of the ASP directory under this directory:

C:\InetPub\wwwroot_iPPS311 *(Adjusted for your installation)*

For iPPSj (the Java implementation), Translated_Path always points to the location of your web application's WEB-INF directory.

```
<APPLICATIONDATA>

<ACTIVESERVER PORT="80" HOST="www" DOMAIN="http://localhost"
ROOT="iPPS31"  TRANSLATED_PATH="c:\inetpub\wwwroot\_iPPS311\asp">

<STATEMGT DOMAIN="http://localhost" HOST="www" ROOT="iPPS31"
PORT="21"  PASSWORD="guest"
USERID="anonymous">c:\inetpub\wwwroot\_iPPS311\import</STATEMGT>

<!--DESTINATION = MEMORY | FILESYSTEM

    TYPES OF CACHE:

        0.  DATABASE QUERIES (NA)

        1.  DBF2XML

        2.  XSLT_TRANSFORMS (NA)

        3.  ENTRY_FORMS

        4.  IMAGES (JPG, GIF) (NA)

        5.  NOT USED

        6.  NOT USED

        7.  TOOLBAR (NA)

        8.  TEMPLATES (NA)

        9.  GROUPS

        10. FORMS LIST

        11. NOT USED

        12. FORMSET SHELL

            CACHE BIT SETTING

        =================================

            0001000000000 = ENTRY_FORMS CACHE  ACTIVE

-->

<OBJECTCACHE ACTIVE="YES" EXPIRETIME="" MAXSIZE=""
TYPE="0001000000000"  DESTINATION="" SERIALIZE=""/>

</ACTIVESERVER>

</APPLICATIONDATA>
```

Listing 8: Sample Application Data section

Listing 9 shows a DBASE section for a COM+ installation. In this instance, the server is using a Microsoft Access database file for the Users database. The driver and file location are identified in the settings. On UNIX, you must use some other database since Access is not available.

```
<?xml version="1.0" ?>

<XMLINI TYPE="IPPS" VERSION="1.0">

<CONFIG NAME="SYSTEM">

<APPLICATIONDATA>

<DBASE>

<TABLENAME="iUserDB">

<DSN USERID="" PASSWORD="" CLASS="">driver={Microsoft Access Driver
(*.mdb)};dbq=c:\inetpub\wwwroot\_iPPS311\user\USER.mdb;uid=Admin</
DSN>

</TABLE>

</DBASE>

<REQTYPES>

<SERVICES>

</CONFIG>

<CONFIG NAME="AMERGEN" LIBRARY="Amergen Insurance">

</XMLINI>
```

Listing 9: A sample DBASE section from a COM+ installation

Listing 10 shows a typical request type from the GLOBAL.XML file. All request types follow a general structure of elements. The first element is a ReqType Description. This identifies the name of the request type and associates an internal number to it (Low.Level.Service). For each request type in the GLOBAL.XML file, there will be a corresponding request type on the IDS side. The names will match (even in capitalization).

```
<REQTYPE DESCRIPTION="" NAME="i_Print" LOW.LEVEL.SERVICE="5180">

<INPUT>

  <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"  DATAXPATH="DOCUMENT/
SESSION/USERID" />

  <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"  DATAXPATH="DOCUMENT/
DOCSET/LIBRARY/@CONFIG" />

  <VAR NAME="IMPORTFILE" SOURCE="" SENDVIAQUEUE="NO" />

  <VAR NAME="FILETYPE" SOURCE="" XPATHSOURCE="2"
DATAXPATH="PUBLISHEXPORT" />

  <VAR NAME="PRINTFILE" SOURCE="" RETURNVIAQUEUE="NO" />

  <VAR NAME="PRTTYPE" SOURCE="" />

  <VAR NAME="RECIPIENT" SOURCE="" />

  <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>

  </INPUT>

- <OUTPUT>

  <VAR NAME="RESULTS" />

  <VAR NAME="PRINTFILE" />
```

```
                    </OUTPUT>

                    </REQTYPE>
```

Listing 10: Sample Request Type Element

Following the request type description is the input parameter section. This consists of one or more elements that begin with *VAR*. These are input values for the request type. When the rule set is run against the request type, each rule has access to these input values.

The Output element defines what is sent back in the COM+ version of iPPS. In comparison, the Java version of iPPS returns all output, not just the specific output listed in these elements.

For each request type, the ReqType Description element is repeated. All of the individual MRLs defined in the GLOBAL.XML file have access to the request types listed in the System section. Custom request types are written for specific MRLs and are contained in their own section as we'll see in the next topic.

MRL section   Oracle Insurance ships a sample MRL for use with iPPS called *Amergen*. You can use the Amergen sections of the GLOBAL.XML file as a template when you set up your own MRLs. The elements in the Amergen section of the GLOBAL.XML file are customized for that specific application. None of these settings are visible to the other MRLs you define in the GLOBAL.XML file.

Listing 11 shows the high level view of the Amergen section. The items in this portion of the GLOBAL.XML file are specific to the MRL.

Allcaps tells the system to convert data entry text values to upper case. MaxSelectList establishes the maximum number of items that will be displayed in a list. For example, in the WIP list or Archive list, the setting shown below would only show up to 10 items in the list. You can adjust this value if you want a different number of items to appear. PublishExport deals with the format of exported items. By default, it is in XML format.

```
<CONFIG NAME= "AMERGEN" LIBRARY="Amergen Insurance">

<ALLCAPS>true</ALLCAPS>

<MAXSELECTLIST>10</MAXSELECTLIST>

<PUBLISHEXPORT>XML</PUBLISHEXPORT>

<PPS>

<APPLICATIONDATA>

<DBASE>

<MESSAGEQUEUE>

<REQTYPES>

<SERVICES>

<LOOkUP>

</CONFIG>
```

Listing 11: Amergen configuration settings

The PPS element establishes values for the iPPS system. These options are not covered here but are documented in the PPS documentation. Essentially, they let you customize some of the PPS system level behavior.

Listing 12 shows the Amergen ApplicationData section of the GLOBAL.XML file. The ActiveServer settings re-state the web server settings. This information has to be the same as the ActiveServer information in the System section of the GLOBAL.XML file.

```
- <APPLICATIONDATA>

- <ACTIVESERVER PORT="80" HOST="$HOST" DOMAIN="$DOMAIN"
ROOT="$IPPSROOT">

  <IDSIMPORT>$IDSIMP</IDSIMPORT>

  <IDSSPOOL>$IDSSPL</IDSSPOOL>

  <IIMPORT DOMAIN="$DOMAIN" HOST="$HOST" ROOT="$ROOT" PORT="21"
PASSWORD="guest" USERID="anonymous">$IPPSIMPPATH</IIMPORT>

  <ISPOOL DOMAIN="$DOMAIN" HOST="$HOST" ROOT="$ROOT" PORT="21"
PASSWORD="guest" USERID="anonymous">$IPPSSPLPATH</ISPOOL>

  <CACHE ROOT="$CACHEPATH" CLEARONSTARTUP="TRUE" ACTIVE="TRUE" />

- <!--

Use this node if define HTML location if not using MRL database

            PROTOCOL Attribute valid values:

            DBMS, IDS, URL, URLMULTIPART, PLUGIN

  -->

- <!-- <IFORMS PROTOCOL="URL" DOMAIN="$DOMAIN" HOST="$HOST"
ROOT="$ROOT" PORT="21" PASSWORD="guest"
USERID="anonymous">$FORMSLOC</IFORMS>

  -->

  </ACTIVESERVER>

</ APPLICATIONDATA>
```

Listing 12: Amergen Application Data section of the GLOBAL.XML file

The IDSImport and IDSSpool elements identify directories where the system gets and puts files for the IDS system. For HTML-based PPS systems, the Cache Root establishes the directory where the HTML files are cached. For this to work, set the Active setting to True. The ClearOnStartup setting flushes the cache when the system starts.

The IFORMS PROTOCOL setting is commented out in Listing 12. However, that section will be added to support the WIP Edit ActiveX control. Specific changes to support WIP Edit are discussed in Modifying the GLOBAL.XML File on page 115.

Listing 13 shows the Amergen DBASE section of the GLOBAL.XML file. The section tells where the MRL database is stored. It also specifies the location of any lookup tables used for the MRL. This tells you who the provider is for your MRL information. You can choose either a database or IDS. If the value is set to...

MRLSOURCE = IDS

resources are pulled form IDS in real time.

```
<DBASE>
<TABLE NAME="iMRLDB">
<!--
 MRLSOURCE attribute defines the location for MRL resources
            MRLSOURCE Attribute valid values:
            DBMS, IDS
 -->
 <DSN USERID="" PASSWORD="" CLASS="" MRLSOURCE="DBMS">$MRLDB</DSN>
 </TABLE>
 <!--  Example of table lookup entry. Only used if applicable
 -->
 <TABLE NAME="LOOkUP">
 <DSN USERID="" PASSWORD="" CLASS="">$LOOkUPDB</DSN>
 </TABLE>

 </TABLE>
 </DBASE>
```

Listing 13: Amergen DBASE section of the GLOBAL.XML file

Listing 14 shows a typical queuing service configuration for an MRL. These settings are only used by the Java version of iPPS. The COM+ version uses the disco.dll, which is solely dependent on the configuration specified in the DSI.INI file on the back end server.

```
<MESSAGEQUEUE>
 <QUEUEMANAGER NAME="QM">
  <REQUESTQ WAITSECONDS="30">REQUESTQ</REQUESTQ>
  <REPLYQ>RESULTQ</REPLYQ>
  <MODELQUEUE>SYSTEM.DEFAULT.MODEL.QUEUE</MODELQUEUE>
  <CHANNEL>SYSTEM.DEF.SVRCONN</CHANNEL>
  <PORT>1414</PORT>
  <HOST>127.0.0.1</HOST>
  <TRANSPORT>MQSeries Client</TRANSPORT>
  <EXPIRYTIME>1800</EXPIRYTIME>
  <CHARACTERSET>819</CHARACTERSET>
 <QUEUEFACTORY> DSIMQMessageQueueFactory</QUEUEFACTORY>
  </QUEUEMANAGER>
  </MESSAGEQUEUE>
```

Listing 14: Queue Configuration for Amergen

In Listing 14, the Amergen MRL is using MQ Series for the queue service. The system administrator configures items to identify things like the queue manager, the IP address of the machine running MQ Series, the port used, the default channel, and so on.

To use WIP Edit, you must have a queue configured. You have a choice between MQ Series (or WebSphere MQ), Microsoft Queue Manager (MSMQ) or HTTP queues.

The GLOBAL.XML file also has a ReqType section in the MRL section of the file. This is where you any custom request types that you have created. Custom request types is discussed in the IPPS manual.

Modifying the GLOBAL.XML File on page 115 provides detailed instructions on modifying the GLOBAL.XML file for use with WIP Edit.

Request Types and Attachment Variables

The iPPS server is responsible for getting the client input via HTTP Gets/Puts and Submits and translating that to a message that can trigger an action on IDS. To process requests, iPPS must know how to assemble and disassemble messages used by IDS. The iPPS server deals with request types and attachment variables. The attachment variables consist of input variables and results.

This table shows you the request types and the attachment variables associated with the request types. This information is stored in the GLOBAL.XML file.

Request type	Direction	Attachment variable	Description
i_Print			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		IMPORTFILE	The input attachment for the request type
		FILETYPE	Specifies the IMPORTFILE type
		PRINTFILE	Specifies the return file with the absolute name
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		RECIPIENT	Specifies the recipient for a published document
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		PRINTFILE	Specifies the return file with the absolute name
i_PrintFomrsetXML			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		IMPORTFILE	The input attachment for the request type
		FILETYPE	Specifies the IMPORTFILE type
		PRINTFILE	Specifies the return file with the absolute name
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		XMLALLFIELDS	All of the section (image) level, form level, and global level fields are defined per transaction
		ALLRECIPIENTS	All the defined recipients per form set
		DSITIMEOUT	Timeout interval for the request

Request type	Direction	Attachment variable	Description
	Out	RESULTS	Returns SUCCESS or FAILURE
		PRINTFILE	Specifies the return file with the absolute name
i_Proof			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		IMPORTFILE	The input attachment for the request type
		FILETYPE	Specifies the IMPORTFILE type
		PRINTFILE	Specifies the return file with the absolute name
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		RECIPIENT	Specifies the recipient for a published document
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		PRINTFILE	Specifies the return file with the absolute name
i_CheckPolicy			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		FIELDS	Searchable fields from an DFD index
		<Key1>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key1 is mapped to this name.
		<KeyID>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, KeyID is mapped to this name.
		CASESENSITIVE	Specifies whether the search is case sensitive
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRAddWIP			
	In	USERID	The user's login ID

Request type	Direction	Attachment variable	Description
		CONFIG	Defines the MRL resources that are available
		IMPORTFILE	The input attachment for the request type
		<Key1>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key1 is mapped to this name.
		<Key2>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key2 is mapped to this name.
		<KeyID>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, KeyID is mapped to this name.
		<TRANCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, TranCode is mapped to this name.
		<STATUSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, StatusCode is mapped to this name.
		CURRUSER	Specifies the current user based on the XML session
		DESC	The user's description entry
		ASSIGNDESC	Same as above
		ASSIGNUSERID	Targeted user for the transaction defined in the UserInfo database.
		UNIQUE	Unique identifier for this transaction
		FILETYPE	Specifies the IMPORTFILE type
		DSITIMEOUT	Timeout interval for the request
	Out	None.	
i_DPRModifyWIPData			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RECNUM	Unique identifier for a transaction
		IMPORTFILE	The input attachment for the request type
		FILETYPE	Specifies the IMPORTFILE type
		DSITIMEOUT	Timeout interval for the request

Request type	Direction	Attachment variable	Description
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRArcWIP			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		<ArchiveKeyNodes>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, ArchiveKeyNodes is mapped to this name.
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRAssignWIP			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RECNUM	Unique identifier for a transaction
		ASSIGNUSERID	Targeted user for the transaction defined in the UserInfo database.
		ASSIGNDESC	The user's description entry
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRFindWIPRecord			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		STARTRECORD	Starting record for the search
		MAXRECORDS	Max number of records to return in the result set
		CURRUSER	Specifies the current user based on the XML session
		<STATUSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, StatusCode is mapped to this name.
		<Key1>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key1 is mapped to this name.

Request type	Direction	Attachment variable	Description
		<Key2>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key2 is mapped to this name.
		<KeyID>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, KeyID is mapped to this name.
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		RECORDS	Returns the search results
i_DPRUpdateWIP			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		WIPS	Key for the update
		WIPS1.RECNUM	The specific index field based on the DFD to change
		GOCHANGE	Value specifies that the change should occur
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRDeleteWIP			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RECNUM	Unique identifier for a transaction
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRGetWIPEntry			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RECNUM	Unique identifier for a transaction
		EXPORT	The file to export

Request type	Direction	Attachment variable	Description
		FILETYPE	Specifies the export file type
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRWIPSearch			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		STARTRECORD	Starting record for the search
		MAXRECORDS	Max number of records to return in the result set
		<STATUSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, StatusCode is mapped to this name.
		CURRUSER	Specifies the current user based on the XML session
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		RECORDS	Returns the search results
i_PrintWIPFormset			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RECNUM	Unique identifier for a transaction
		PRINTFILE	Specifies the return file with the absolute name
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		ALLRECIPIENTS	All the defined recipients per form set
		DPRPROOFLOGO	Add a proof logo to the print output
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_ArcRetrieve			
	In	USERID	The user's login ID

Request type	Direction	Attachment variable	Description
		CONFIG	Defines the MRL resources that are available
		<ARCKEY>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, ArcKey is mapped to this name.
		<KEYID>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, KeyID is mapped to this name.
		<TRANSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, TransCode is mapped to this name.
		<STATUSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, StatusCode is mapped to this name.
		DESC	The user's description entry
		EXPORT	The file to export
		FILETYPE	Specifies the export file type
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_ArchSearch			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		FIELDS	The fields to search
		MAXRECORDS	Max number of records to return
		MAXSELECTLIST	Max number of records to include in the return
		PARTIALMATCH	Specifies that partial matches are permitted in the search.
		CASESENSITIVE	Whether the search is case sensitive
		<ARCKEY>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, ArcKey is mapped to this name.
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		RECORDS	The record set returned

Request type	Direction	Attachment variable	Description
		LASTRECORD	Record indicator for the last record returned
		MAXRECORDS	Max number of records to return in the result set
		MORERECORDS	Indicates that there are more records that match the search criteria
i_ArcRecip			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		<ARCKEY>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, ArcKey is mapped to this name.
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		RECORDS	The record set returned
i_ArcPrint			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		<ARCKEY>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, ArcKey is mapped to this name.
		RECIPIENT	Specifies the recipient for a published document
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		PRINTPATH	A fully qualified path to the print file
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		REMOTEPRINTFILE	The name of the output file
		SENDBACKPAGE	
i_ArchiveFormset			
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available

Request type	Direction	Attachment variable	Description
		IMPORTFILE	The input attachment for the request type
		FILETYPE	Specifies the IMPORTFILE type
		CREATETIME	
		<Key1>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key1 is mapped to this name.
		<Key2>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, Key2 is mapped to this name.
		<KeyID>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, KeyID is mapped to this name.
		<STATUSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, StatusCode is mapped to this name.
		<TRANSCODE>	The Name attribute is blank, which lets the customer specify a custom name setting in the IDS configuration file. Internally, TransCode is mapped to this name.
	Out	RESULTS	Returns SUCCESS or FAILURE
i_GetMRLResource			Returns groups from IDS
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		DOCUMENTSTREAM	The returned attachment
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
SPELL			Performs a spell check
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		IMPORTFILE	The print file for the request type
		EXPORTFILE	The print file for the output
		USERDICT	Defines which user dictionary to use
		LANGUAGEOPT	Language used in the dictionary

Request type	Direction	Attachment variable	Description
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
RETUSERDICT			Returns the user dictionary
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		EDITFILE	The user's file with changes
		USERDICT	Defines which user dictionary to use
		LANGUAGEOPT	Language used in the dictionary
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
EDTUSERDICT			Edits the user dictionary
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		EDITFILE	The file that will be spell checked
		USERDICT	Defines which user dictionary to use
		LANGUAGEOPT	Language used in the dictionary
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_Tbllkup			Pulls data from IDS for tables
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		TABLEFILE	The table lookup file
		TABLEID	The ID within the table file
		TABLERETURNS	Returns for a particular table ID
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE

Request type	Direction	Attachment variable	Description
		RECORDS	The record set returned
i_PluginInit			
	In	USERID	The user's login ID
		PASSWORD	The user password
		CONFIG	Defines the MRL resources that are available
		PRTTYPE	The type of file to print (PDF, XML, and so on)
		RF_POSTFILE	Filename for the DPW file
		RECNUM	Unique identifier for the item in WIP
		HTTPQUERYSTRING	Defaults to 1
		HTTPQUERYSTRING1.NAME	Used to pass any number of query string variables to pass through WIP Edit. By default the system sets the Session ID internally.
		HTTPQUERYSTRING1.VALUE	Session information
		SAVE_REQTYPE	i_PluginSave (Request name defined in the REQTYPE section for saving plug-in data to WIP.
		SCRIPT	$IPPSROOT/wipsave.asp
		GETSCRIPT	$IPPSROOT/wipdownload.asp
		PUTURL	The URL to the web server used to save data
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_PluginSave			Saves Plug-in data back to WIP
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RF_POSTFILE	DPW file used for saving
		DPWRECNUM	The unique identifier for the WIP transaction
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE

Request type	Direction	Attachment variable	Description
i_PluginGetResource			Retrieves resource from IDS
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		RETURNFILE	The resource requested
		RESOURCENAME	Name of the resource to retrieve
		RESOURCETYPE	One of the following: DAL = DAL script FAP = FAP file TLX = dictionary for spell check TBL = table HLP = help file
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
i_DPRFindWIPRecord			This request type performs a WIP filter
	In	USERID	The user's login ID
		CONFIG	Defines the MRL resources that are available
		STARTRECORD	Starting record number
		MAXRECORDS	Max number of records to search
		DSITIMEOUT	Timeout interval for the request
	Out	RESULTS	Returns SUCCESS or FAILURE
		MORERECORDS	Indicates if more records are available
		RECORDS	The record set returned

STARTING THE SYSTEM

You have now completed the installation and initial configuration necessary to have iPPS work with WIP Edit. The next step is to start your system and test it. Follow these instructions:

NOTE: For Windows, you can install the WIP Edit plug-in on the iPPS server for testing purposes.

1 Go to the docserv directory and start IDS. On Windows, you enter this command:

docserver.bat

Current version of IDS, is started with docserver.bat on windows or docserver.sh on unix. This is applicable for IDS 2.0 and greater.

2 If you have installed iPPS on Windows along with WIP Edit, start your browser and enter

http://localhost/ipps311

If you are using the Amergen MRL, you will see the login window. You can use this ID and password to log in:

For	Enter
Login ID	**demo1**
Password	**demo1**

You are now finished with the initial installation and configuration. The next step is to customize your system and the resources to meet your business needs.

USING THE WIP EDIT ACTIVEX CONTROL

ADDITIONAL CLIENT SIDE INFORMATION

DPW Files

When the user requests a form, the information necessary to construct that form such as sections, graphics, tables, and so on, is placed into a file with a *DPW* file extension and sent to the client browser.

NOTE: *DPW* is a proprietary Oracle Insurance file format.

The system prompts the user to open or save the DPW file. If the user has WIP Edit installed and opens the file, a WYSIWYG image of the form appears, as shown below. The objects that make up the DPW file are unpacked on the user's hard drive in the location pointed to by the TEMP environment variable. By default, TEMP points to a directory similar to the one shown here:

C:\Documents and Settings\user name\local settings\temp
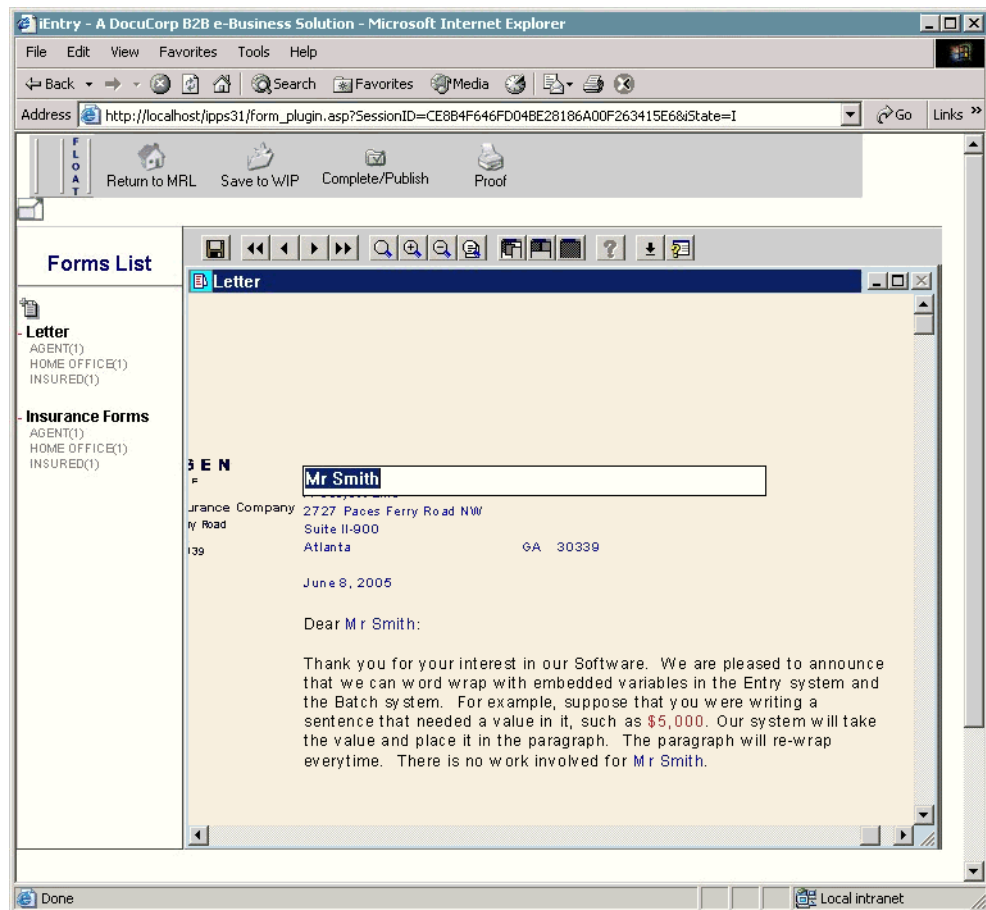
Figure 4: Requested Form Displayed in Browser
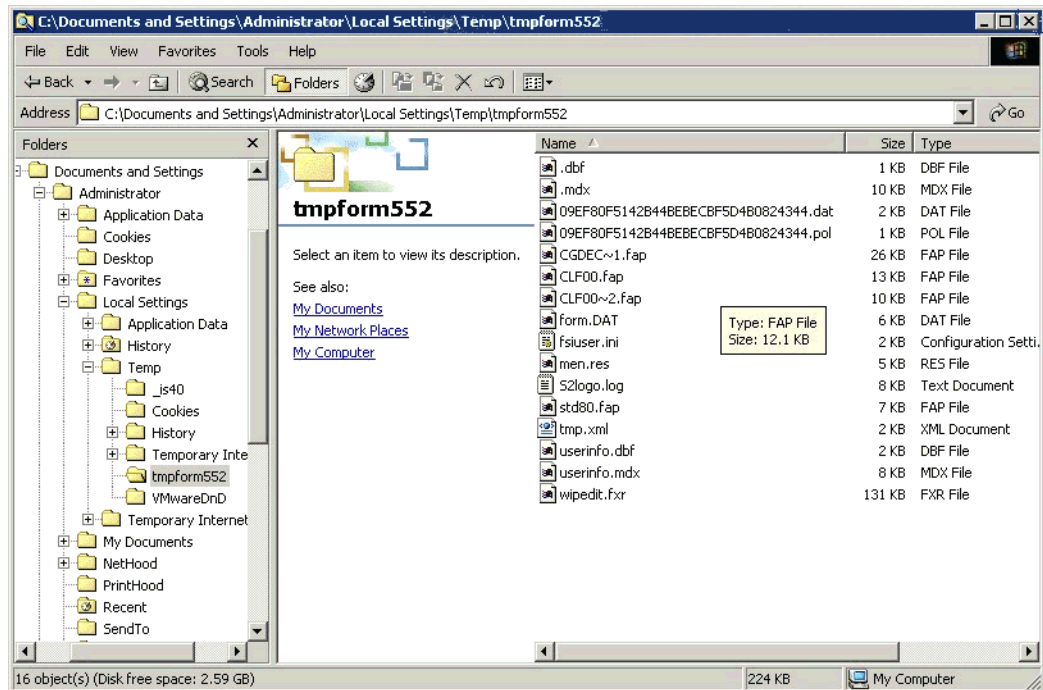
Figure 5: DPW File Expanded into a Temp Directory

The DPW file is extracted under the temp directory into a folder named *tmpformPID*, where *PID* is the process ID. As seen in Figure 5, the tmpform directory contains the files necessary to build the form image. The directory contains the DAT and POL files, several FAP files, graphics files (*.log), and the MEN.RES file. It also has a local copy of the FSIUSER.INI file and the USERINFO.DBF and USERINFO.MDX database files.

NOTE: If you are having trouble receiving files, check for a temp directory like this and see if the files are being downloaded to the client machine. If you cannot locate this directory on the client, check the settings in Internet Explorer's Tools, Folder Options window and make sure the Show Hidden files and directories option is selected.

When the user exits the form currently displayed in the browser, the tmpform directory is erased from the client side hard drive.

Checking Spelling

Once a document is displayed on the client side, the user can run a spell check against it. Oracle Insurance supplies a common dictionary that resides on the server side (IDS). In addition, each user can have a custom dictionary. This dictionary can be stored on the user's local machine or it can reside on the server. The storage location is define using an INI option on the server side. The dictionary files have an extension of *.TLX (an index file) and *.CLX (the data file). Refer to Requesting a Dictionary from IDS on page 41 for specific information on how to set up local and server side dictionary storage.

Locally stored dictionary

A user can have a local copy of the dictionary. For local storage, however, it is implementation specific on how a dictionary is obtained. Locally stored dictionaries are not automatically synchronized with server side dictionaries. Any change the user makes to that dictionary only affects the local copy.

Server side dictionary

Alternatively, each iPPS user can have a dictionary stored on the server. In this case, when the user logs onto the system, his user ID is captured as part of the session information. When a spell check is run, the spell checker acquires an existing dictionary with that user's ID or creates one if no dictionary with that user ID exists. A copy of this dictionary is downloaded to the client machine. For the duration of the session, spell checks will use this local copy. If the user changes the dictionary, the changes are confined to the local copy until the session ends. At the end of the session, the local dictionary copy is sent back to the server.

If there are 100 users, each with a unique ID, there will be 100 user dictionaries stored in the default or specified path. If many users login using the same user ID (such as Clerk or Underwriter), then a single shared dictionary is created or used for that group of users.

Regardless of local or server storage, the dictionary gets put in the same directory as the plug-in binaries on the client machine. On the default install, this is...

> C:\Program Files\Docucorp International\Docucorp WipEdit\

When the user with a server side dictionary terminates his session, a copy of his local working dictionary is uploaded to the server and saved. The next time the user logs in, a copy of this dictionary is downloaded to the user's local machine.

Fonts

The WIP Edit installation program installs and registers a set of True Type fonts. It stores them in the directory where the main WIP Edit program files are located. The default location for the files is:

> C:\Program Files\Docucorp International\Docucorp WIP Edit

If your MRL requires additional fonts, you must install them on the client side and register them with the Windows Font Manager. Refer to the Help files on your client machine for information on how to register a font in Windows.

Printing

There is a glyph button bar at the top. Those should be used. Tell the user not to print locally. Instead they should click on the Proof button. This generates a PDF file and launches it in Adobe Reader. Users should print using Acrobat Reader's File, Print option.

Saving

Use the Save to WIP button. This lets you specify a description and other options. When you click Submit, the system sends the WIP to the server where it is saved.

Zoom in and Zoom out

You can update the zoom by clicking on the Zoom in/out button on the tool bar. The default zoom can be set in the wipedit.ini based on the Control group option:

```
<Control>
Zoom = value
```

The following are the values:

- ToWidth: ToWidth indicates to fit to width.

- ToWindow: ToWindow indicates to fit the display to the current window size.

- 25, 50, 75, Normal, 100, 125, 150, 200, 300, and 400: The numeric values are actual zoom values. Normal and 100 are equivalent.

Note: Any other values other than the above listed will be ignored.

There is an additional Control group option for Zoom settings, that is The SaveZoom option, when set to Yes, indicates that the user's last zoom setting is retained. If SaveZoom is Yes, the Zoom option default value is not used.

WIP Edit APIs

This listing shows the WIP Edit IDL. Since WIP Edit is an ActiveX control, it is derived from IDispatch. The methods listed in that section of the interface are callable by external programs or scripts. You can write custom scripts or code and call the interfaces in WIP Edit to customize how you want the control to display your forms or sections.

WIP Edit IDL

```
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib filename: wipctl.dll

[
  uuid(39B7F35B-067C-488A-8D01-717DA4D9E1DF),
  version(1.0),
  helpstring("wipctl 1.0 Type Library"),
  custom(DE77BA64-517C-11D1-A2DA-0000F8773CE9, 100663657),
  custom(DE77BA63-517C-11D1-A2DA-0000F8773CE9, 1116428288),
  custom(DE77BA65-517C-11D1-A2DA-0000F8773CE9, Created by MIDL version
6.00.0361 at Wed May 18 10:58:07 2005
)

]
library WIPCTLLib
{
    // TLib :    // TLib : OLE Automation : {00020430-0000-0000-C000-000000000046}
    importlib("stdole2.tlb");

    // Forward declare all types defined in this typelib
    interface IWipEd;

    [
      uuid(76469354-4683-4ECE-B0A0-52A56A590275),
      helpstring("WipEd Class")
    ]
```

```
coclass WipEd {
    [default] interface IWipEd;
};

[
  odl,
  uuid(24654DF4-8E2B-42D3-A323-AABDD41F23B9),
  helpstring("IWipEd Interface"),
  dual,
  oleautomation
]
interface IWipEd : IDispatch {
    [id(0x00000001), helpstring("method cmd")]
    HRESULT cmd(int cmd);
    [id(0x00000002), helpstring("method GotoForm")]
    HRESULT GotoForm(
                BSTR formname,
                int formno,
                int pageno);
    [id(0x00000003), helpstring("method Save")]
    HRESULT Save();
    [id(0x00000004), helpstring("method FitToWidth")]
    HRESULT FitToWidth();
    [id(0x00000005), helpstring("method FitToWindow")]
    HRESULT FitToWindow();
    [id(0x00000006), helpstring("method ZoomIn")]
    HRESULT ZoomIn();
    [id(0x00000007), helpstring("method ZoomOut")]
    HRESULT ZoomOut();
    [id(0x00000008), helpstring("method ZoomNormal")]
    HRESULT ZoomNormal();
    [id(0x00000009), helpstring("method FormPrevious")]
    HRESULT FormPrevious();
    [id(0x0000000a), helpstring("method FormNext")]
    HRESULT FormNext();
    [id(0x0000000b), helpstring("method FormSelect")]
    HRESULT FormSelect();
    [id(0x0000000c), helpstring("method Refresh")]
    HRESULT Refresh();
    [id(0x0000000d), helpstring("method FieldTemplate")]
    HRESULT FieldTemplate();
    [id(0x0000000e), helpstring("method AutoFocus")]
    HRESULT AutoFocus();
    [id(0x0000000f), helpstring("method Information")]
    HRESULT Information();
    [id(0x00000010), helpstring("method FixedEdit")]
    HRESULT FixedEdit();
    [id(0x00000011), helpstring("method FixedPrompt")]
    HRESULT FixedPrompt();
    [id(0x00000012), helpstring("method Cascade")]
    HRESULT Cascade();
    [id(0x00000013), helpstring("method Tile")]
    HRESULT Tile();
    [id(0x00000014), helpstring("method Stack")]
    HRESULT Stack();
```

```
[id(0x00000015), helpstring("method StackOnly")]
HRESULT StackOnly();
[id(0x00000016), helpstring("method HelpContents")]
HRESULT HelpContents();
[id(0x00000017), helpstring("method HelpHowTo")]
HRESULT HelpHowTo();
[id(0x00000018), helpstring("method HelpGlossary")]
HRESULT HelpGlossary();
[id(0x00000019), helpstring("method UsingHelp")]
HRESULT UsingHelp();
[id(0x0000001a), helpstring("method PagePrevious")]
HRESULT PagePrevious();
[id(0x0000001b), helpstring("method PageNext")]
HRESULT PageNext();
[id(0x0000001c), helpstring("method SelectSection")]
HRESULT SelectSection();
[id(0x0000001d), helpstring("method ProductionInformation")]
HRESULT ProductionInformation();
[id(0x0000001e), helpstring("method HelpShortcuts")]
HRESULT HelpShortcuts();
[id(0x0000001f), helpstring("method CmdWithMessage")]
HRESULT cmdWithMessage(
        int cmd,
        VARIANT* v);
[id(0x00000020), helpstring("method Terminate")]
HRESULT Terminate();
[id(0x00000021), helpstring("method ReservePort")]
HRESULT ReservePort(VARIANT* port);
[id(0x00000022), helpstring("method SetBasePort")]
HRESULT SetBasePort(int basePort);
[id(0x00000023), helpstring("method FreePort")]
HRESULT FreePort(int FreePort);
[id(0x00000024), helpstring("method GetWipField")]
HRESULT GetWipField(
        VARIANT name,
        VARIANT* value);
[id(0x00000025), helpstring("method SetWipField")]
HRESULT SetWipField(
        VARIANT name,
        VARIANT value);
[id(0x00000026), helpstring("method SetCurrentPort")]
HRESULT SetCurrentPort(int basePort);
[id(0x00000024), helpstring("method GetWipField")]

        };
    };
```

Listing 15: ActiveX IDL File for the WIP Edit Control

WIP Edit ActiveX control methods

This table shows a list of methods exposed in the WIP Edit ActiveX control. If you write custom scripts, you can call these methods to make the control display your forms and sections in customized ways.

Method	Description	Comments
void cmd(int cmd)	Executes functions defined in WIPEDIT.RES. WIPEDIT.RES is the file that defines the menu for WIP Edit.	Available for 12.1+. While the WIPEdit.res is no longer supported in 12.0+, the following commands are still available: 260 - Save 262 - Check required fields 263 - Get Formset field data. 1014 - Fit to Width. 1072 - Fit to Window 1010 - zoom in 1011 - zoom out 1012 - zoom normal 1008 - previous form 1007 - next form 1034 - previous page 1033 - next page 264 - hide form navigation window 265 - show form navigation window 266 - toggle between show/ navigation show window.
void GotoForm(BSTR formname, int formno, int pageno	Changes the current form that is being edited. Form name is the name of the form in the form set. Form no is the instance of the form. Form set can have multiple forms with the same name. Page no is the page within the form.	Not available in version 12.0+ due to the revised user interface (UI).
void Save(void)	Saves the document. The document is saved on the server but does not close the document in WIP Edit.	
void FitToWidth(void)	Displays the full width of the section in the active window.	
void FitToWindow(void)	Displays the entire section in the active window	

*1 - Only available in version 11.1 of WIP Edit.

Method	Description	Comments
void ZoomIn(void)	Increases the magnification	
void ZoomOut(void)	Decreases the magnification	
void ZoomNormal(void)	Returns to the 100% magnification value	
void FormPrevious(void d)	Moves to the previous form in the form set	
void FormNext(void)	Moves to the next form in the form set	
void PagePrevious(void)	Moves to the previous page	
void PageNext(void)	Moves to the next page.	
void hideNavBar(void)	Hides the form from the navigation window.	
void showNavBar(void)	Displays the form in the navigation window.	
void toggleNavBar(void)	Select to toggle navigation bar.	
void FormSelect(void)	Selects the form in a form set. This is helpful when you view a stacked form set.	
void Refresh(void)	Redraws the display	Not available for version 12.0+.
void FieldTemplate(void d)	Allows to view the size and location of variable fields in a form.	Not available for version 12.0+.
void AutoFocus(void)	Tells the system to scroll the form as you move through the fields on the form. The current field always stays in the view. If you turn this off, your current field may not stay displayed on screen.	Not available for version 12.0+. This is a default action.
void Information(void)	Displays additional information about the variable fields in the section.	Not available in version 12.0+.This information is always displayed.

*1 - Only available in version 11.1 of WIP Edit.

Method	Description	Comments
void FixedEdit(void)	Anchors the data entry area at the top of your screen instead of having it move as you move through various fields.	Not available for version 12.0 and 12.1 and Implemented for 12.1.1+.
void FixedPrompt(void)	Displays the fixed prompt information.	Not available in version 12.0+.This information is always displayed.
void Cascade(void)	Displays multiple form or section windows in layers.	Not available in version 12.0+ due to the revised user interface (UI).
void Tile(void)	Displays multiple form or section windows in an arrangement on the screen.	Not available in version 12.0+ due to the revised user interface (UI.
void Stack(void)	Displays multiple form or section windows stacked on top of each other	Not available in version 12.0+ due to the revised user interface (UI)
void StackOnly(void)	Displays a form set in a stack. This is the default mode when you retrieve archived form sets.	Not available in version 12.0+ due to the revised user interface (UI).
void HelpContents(void)	Displays the Help file table of contents.	Not available in version 12.0+. User can view the help content on OTN by accessing the link from the Documaker Interactive or from the application.
void HelpHowTo(void)	Displays the How to topics from the help file.	Not available in version 12.0+. User can view the help content on OTN by accessing the link from the Documaker Interactive or from the application.
void HelpGlossary(void)	Displays the Help file glossary.	Not available in version 12.0+. User can view the help content on OTN by accessing the link from the Documaker Interactive or from the application.
void UsingHelp(void)	Displays the How to use Help topic	Not available in version 12.0+. User can view the help content on OTN by accessing the link from the Documaker Interactive or from the application.
void PagePrevious(void)	Moves to the previous page in a form set	

*1 - Only available in version 11.1 of WIP Edit.

Method	Description	Comments
void PageNext(void)	Moves to the next page in a form set	
void SelectSection(void)	Selects the specific page you want to view. This is helpful when you are viewing a stacked form set.	Not available in version 12.0+ due to the revised user interface (UI).
void ProductInformatio n(void)	Displays the product information	Not available in version 12.0+.
void HelpShortcuts(voi d)	Displays the shortcuts in the Help file	Not available in version 12.0+. User can view the help content on OTN by accessing the link from the Documaker Interactive or from the application.
void cmdWithMessage(i nt cmd, variant v)	If a user writing a script receives a response message from WIP Edit. The function in WIPEDIT.RES must return a result. A MEN.RES function must be written to handle the said situation. For an example, see Not available. replaced with cmdGetResponse RACCCheckRequiredFields. The variant contains the response from IDS.	Not available. replaced with cmdGetResponse
BSTR cmdGetResponse(i nt cmd)	Can be used with cmd id's 260 and 262 to determine the status of save attempt and check required fields.	
void Terminate	Reserved	Not available in version 12.0+.
void ReservePort(VARI ANT *port)	Gets the next available port address. The parameter returns a valid port address.	Not available in version 12.0+.
void SetBasePort(int basePort)	Sets the beginning address of the port. All port addresses will be greater than or equal to this one. If you do not use this method, the default will be 49180. This is available to fix conflicts with other applications.	Not available in version 12.0+.
void FreePort(int FreePort);	Makes the port address available to another instance of IDS.	Not available in version 12.0+.
void GetWipField(VAR IANT name, VARIANT *value);	Lets you retrieve a WIP index field from the document open in WIP Edit.	

*1 - Only available in version 11.1 of WIP Edit.

Method	Description	Comments
void SetWipField(VARIANT name, VARIANT value);	Lets you set a WIP index field in the document that is open in WIP Edit.	
void SetCurrentPort(int port);	Tells the WIPCTL.dll to use this port address.	Not available in version 12.0+.
BSTR GetWipIndex(VARIANT fldName);	Lets you retrieve a WIP index field from the document open in WIP Edit.	
BSTR isdirty()	Returns 1 if formset has changed. Returns 0 if formset has not been changed the same.	
BSTR ReturnReservePort()	Gets the next available port address. The parameter returns a valid port address.	
BSTR checkRequiredField()	Returns true if all required fields have data. Returns false if a field retires data and is empty.	
BSTR getRequiredFieldName();	Returns true if all required fields have data. Returns false if a field requires data and is empty.	

*1 - Only available in version 11.1 of WIP Edit.

Method	Description	Comments
void cmdCallBack(long cmd, VARIANT callback)	When the cmd function is completed a javascript function is called and the name of the javascript function is defined by the callback parameter	The WIPEdit.res is no longer supported in 12.0+. The following commands are still available: 260 - Save 2 62 - Check required fields 263 - Get Formset field data. 1014 - Fit to Width. 1072 - Fit to Window 1010 - zoom in 1011 - zoom out 1012 - zoom normal 1008 - previous form 1007 - next form 1034 - previous page 1033 - next page 264 - hide form navigation window 265 - show form navigation window 266 - toggle between show/ navigation show window.
void checkRequiredFiel dCallBack()	Returns true if all required fields have data. Returns false if a field retries data and is empty.	
BSTR cmdGetResponse WithParm(int cmd, VARIANT parm);	It can only be used with custom functions.	Not available in version 12.0+.
BSTR SetPortByUniqueI d(BSTR sessid)	obsolete	obsolete
void setInputEncoding(int encode);	If encode flag is set to 1 then it is assumed parameters to all functions are already UTF-8.	
BSTR getVersion(void)	Return version of plug-in.	

*1 - Only available in version 11.1 of WIP Edit.

USING THE WIP EDIT PLUG-IN

The Documaker wipedit plug-in allows you to present WIP documents in the browser.The Internet Explorer version of the plug-in uses an Active Document Server for Documaker 12.0 and lower.Version 12.3 uses ActiveX control. Firefox version of the plug-in uses the NPAPI plug-in.

The plug-in is utilized by Documaker web applications to enable document editing. Alternatively, you may use the plug-in within your own web application to enable editing. The following examples show how to implement the plug-in within your own web application.

Internet Explorer Markup below.

```
<object classid="clsid:F894A210-B1E8-44D2-A3DB-5C2E86C7408D" id='plugin' height='1000' width='1000'/>

<!--param id='cookie' name='cookie' value=<%out.write(UUID.randomUUID().toString());%>-->

<param id='cookie' name='cookie' value="<%

String cookieName = "username";

Cookie cookies [] = request.getCookies ();

if (cookies != null)

{

for (int i = 0; i < cookies.length; i++)

{

    out.write(cookies[i].getName() + "=" + cookies[i].getValue() +"; ");

}

}

%>">

<param name = 'docsavescript' value='save_script'>

<param name = 'getscript' value='get_script'>

<!-- The puturl will need to be customized for your system -->

<param name = 'puturl' value='url'>

<!-- The src parameter will need to be customized for your system -->

<param name = 'src' value='url for dpw file'>

</object>


Firefox plug-in mark-up:
```

<object id='plugin' data=path to generate a DPW file height='1000'

width='1000' type='application/x-dpwfile'/>

<param id='cookie' name='cookie' value=list of cookies name value pairs>

<param name = 'docsavescript' value='save_script'>

<param name = 'getscript' value='get_script'>

<param name = 'puturl' value=url to save DPW file>

The WIP Edit plug-in dynamically requests the downloading of the following resourcesfrom IDS. The DPRGetResource rule looks in your INI options to locate any resources requested. Once obtained, the resources are packaged into a DPW file and downloaded to the client machine.

• FAP files

• DAL scripts

• Tables

• Help files

We use the docserv.xml or the include file specified in the docserv.xml to provide the rules and functions needed to perform the GetResource activity.

<section name="INIFiles">

</section>

[ReqType:GETRESOURCE]

function = atcw32->ATCLogTransaction

function = atcw32->ATCLoadAttachment

function = atcw32->ATCUnloadAttachment

function = dprw32->DPRSetConfig

function = dprw32->DPRDecryptLogin

function = dprw32->DPRDefaultLogin

function = dprw32->DPRCheckLogin

function = atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary

function = dprw32->DPRGetResource,RETURNFILE

You can add entries to WIP by including these request types. This request type creates a DPW file that triggers the Form Selection window.

[ReqType:GETEMPTYWIP]

function = atcw32->ATCLogTransaction

function = atcw32->ATCLoadAttachment

function = atcw32->ATCUnloadAttachment

function = dprw32->DPRSetConfig

function = dprw32->DPRDecryptLogin

function = dprw32->DPRDefaultLogin

function = dprw32->DPRCheckLogin

function = atcw32->ATCSendFile,RETURNFILE,RETURNFILE,Binary

function = dprw32->DPRCreateEmptyWipXML,RETURNFILE

function = dprw32->DPRFile2Dpw,RETURNFILE

function = dprw32->DPRIni2XML

Set this request type to determine if a policy number is already being used.

[ReqType:WFIND]

function = atcw32->ATCLogTransaction

function = atcw32->ATCLoadAttachment

function = atcw32->ATCUnloadAttachment

function = dprw32->DPRSetConfig

function = dprw32->DPRDecryptLogin

function = dprw32->DPRDefaultLogin

function = dprw32->DPRCheckLogin

function = dprw32->DPRFindWipRecords

Here are examples of entries in the INI2XML control group:

< INI2XML >

PolicyScript = doc-prog/iwip/sampco/wipfound.asp

GetScript = doc-prog/iwip/sampco/wipdownload.asp

Key1 = FORMMAKER PACKAGE

Key2 = PROPERTY;INLAND MARINE

CUSTOMIZING IDOCUMAKER, IPPS, AND WIP EDIT

Here are some things to keep in mind as you use the WIP Edit plug-in:

Turning on debugging

You can use the Debug option to turn on debugging. This lets you turn on debugging without having to individually set the environment variable on client machines running the WIP Edit plug-in.

To turn on debugging using the Debug option, include this option in the wipedit.ini file:

< WIPEdit >

    Debug = Yes

Automatically sending the WIPEDIT.FXR file

The IDS rules that create the DPW file can automatically send the WIPEDIT.FXR file to the WIP Edit plug-in when these conditions are met in the INI file:

· The DownloadDPWFonts option in the WIP2DPW control group is set to No.

· The XRFToken option is not set in the File2DPW control group.

In the sampco.ini file, comment out the XRFToken option, as shown in this example:

< File2DPW >

;   XRFToken = mstrres\sampco\deflib\rel102sm.fxr

NOTE: You can only have one installation of the WIP Edit plug-in on a PC.

Saving documents with invalid certificates

The WIP Edit plug-in ignores invalid web certificates, such as when the web certificate has expired. If the certificate is invalid, the system can save the document from the WIP Edit plug-in with the following INI options:

To begin, download an INI file to the WIP Edit plug-in. For this example, use the USER.INI file. Add the following to the configuration specific INI:

< File2DPW>

    INIToken = user.ini

The USER.INI file should contain the following.

< ICMLib>

    IgnoreInvalidCertificate = Yes

Running the plug-in outside the browser

You can run the WIP Edit plug-in in its own window (outside the browser) by changing the content type header in the WIPEDIT.ASP or WIPEDIT.JSP web page.

Changing values in the WIP index

Use the UpdateDpwIndex INI option to change values in the WIP index based on session variables created in the wipedit.jsp or wipedit.asp page. This option will probably always be used with a customization to the web page to update the WIP index with data from an external source.

If you need to change a WIP index field with a value that originates in the ASP/JSP page, you can use the UpdateDPWIndex option to modify the WIP record when the document is saved. For example, you can use this option to track some other user ID than the login ID the page prompts for.

The following lines are in the wipedit.asp page. A session variable is created called SETORIGUSER. This information is passed to IDS in the form of an attachment variable by the DSI.ProcessQ:

session("SETORIGUSER") = "testchange"

On Error Resume Next

DSI.ProcessQ 'Execute Request From Attachment

The configuration specific INI must have the following UpdateDpwIndex option:

< UpdateDPWIndex >

OrigUser = #SETORIGUSER

The # character tells the system to get the data from the attachment variable named SETORIGUSER. Without the #, the WIP index is updated with the text in the INI file.

The DPRIndex2Xml rule reads the UpdateDpwIndex control group and makes changes in the index portion of the DPW file. When the DPW file is saved, the DPRDpw2Wip rule updates the WIP index with the change.

Using WIP Edit with SiteMinder®

You can use the WIP Edit plug-in with web sites protected by SiteMinder® and with web sites that use clustered web servers. SiteMinder stores security information in a cookie. The WIP Edit plug-in looks for this cookie and attaches the cookie information to requests for resources and the saving of documents.

SENDING PASSWORDS

IDS can use the DPRIni2Xml rule to pass an encrypted password to the WIP Edit plug-in to provide authentication when saving data back to IDS.

```
< INI2XML >

    HTTPUserID        = encrypteduserID

    HTTPPassword      = encryptedpassword
```

You can also use the cryruw32 program to create an encrypted value that can be understood by the WIP Edit plug-in. This lets you avoid putting passwords in the INI file where they can easily be read. For instance, if you enter this from the command line:

```
cryruw32.exe password
```

you will see the output similar to the following:

```
Encrypted string (2XAUnkxUYlx7i5AnQ4m4E1m00)
```

REQUESTING A DICTIONARY

The WIP Edit plug-in can request a user spelling dictionary from IDS when running a spell check.

Use the DPRINI2XML rule to calculate a CRC (Cyclic Redundancy Check) that will be stored in the DPW file. This line will calculate the CRC of a spelling dictionary specified by the user ID:

```
< INI2XML >

    CalcCRC = d:\docserv1\spell\#USERID.tlx!TLX
```

To update the spelling dictionary if the WIP Edit plug-in has changed it, use the DPRPutResource rule:

```
[ ReqType:PUTRESOURCE]
```

```
function = atcw32->ATCLogTransaction

function = atcw32->ATCLoadAttachment

function = atcw32->ATCUnloadAttachment

function = dprw32->DPRSetConfig

function = dprw32->DPRDecryptLogin

function = dprw32->DPRDefaultLogin

function = dprw32->DPRCheckLogin

function = dprw32->DPRPutResource
```

Specifying the user dictionary

Use the UserDict option to specify the name of the dictionary file you want to use in the WIP Edit spell check process. If you omit this option, the spell dictionary file name is based on the user ID.

To begin, download an INI file to the WIP Edit plug-in. For this example, use USER.INI. Add the following to the configuration-specific INI file:

```
< File2DPW >

    INIToken = user.ini

< Spell >

    UserDict = dictionary.tlx
```

TRAPPING EVENTS

The options to control the trapping of events were implemented because web pages that use anchor tags cause WIP Edit to exit prematurely. If your web page contains anchor tags you may need these options.

These INI options are in the INI file downloaded to WIP Edit, usually named *WIPEDIT.INI*. The INI file is specified in the INIToken option, as shown below:

```
< INI2XML >

    INIToken                = wipedit.ini

< WIPEdit >

    DisableRightClick       =

    TrapEvents              =

    TrapOnlyQuitEvent       =
```

NOTE: Whether the document is saved or whether you are prompted to save the document depends on the following options in the WIPEDIT.INI file. If you set the OverridePrompt option to Yes, you are not prompted when the plug-in closes. The default is No.

```
< WIPSave >

    OverridePrompt =
```

If you want WIP Edit to automatically save the document. Set the OverridePrompt option to Yes and set the SaveOnExit option to Yes.

```
< WIPSave >
```

SaveOnExit =

The default for the SaveOnExit option is No.

Note: Whether the document is saved or whether you are prompted to save the document depends on the following options in the WIPEDIT.INI file. If you set the OverridePrompt option to Yes, you are not prompted when the plug-in closes. The default is No.

< WIPSave >
    OverridePrompt =

If you want WIP Edit to automatically save the document. Set the OverridePrompt option to Yes and set the SaveOnExit option to Yes.

< WIPSave >
    SaveOnExit =

The default for the SaveOnExit option is No.

TRACKING SESSION INFORMATION

The WIP Edit plug-in will let a web application specify data that will be sent back to the web server when a document is saved. This lets iPPS or iDocumaker send session information to the web server/IDS when saving data or getting resources.

The DPRPrintDpw rule looks for groups of attachment variables to add information to the DPW file. This information is used by WIP Edit to add data to the GETRESOURCE and WIPSAVE request.

Use HTTPFORMDATA variables to add multiform post data:

Variable	Description
HTTPFORMDATA	The number of variables to add multiform post data
HTTPFORMDATA#.NAME	The name of the variable.
HTTPFORMDATA#.VALUE	The value of the variable.

Use HTTPQUERYSTRING variables to add the query string:

Variable	Description
HTTPQUERYSTRING	The number of variables to add to the query string
HTTPQUERYSTRING#.NAME	The name of the variable.
HTTPQUERYSTRING#.VALUE	The value of the variable.

Use the HTTPHEADER variables to add the HTTP header:

Variable	Description
HTTPHEADER	The number of variables to add to the HTTP header
HTTPHEADER#.NAME	The name of the variable.
HTTPHEADER#.VALUE	The value of the variable.

Use the HTTPCOOKIE variables to add the cookie header:

Variable	Description
HTTPCOOKIE	The number of variables to add to the cookie header
HTTPCOOKIE#.NAME	The name of the variable.
HTTPCOOKIE#.VALUE	The value of the variable.

Examples   Here are some examples:

To add multipart form data to the HTTP request the following attachment variables were added to the request that creates the DPW file:

```
HTTPFORMDATA = 1
HTTPFORMDATA1.NAME = nameformdata1
HTTPFORMDATA1.VALUE = valueformdata1
```

The resulting line in the HTTP request would look like this:

```
-----------------------------7d32f01b1003de
Content-Disposition:form-data; name="nameformdata1"


valueformdata1
```

To add data to the query string for the HTTP request these attachment variables were added to the request that creates the DPW file.

```
HTTPQUERYSTRING 1
HTTPQUERYSTRING1.NAME = SESSIONID
HTTPQUERYSTRING1.VALUE = 8010e572-001b-43e3-98f4-e1b0e0116933
```

In the resulting line in the HTTP request, HTTPQUERYSTRING adds the following information to the URL. Here is an example:

```
/doc-prog/iwip/sampco/wipsave.asp?SESSIONID= 8010e572-001b-43e3-98f4-
e1b0e0116933 HTTP/1.1
```

To create a header for the HTTP request these attachment variables were added to the request that creates the DPW file:

```
HTTPHEADER = 1
HTTPHEADER1.NAME = someheader1
HTTPHEADER1.VALUE = someVALUE1
```

In the resulting line in the HTTP request, HTTPHEADER adds information to the HTTP header. The following example is from a save request:

```
someheader1:someVALUE1
```

To add data to the cookie header the HTTP request the following attachment variables were added to the request that creates the DPW file:

```
HTTPCOOKIE = "1"
HTTPCOOKIE1.NAME = "cookie"
HTTPCOOKIE1.VALUE = "Toolfloat=false; Toolbottom=false; IX=%E9%C4%92%
08%C9%D3xo%D9%2D%AF%D3%A0%AC%26%15%7E%FA%23M%01%D9%FDt%23%
A2%13%7E%CAN%95%80%B2%
E5cC%0Enj%E7%1E%E4;
ASPSESSIONIDACRTQSCA=EGPPLAECPNDMGOIIMANOAPPB"
```

The resulting cookie header in the HTTP request would look like this:

```
Cookie: Toolfloat=false; Toolbottom=false;
IX=%E9%C4%92%08%C9%D3xo%D9%2D%AF%D3%A0%AC%26%15%7E%FA%23M%
01%D9%FDt%23%A2%13%7E%CAN%95%80%B2%E5cC%0Enj%E7%1E%E4;
ASPSESSIONIDACRTQSCA=EGPPLAECPNDMGOIIMANOAPPB;
ASPSESSIONIDQSBCQTCA=JHCKEELAANHOGDGAPMABIHDL
```

Advanced Topics

This chapter discussed a variety of tasks you can to do enhance and customize your web-enabled solution once you perform the initial installation and configuration.

These topics include:

NOTE: For information on how to configure passwords with basic Windows authentication, refer to .

DEBUGGING

Server Side Debugging

DPW file contents

You can set an option your configuration INI file (such as AMERGEN.INI) to turn on server side debugging for the WIP Edit. The configuration option looks like this:

```
< WIP2DPW >
    Debug = Yes
```

When IDS is configured with this setting, it retains a copy of all of the files that comprise the DPW file and store them in a local temporary directory. This increases the amount of storage required and the files *are not* automatically erased. Therefore, only use this debugging option when you need to track down a problem.

The files are stored in directories that are named by GUIDs under the data directory within the Docserv directory as shown in Figure 6. There will be a directory for each instance of IDS.
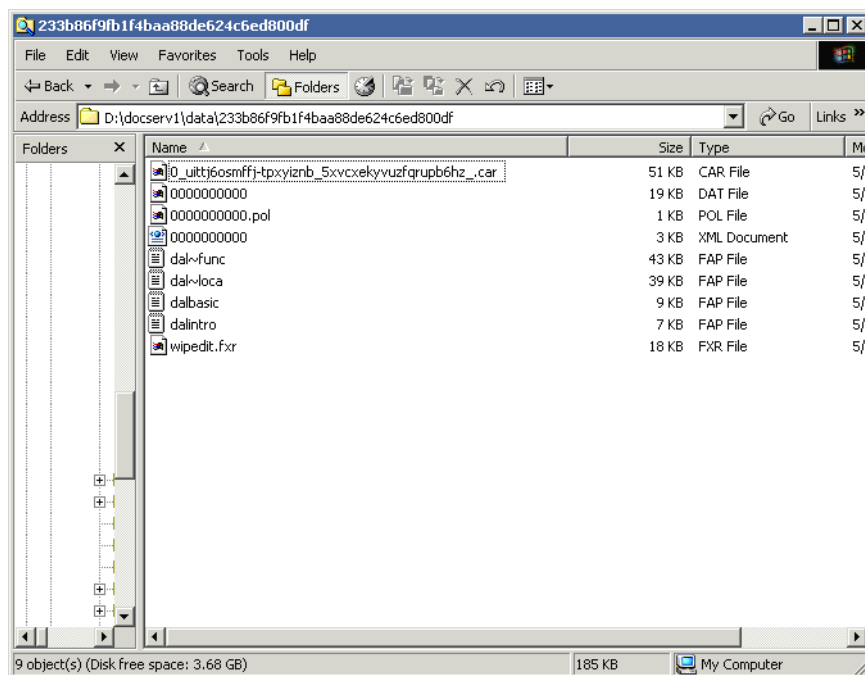
Figure 6: Debug Files in the Temp directory

In addition to retaining the files that make up the DPW file that is sent to the client side WIP Edit ActiveX control, additional logging information is also generated. With this option set, information is written to the Docserv trace log file (C:\Docserv\dsrvtrc.log). Listing 16 shows an extract of a Docserv trace file with this debugging option turned on.

```
1. Thu May 26 06:36:03.298 2005 pid=00001996 DPWInitDpwPi()  - Begin
2. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.currDirectory=<C:\docserv>
3. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.dataPath=<C:\docserv\data>
4. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.wipPath=<mstrres\sampco\wip\>
5. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.wipFile=<mstrres\sampco\wip\wip>
```

```
6. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.menuFile=<C:\docserv\wipedit.res>
7. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.fileName=<0upm3sx_cweebhues2h76kilf_vmj4nkh-ecktyvvokp7>
8. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.transactionId=<1146>
9. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.groupName1=<DOCUCORP PACKAGE>
10. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.groupName2=<DOCUMENT AUTOMATION LANGUAGE>
11. Thu May 26 06:36:03.308 2005 pid=00001996 dpwpi.groupName3=<>
12. Thu May 26 06:36:03.308 2005 pid=00001996 dpwpi.recipient=<>
13. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.arcPath=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df>
14. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.appidxFile=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df\
appidx>
15. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.catalogFile=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df
\>
16. Thu May 26 06:36:03.308 2005 pid=00001996 dpwpi.dpwFile=<0uS-
FvQQu9wVduVNivwh5ywMCkBZvz29ViUEj0Q0KS0nT.DPW>
17. Thu May 26 06:36:03.308 2005 pid=00001996
dpwpi.naFile=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df\0000
000000.dat>
18. Thu May 26 06:36:03.318 2005 pid=00001996
dpwpi.polFile=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df\000
0000000.pol>
19. Thu May 26 06:36:03.318 2005 pid=00001996
dpwpi.indexFile=<C:\docserv\data\233b86f9fb1f4baa88de624c6ed800df\0
000000000.xml>
20. Thu May 26 06:36:03.318 2005 pid=00001996 DPWGetDpwIndex()  -
Begin
21. Thu May 26 06:36:03.369 2005 pid=00001996 DPWGetDpwIndex()  - End
```

Listing 16: Sample DSRVTRC.LOG file output

Seeing messages sent by client and server

IDS dumps the contents of the send and receive messages to a file if you add a debug statement to the DOCSERV.INI file.

If the environment does not appear to be working and you get errors that say the system cannot initialize the queue, open the DOCSERV.INI file and add this option:

```
< DBHandler:MQSERIES >
    Debug = Yes
```

This option tells the system to include detailed debug information in the error descriptions. This will help you determine the actual problem. The system writes this information into the DSRVTRC.LOG file (the trace log file).

In the DOCSERV.INI file there is or can be a separate section for debugging. This table shows the items you can include in a Debug section:

Item	Description
[debug]	
xmlmessage = yes	Creates two files: send.msg and receive.msg that have all messages sent to and from IDS.
xmlmessageappend = yes	Works with xmlmessage. Appends messages rather than over write them.

Listing 17: DOCSERV.INI File Debug section

CLIENT SIDE DEBUGGING

You can download a copy of the WIPEDIT.INI locally on the client machine and make it available to the WIP Edit ActiveX control. The INI file will be transformed to a FSIUSER.INI file and will be stored in the tmpform(pid) directory. To do this, modify your configuration file (such as AMERGEN.INI) and add the following settings:

```
< File2Dpw >
    INIToken = wipedit.ini
```

To turn on debugging and logging you have these choices. Pick one of these options:

• Modify the WIPEDIT.INI file (on the server side) and add the following:

```
< WIPEdit >
    Debug = Yes
```

• Set an environment variable (WIPEDITDEBUG=Yes) on the client machine.

Either approach creates a log file on the client machine. Listing 18 shows a sample output of the log file that is created on the client machine when debugging is turned on. You should only activate this setting if you are having difficulty and need to do some troubleshooting.

```
1. Thu May 26 08:08:20.991 2005 pid=00002124 IE instance matched=0
2. Thu May 26 08:08:46.688 2005 pid=00002124 Attempting to load
icmw32.dll
3. Thu May 26 08:08:46.688 2005 pid=00002124 Loaded icmw32.dll
successful
4. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMGetConnection
5. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMGetTermConnection
6. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMGetStatus
7. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMBuildIdsVariables
8. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMSendRequest
9. Thu May 26 08:08:46.688 2005 pid=00002124 Query Function
ICMGetResponse
10. Thu May 26 08:08:46.688 2005 pid=00002124 icmw32.dll loaded and
all functions queried successfully
11. Thu May 26 08:08:46.688 2005 pid=00002124 From DPW file recnum =
90 config = SAMPCO url = stevesmith2:49200
```

12. Thu May 26 08:08:46.688 2005 pid=00002124 Port ID obtained from URL port = 1242888

13. Thu May 26 08:08:46.688 2005 pid=00002124 Attempting connection to url=stevesmith2 port=49200

14. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to get URLPARM

15. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to get ENCRYPTEDLOGIN

16. Thu May 26 08:08:46.728 2005 pid=00002124 Got ENCRYPTEDLOGIN from DPW file 181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0

17. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to set IDS variable (ENCRYPTEDLOGIN)(181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0)

18. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to set IDS variable (REQTYPE)(WIPSAVE)

19. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to set IDS variable (CONFIG)(SAMPCO)

20. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to set IDS variable (DPWRECNUM)(90)

21. Thu May 26 08:08:46.728 2005 pid=00002124 Attempting to Send (doc-prog/iwip/sampco/wipsave.asp)

22. Thu May 26 08:08:47.439 2005 pid=00002124 Number bytes recv'd=26 HTTP status=200

23. Thu May 26 08:08:47.439 2005 pid=00002124 Sent request to IDS succesfully

24. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to load icmw32.dll

25. Thu May 26 08:09:28.789 2005 pid=00002124 Loaded icmw32.dll successful

26. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMGetConnection

27. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMGetTermConnection

28. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMGetStatus

29. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMBuildIdsVariables

30. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMSendRequest

31. Thu May 26 08:09:28.789 2005 pid=00002124 Query Function ICMGetResponse

32. Thu May 26 08:09:28.789 2005 pid=00002124 icmw32.dll loaded and all functions queried successfully

33. Thu May 26 08:09:28.789 2005 pid=00002124 From DPW file recnum = 90 config = SAMPCO url = stevesmith2:49200

34. Thu May 26 08:09:28.789 2005 pid=00002124 Port ID obtained from URL port = 1242888

35. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting connection to url=stevesmith2 port=49200

36. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to get URLPARM

37. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to get ENCRYPTEDLOGIN

38. Thu May 26 08:09:28.789 2005 pid=00002124 Got ENCRYPTEDLOGIN from DPW file 181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0

39. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to set IDS variable (ENCRYPTEDLOGIN)(181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0)

40. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to set IDS variable (REQTYPE)(WIPSAVE)

41. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to set IDS variable (CONFIG)(SAMPCO)

42. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to set IDS variable (DPWRECNUM)(90)

43. Thu May 26 08:09:28.789 2005 pid=00002124 Attempting to Send (doc-prog/iwip/sampco/wipsave.asp)

44. Thu May 26 08:09:29.390 2005 pid=00002124 Number bytes recv'd=26 HTTP status=200

45. Thu May 26 08:09:29.390 2005 pid=00002124 Sent request to IDS succesfully

46. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to load icmw32.dll

47. Thu May 26 08:09:36.780 2005 pid=00002124 Loaded icmw32.dll successful

48. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMGetConnection

49. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMGetTermConnection

50. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMGetStatus

51. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMBuildIdsVariables

52. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMSendRequest

53. Thu May 26 08:09:36.780 2005 pid=00002124 Query Function ICMGetResponse

54. Thu May 26 08:09:36.780 2005 pid=00002124 icmw32.dll loaded and all functions queried successfully

55. Thu May 26 08:09:36.780 2005 pid=00002124 From DPW file recnum = 90 config = SAMPCO url = stevesmith2:49200

56. Thu May 26 08:09:36.780 2005 pid=00002124 Port ID obtained from URL port = 1241664

57. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting connection to url=stevesmith2 port=49200

58. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to get URLPARM

59. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to get ENCRYPTEDLOGIN

60. Thu May 26 08:09:36.780 2005 pid=00002124 Got ENCRYPTEDLOGIN from DPW file 181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0

61. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to set IDS variable (ENCRYPTEDLOGIN)(181ckbwMchEiTvkIYfR_Ev-iAuUa5D-We6FdzuzcWyeQKbgS3pkp2zLzUUZi02Yu0)

62. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to set IDS variable (REQTYPE)(WIPUNLOCK)

63. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to set IDS variable (CONFIG)(SAMPCO)

64. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to set IDS variable (DPWRECNUM)(90)

65. Thu May 26 08:09:36.780 2005 pid=00002124 Attempting to Send (doc-prog/iwip/sampco/wipsave.asp)

66. Thu May 26 08:09:36.940 2005 pid=00002124 Number bytes recv'd=24 HTTP status=200

67. Thu May 26 08:09:36.940 2005 pid=00002124 Sent request to IDS succesfully

Listing 18: Sample client-side log file

TROUBLESHOOTING

LINUX CHARACTER SETS

There is an issue with iPPS for Java and Red Hat's Linux, version 7.3. The default character encoding causes problems for iPPSj when the *save2WIP* call is made. Linux 7.3 default encoding is ISO-8859-15 which has no mapping for the following eight characters which caused an issue when saving to WIP.

For this Windows character	Press
¤	ALT + 0164
¦	ALT + 0166
¨	ALT + 0168
´	ALT + 0180
¸	ALT + 0184
¼	ALT + 0188
½	ALT + 0189
¾	ALT + 0190

For more information, go to this web site:

http://www.fileformat.info/info/charset/ISO-8859-15/encode.htm

Character	Encoding
¤	(unmappable)
¥	a5
¦	(unmappable)
§	a7
¨	(unmappable)
³	b3
´	(unmappable)
µ	b5
·	b7
¸	(unmappable)
¹	b9
º	ba
»	bb

Character	Encoding
¼	(unmappable)
½	(unmappable)
¾	(unmappable)

Any issue involving character encoding should be handled in these steps:

1 Request the default encoding and operating system version

· From a Linux/UNIX shell enter this command:

env | [less][more]

· Find the entry for: *LANG=en_US.iso885915*. This entry may be slightly different on other operating systems.

2 Start the AppServer to make sure you have set the operating system to the LANG encoding causing the issue.

· Set the encoding by using this command:

export LANG=[TARGET ENCODING]

For example, *Ex. en_US*. You can find a list of supported OS encoding at:

/usr/lib/locale [Red Hat 7.3] standard install

The location may vary, depending on the operating system.

· Launch the AppServer startup script from the same shell the LANG *env* was set.

3 Start the iPPSj web application.

· Test the characters that are causing the problem by entering them in a rich text area. Reload the page in the browser by clicking the highlighted page.

· Notice any issues with the entered characters.

· Save the transaction to WIP.

· Retrieve the transaction from WIP.

· Navigate to the data Entry page. Note any issues with character encoding for data entry fields.

The results of this test will determine if iPPSj has an issue with a particular character mapping.

If your current environment does not support the character set, research an acceptable replacement. Also, be careful that characters entered from the client side browser are supported by the AppServer/OS. If they are not you will likely get question marks (?) when a character mapping is not possible. Remember iPPSj can only use characters that are supported by the AppServer/OS. If the AppServer returns a question mark that's what the iPPSj application has to use.

WIP EDIT REGISTRATION

The WIP Edit installation program should automatically take care of registering the ActiveX control with the Windows operating system. However, when you try to display a form and see something similar to Figure 7 instead, this indicates that the plug-in is not properly registered with the Windows environment. To correct this, you can either re-install the plug-in or you can enter this command in an operating system window:

```
wipedw32.exe
```

This should properly register the ActiveX control with the environment.
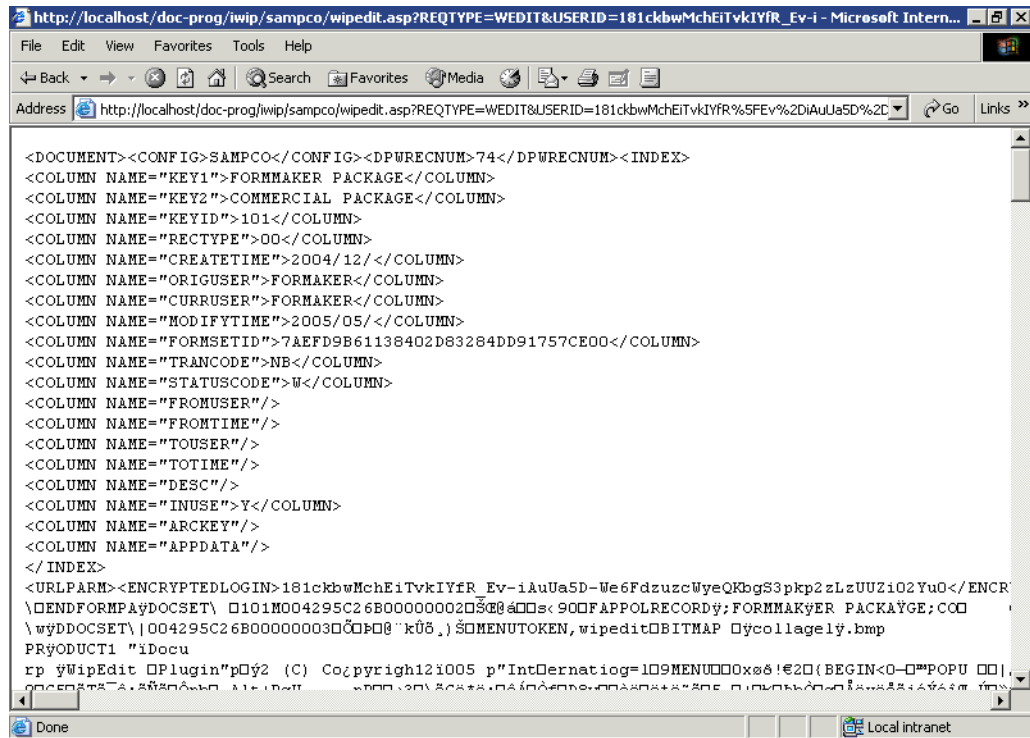
Figure 7: WIP Edit Plug-in is not registered

INTERNET EXPLORER WARNINGS

You can have buttons on your ASP page that tells WIP Edit to save the document, for instance. If the buttons do not seem to work, check to see if Internet Explorer is reporting an error in the lower left corner of the screen. If you see this message:

```
ActiveX component can't create object Wipctl.WipEd.1
```

Try reinstalling the plug-in or register wipctl.dll using regsvr32

If you have buttons defined on the ASP page that tell WIP Edit to save the document and perform other common tasks, but they do not seem to perform the operation, Internet Explorer may report an error similar to Figure 8. If you see this message:

```
ActiveX component can't create object Wipctl.WipEd.1
```

Try re-installing the plug-in or register wipctl.dll using regsvr32 in a DOS window.

Figure 8: WIP Edit control error in Internet Explorer

COULD NOT PARSE THE DPW FILE

If you see an error message like the one shown in Figure 9, check the page that opens the plug-in. In the sample Amergen application it is named WIPEDIT.ASP or WIPEDIT.JSP. Look for a statement like the following (the items in red should match):

```
buff = DSI.ReceiveFileAsBuffer ("RF_POSTFILE")
```

The variable *post file* should match the second parameter of the ATCSendFile rule that is located in the DOCSERV.INI file. Refer to Listing 19 on page 76 for the argument settings.
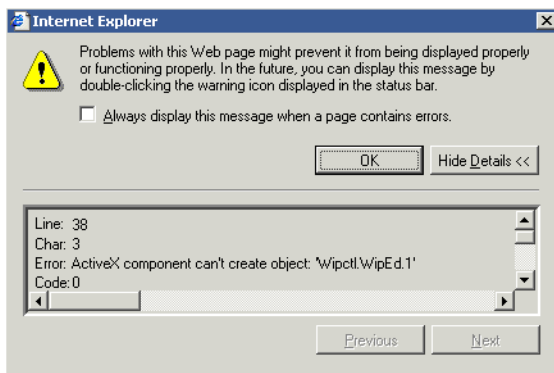

Figure 9: Could not parse DPW File message

```
< ReqType:WEDIT >
    function = atcw32->ATCLogTransaction
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = dprw32->DPRSetConfig
    function = dprw32->DPRGetWipFormset
    function = dprw32->DPRLockWip
    function = atcw32->ATCSendFile,RF_POSTFILE,RF_POSTFILE,Binary
    function = dprw32->DPRWipIndex2XML
    function = dprw32->DPRFile2Dpw,RF_POSTFILE
    function = dprw32->DPRWip2Dpw,RF_POSTFILE
    function = dprw32->DPRIni2XML
```

Listing 19: WIP Edit Section of DOCSERV.INI file

Also, there can be a problem if the rules have different file name parameters. All the parameters in red must match. If the ATCSendFile has a certain value for the first parameter the other rules must match as shown in Listing 20.

```
< ReqType:WEDIT >
    function = atcw32->ATCLogTransaction
    function  = atcw32->ATCLoadAttachment
    function  = atcw32->ATCUnloadAttachment
    function  = dprw32->DPRSetConfig
    function  = dprw32->DPRGetWipFormset
    function  = dprw32->DPRLockWip
    function  = atcw32->ATCSendFile,RF_POSTFILE,RF_POSTFILE,Binary
    function  = dprw32->DPRWipIndex2XML,RF_POSTFILE
    function  = dprw32->DPRFile2Dpw,RF_POSTFILE
    function  = dprw32->DPRWip2Dpw,RF_POSTFILE
    function  = dprw32->DPRIni2XML,RF_POSTFILE
```

Listing 20: File parameters in the WIP Edit Request Type

ANOTHER COULD NOT PARSE DPW FILE MESSAGE

After Documaker Bridge 11.0 Patch 04, you can use the DPRPrintDpw rule as shown in Listing 21 and Listing 22. The rules that were separate in early version have been consolidated. This, however, can cause problems if you do not change the WIPEDIT.ASP page to reflect the new attachment variable names.

Change *RF_POSTFILE* to *PRINTFILE* in the DSI.ReceiveFileAsBuffer call. Otherwise, you will see this message:

```
Could not Parse DPW file
```

because zero (0) bytes will be downloaded in the DPW file.
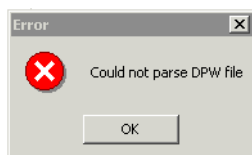
```
< ReqType:WEDIT >
    function  = atcw32->ATCLogTransaction
    function  = atcw32->ATCLoadAttachment
    function  = atcw32->ATCUnloadAttachment
    function  = dprw32->DPRSetConfig
    function  = atcw32->ATCSendFile,PRINTFILE,PRINTFILE,Binary
    function  = dprw32->DPRInitLby
    function  = dprw32->DPRGetWipFormset
    function  = dprw32->DPRPrintDpw
```

Listing 21: Post Documaker Bridge 11.0 p04 Changes for Print Preview

```
< ReqType:i_PluginInit >
;   Plugin initialization. Get a DPW file
    function = atcw32->ATCLoadAttachment
    function = atcw32->ATCUnloadAttachment
    function = atcw32->ATCSendFile,RF_POSTFILE,PRINTFILE,Binary
    function = dprw32->DPRSetConfig
    function = dprw32->DPRInitLby
    function = dprw32->DPRGetWipFormset
    function = dprw32->DPRPrintDpw
```

Listing 22: DPRPrintDpw Rule for iPPS

BIND ERROR

If you see the bind error as shown in Figure 10, you probably have more than one instance of WIP Edit running. However, it is possible that another application could be using the same port.

The WIPEDITPORT environment variable lets you change the port that is used. Version 11.0 and 10.3 of WIP Edit default to port 49150. The default for version 11.1 will use port 49180. Only version 11.1 can support more than one instance of the plug-in running at the same time. ASP pages and INI options, however, must be modified to allow multiple instances of the plug-in to work.
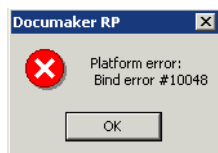
Figure 10: Bind error message

ERRORS WHILE SAVING DOCUMENTS

When a WIP document is saved, the ActiveX control retrieves the address information that will be used, from the DPW file. If any information is incorrect, you will not be able to save the changes. The information is contained in the MRL INI file (such as the AMERGEN.INI file) in the INI2XML control group.

If the PUTURL item is incorrect in the INI file you will see an error message similar to Figure 11. There is a article on line that explains error codes such as 12007. For more information, go to this web site:

http://support.microsoft.com/default.aspx?scid=kb;en-us;193625

Figure 11: Platform error message while saving a document

The following entries in the MRL configuration file (such as AMERGEN.INI) are used to save the document back to WIP. It may be useful to take the values for PUTURL and SCRIPT and paste them together in your browser. This lets you see whether the browser has access to the web page when it tries to save changes. Here is an example:

```
=> http://pd.mysite.com:8080/doc-progr/iwip/amergen/wipsave.asp
```

This URL is based on the following entries in your MRL INI file:

```
< INI2XML >
   PUTURL = pd.mysite.com:8080
   SCRIPT = doc-prog/iwip/amergen/wipsave.asp
```

Error while saving documents on 12.0 P1

If you are running WIPEdit 12.0 P1 or earlier and you experience a failure when saving documents in the WIP Edit plugin, check to see if the web application is running on Unix and if the error code generated by IDS/Docupresentment is DPR0039. In that case, it's possible that you need to update your version of WIP Edit plugin to 12.0 P02 or higher to eliminate the issue as that version has been updated to resolve an issue where the content-type header was improperly set to text type.

> **NOTE:** If you upgrade to 12.0 P02 or higher and start to experience WipEdit plug-in save errors where you didn't previously, there may be a content-header type mismatch and Support should evaluate to determine if this option could correct the issue.
>
> ```
> <icmlib>
> contenttype = "Content-Type: text/plain"
> ```

AUTHENTICATION ERRORS

If the web site has authentication turned on and you have not setup the DPW file to provide password and user information you will see an error message similar to the one shown in Figure 12. Look for *You are not authorized to view this page*. See Sending Passwords to WIP Edit on page 28 for additional information.
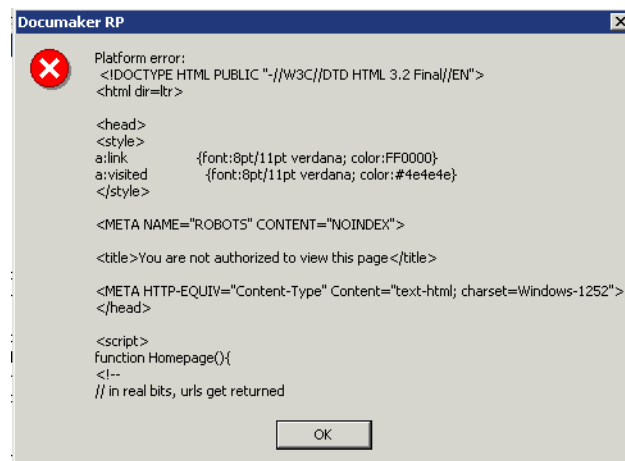
Figure 12: Authentication error message

DEBUGGING TIP 1

If you look at the windows temporary directory while the WIP Edit ActiveX control has a document open you can find the directory where WIP Edit stores the individual files that make up the DPW file on the client machine.

You will always have a FSIUSER.INI file in this directory regardless of whether the file was downloaded from IDS. The WIP Edit control always creates a FSIUSER.INi file and you can view this to determine the behavior of the control.

The temp directory that is used can be located by looking at some environment variables on the client machine. If you have an environment variable called *WIPEDITTMP* defined, the WIP Edit control will use that directory to store temporary files. If that variable is not defined, look for the path defined by the TMP environment variable.

Using Windows Task Manager, look for the process ID associated with the wipedw32.exe program. The format for the temp directory name is as follows:

```
Temp Dir defined by Env variable + tmpform + process ID
```

So if you have TMP set to *C:\winnt\temp* and the process ID is 872, the temp WIP Edit directory on the client machine will be *C:\Winnt\temp\tmpform872*. If you look in this directory, you will find the individual files that were shipped in the DPW file from IDS.

DEBUGGING TIP 2

When the WIP Edit ActiveX control attempts to download resources from IDS and if fails to receive one, you will get an error message similar to the one shown in Figure 13.
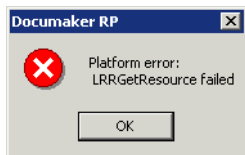
Figure 13: Platform Error Resource Not Found

On the IDS side you will see the following error.

```
**************************************************
DPRGetResource <1> <1> FAPLoadImage failed (4column)
**************************************************
**************************************************
ATCSendFile <1> <1> Unable to ATCLocateValue for <RETURNFILE>.
**************************************************
```

The following list identifies the types of files that can be downloaded dynamically:

• FAP (default location <Mstrres><FormLib>)

• DAL scripts (default location <Mstrres> <Deflib>)

• Tables (default location <Mstrres> <Tablelib>)

• Helps (default location <Mstrres> <Helplib>)

• FOR, GRP, BDF

If you see an error, check the MRL INI file for the values set in GETSCRIPT and PUTURL. If either of these is incorrect, the dynamic resource download will not work. The control group in the MRL INI file looks like this:

```
< INI2XML >
   GetScript = /doc-prog/iwip/amergen/wipdownload.asp
   PutURL = localhost
```

HANDLING ERROR MESSAGES

This topic discusses the various errors that can occur. Each subsection identifies the error classification, the error message, and a resolution on what to do about it.

CLIENT SIDE ERRORS

Errors when opening a document

These errors can occur when opening a document:

```
INIMakeContext failed
```

If you see this error, get the latest patch level and reinstall WIP Edit.

```
Could not parse DPW file
```

See the item in the debugging section.

```
Could not Set INI file
```

If you see this error, get the latest patch level and reinstall WIP Edit.

```
Could not RACInitCtrl
```

If you see this error, get the latest patch level and reinstall WIP Edit.

Errors if the temp directory can't be removed

The following error messages can appear if you cannot remove the tmpform(pid) directory where the DPW file contents have been extracted. The message appears in a message box when you try to navigate the browser to another page.

```
dir not empty
invalid path
Unknown Error removing directory
```

The corrective action is to manually delete the directory.

Problem with the installation

The following error messages can only occur if there is something wrong with the installation. These will appear in a message box. Remove the current WIP Edit Control installation and reinstall.

```
LoadRacDLL failed
could not get racw32.dll
could not get RACSetCmdLineArg
could not get RACThread
could not get RACGetClientWindow
could not get RACTerminate
could not get RACGetAccelHandle
could not get RACSave
could not get RACSetWndProc2Close
could not get RACSetParent
could not get RACSetWhatSystem
could not get RACSetAfeHab
could not get RACRetrieveFile
could not get RACLoadMenu
could not get RACEnableMenu
could not get RACSubClass
could not get RACLoadMenuToolbar
could not get RACWorkingPath
could not get RACSetMenuFile
could not get RACUpdateWipFile
could not get RACSetIniFile
could not get RACInitCtrl
```

```
could not get RACSetWorkingPath
could not get RACBreakDpwFile
could not get RACRemoveTmpFiles
could not get RACSetCmdLineArg
could not get RACCommand
could not get RACPostCommand
could not get RACSaveDpwFile
could not get RACClose
could not get RACLoadForms
could not get RACGetWipField
could not get RACSetWorkingPath
could not get RACSetWipField
```

SERVER SIDE ERRORS

When opening a DPW file

When IDS produces an error, the web page receive san error code. Figure 14 shows an example of a web page error generated by IDS.
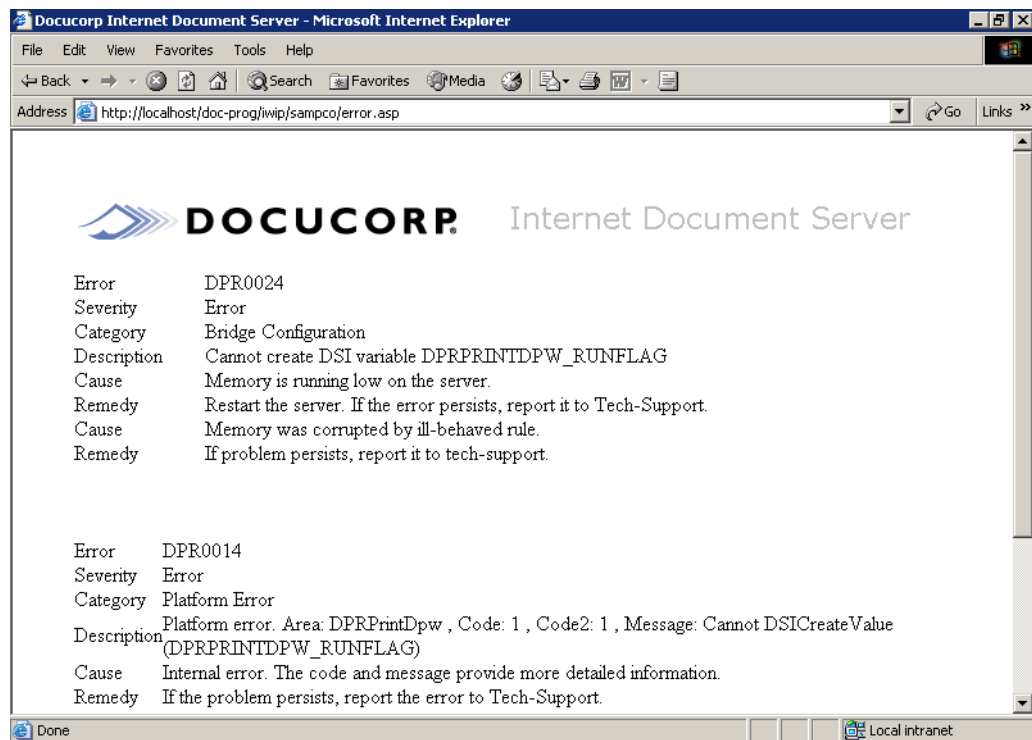
Figure 14: Web Page Error on File Open

IDS print errors

The DPRPrintDpw rule can create the following errors. This is how you will see the errors in the DOS window if you are running IDS from a command line.

```
DPRPrintDpw <1> <1> Cannot DSICreateValue(DPRPRINTDPW_RUNFLAG)
DPRPrintDPW <1> <1> VMMCreateList failed
DPRPrintDPW <1> <1> RF_POSTFILE cannot be created
```

If the error reoccurs, restart IDS. You should also contact Oracle Insurance Support and provide a copy of your log files (DPRTRC.LOG and DSRVTRC.LOG).

Failure to create a DPW file

If you see the following error on the console and in the error handling web page, you have failed to create the DPW file.

```
DPR0039  LOCATION DPRPrintDpw APINAME DPRPrint failed.
```

See the following link for more information on the DPRPrint rule.

https://support.docucorp.com/doss/document/idsdoc/Int018/V110sofeat.pdf

Cannot open WIP database

If you see the following error on the console screen and in the error handling web page you cannot open the WIP database. Check the INI options for WIP.

```
DPR0039  LOCATION  DPRPrintDpw APINAME WIPOpenWIP failed.
Could not open the WIP database. Check INI options for WIP.
```

Name or value errors

The following attachment variables are used by iPPS to add information to the save request for WIP Edit.

```
HTTPHEADER group puts in the information in an HTTP header.
HTTPHEADER = "3"
HTTPHEADER1.NAME = "someheader1"
HTTPHEADER1.VALUE = "someVALUE1"
HTTPHEADER2.NAME = "someheader2"
HTTPHEADER2.VALUE = "someVALUE2"
HTTPHEADER3.NAME = "someheader3"
HTTPHEADER3.VALUE = "someVALUE3"
HTTPFORMDATA group post the information as form data.
HTTPFORMDATA =  "3"
HTTPFORMDATA1.NAME = "nameformdata1"
HTTPFORMDATA1.VALUE = "valueformdata1"
HTTPFORMDATA2.NAME = "nameformdata2"
HTTPFORMDATA2.VALUE = "valueformdata2"
HTTPFORMDATA3.NAME = "nameformdata3"
HTTPFORMDATA3.VALUE = "valueformdata3"
HTTPQUERYSTRING group puts the information in the query string
HTTPQUERYSTRING = "1"
HTTPQUERYSTRING1.NAME = "SESSIONID"
HTTPQUERYSTRING1.VALUE = "222"
HTTCOOkIE puts the information in the HTTP cookie header.
HTTPCOOkIE = "1"
HTTPCOOkIE1.NAME = "cookie"
HTTPCOOkIE1.VALUE = "Toolfloat=false; Toolbottom=false;
IX=%E9%C4%92%08%C9%D3xo%D9%2D%AF%D3%A0%AC%26%15%7E%FA%23M%01%D9%FDt
%23%A2%13%7E%CAN%95%80%B2%E5cC%0Enj%E7%1E%E4;
ASPSESSIONIDACRTQSCA=EGPPLAECPNDMGOIIMANOAPPB"
```

If the server expects another variable present and cannot find it you will see the following error on the IDS console screen.

```
Could not find HTTPCOOkIE#.NAME in attachment
Could not find HTTPCOOkIE#.VALUE in attachment
```

This is probably due to a problem in the web page that request the DPW file (such as WIPEDIT.ASP or WIPEDIT.JSP).

Resource error when building a DPW file

If the IDS cannot access a resource when building the DPW file you will see the error shown in Figure 15.
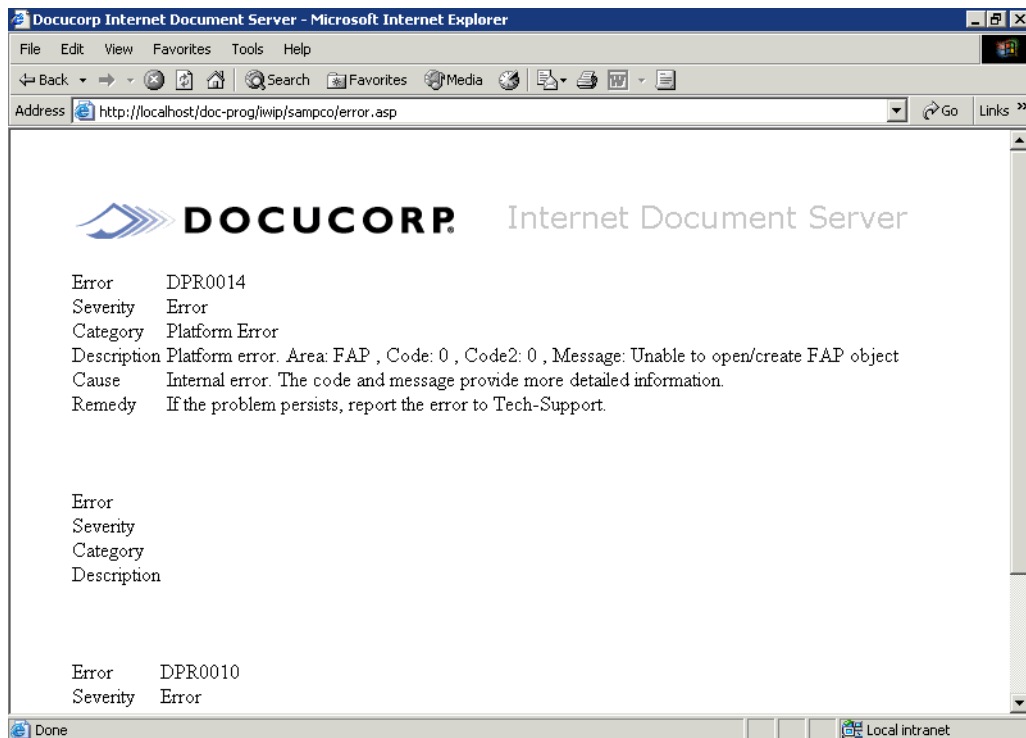
Figure 15: Resource Error

You will see the following error on the IDS console screen.

```
DocuCorp International Internet Server ready!
Version 100.018.

DocuMaker shared objects version 400.111
**************************************************
FAP <0> <0> Unable to open/create FAP object
**************************************************
```

INTEGRATING CUSTOM CODE

Installing Custom DLLs

If you write a custom ASP web application that uses a COM object that is not registered as a COM+ object under Component Services, you may run into a problem. An example of this is the ASP version of Print Preview, which uses the IDSASP.dll COM object. The problem occurs when you use the default account name of *IWAM_boxname*, where *boxname* is the name of the machine you are running on.

If you are using a message queuing system such as MQ Series or WebSphere MQ, you will need to make sure that the user ID (*IWAM_boxname*) is part of the mqm group on the machine where MQ Series is installed. If MQ Series is installed on a different machine than IIS, you will have to create the IWAM_boxname account on the MQ Series machine and provide the same password used to create the account on the IIS machine.

The account is created during the IIS installation and you won't know directly what the password is. So you have to reset it using the Security tab of the virtual directory. You will also have to reset it on the Identity tab of the appropriate node under Component Services.

When you run com objects that are not registered as COM+ in Component Services, they run under the identity of the *IIS-In Process applications* node or under the identity of the *IIS Out-Of-Process pooled applications* node, depending on how your virtual directory was configured to use application protection. See the virtual directory tab of the virtual directory after right-clicking on it and selecting *properties* (see the Application protection drop-down option).

```
low = in process
medium, pooled = out of process pooled
```

Alternatively, you could register the COM objects used by the application under a new node in Component Services and configure the identity you want to use there, then make sure that identity has the appropriate access rights to MQ Series.

USING THE PRINT PREVIEW APPLICATION

You can use WIP Edit without iPPS to provide a subset of functionality. WIP Edit can communicate with a mid-tier server that connects to the IDS application to retrieve WIP documents. You can preview these documents, edit them, or print them locally in PDF format using Adobe's Acrobat Reader. In this mode, you can do everything WIP Edit allows, except create a new transaction.

Oracle Insurance provides a sample web application that can be used to try the WIP Edit print preview features. The sample application is called *Sampco*. Setting Up Print Preview with Tomcat as the Mid-tier on page 88 and Setting Up Print Preview with WebSphere 5.1 as the Mid-tier on page 100 shows how to deploy the Sampco print preview web application using Tomcat as the mid-tier server, and using IBM's WebSphere 5.1 as an application server.

The details of how to configure and deploy the sample application are left to the appendices. If you want to have a group of users with only the Print Preview capability, you can use the Sampco web application as a starting point to customize your own application.

NOTE: If you *are not* using iDocumaker 3.2 (J2EE) without customizations, but are using it as an integration point for WIP Edit or any other feature, make sure your integration point properly handles application-specific session data.

If multiple invocations to this integration point use the same application session, make sure old session data is removed at the start of the integration point.

As a best practice, model your integration point after an existing integration point by duplicating and renaming an existing application integration point. For more information about integration with iDocumaker 3.2 (J2EE) talk with a Services Professional.

Appendix A

Setting Up Print Preview with Tomcat as the Mid-tier

This appendix shows you how to install and configure the Tomcat servlet container on Linux. In general, there is nothing in this appendix that is specific to Linux. What is discussed here should apply to any UNIX environment.

This appendix covers these topics:

OVERVIEW

The mid-tier for this document is hosted on a VMWare workstation Linux machine running Red Hat 9.0. You will need access to the root account. The Tomcat software is built for specific version of the Java virtual machine. Your Linux/UNIX machine needs to have a JVM installed and the JAVA_HOME environment variable set to the top level JVM directory. Different versions of Tomcat use different JVMs. For example, Tomcat 5.0.28 uses Java 1.4 SE, while Tomcat 5.5.x uses Java 1.5 SE. So make sure you get the correct JVM for the version you are installing. You can get a JVM at this web site:

http://www.javasoft.com

Once you log onto the Linux machine as root, make a subdirectory in your home directory called *tomcat*. If your graphical environment desktop is not running, enter **startx**. This lets you use your browser to get the Tomcat software. Download the Tomcat software from the Apache web site and save it to the tomcat subdirectory you created. The Jakarta Tomcat servlet container can be downloaded from this web site:

http://jakarta.apache.org/tomcat/index.html

The Tomcat software is available as a gzip compress tar file. For this appendix, Tomcat version 5.0.28 was used. The download file name is shown here:

```
jakarta-tomcat-5.0.28.tar.gz
```

The file name contains the version information. Once the download has completed, go to the tomcat subdirectory and enter this command:

```
zcat  jakarta-tomcat-5.0.28.tar.gz | tar  xvf  -
```

The zcat program runs the file through the gunzip decompression program. The pipe symbol (|) sends the result to the next command; in this case, the tar command. The dash character at the end of the line is important. It specifies that the piped result is used as the input to the tar command.

The result of this command is a fully extracted Tomcat installation in a subdirectory called *jakarta-tomcat-5.0.28*. Depending on the version you downloaded, the name may differ. In general, you want to put this directory some place other than in root's home tree. Linux distributions like to have 3rd party software installed on the /opt partition. Unless you manually sized this partition during your Red Hat install, however, it is too small. There are ways to create a larger partition, but that information is not covered in this manual.

In the absence of a suitable /opt partition, you can put this directory structure in the /usr/local subdirectory. Red Hat provides enough space in the /usr partition. To move the entire Tomcat directory, enter the following command from the tomcat directory:

```
mv  jakarta-tomcat-5.0.28  /usr/local
```

Now change to that directory ('cd /usr/local/Jakarta-tomcat-5.0.28'). These tasks must be done within this directory structure to get the iWIP software to work:

1 Install the JSP and ASP files in the proper web application directory.

2 Copy the jar files needed for the application into the correct directory.

3 Copy a properties file to the proper directory

4 Create a script that sets the Tomcat environment variables if one is not present.

5 Edit the Tomcat startup script. At the appropriate points, we will specify the properties file to use.

INSTALLING THE JSP AND ASP FILES

Figure 16 shows the iWIP application directory on the back-end Docserv machine (Docserv\html\iWIP). This directory (and all of the subdirectories) contains the WIP Edit application JSP and ASP files. Copy this directory and all of the subdirectories to the Linux machine. One easy way to collect these files is to zip them (recursively) on the Docserv machine, then move them to the Linux machine. Once the ZIP file is there, you can unzip them into the proper place.

NOTE: On a UNIX install of IDS, you have to run the setupsdk.xxx file. This creates a resource directory in the location where you run this file. The JSP files are then located in:

```
resource\clientjsp\html\iwip
```

To create a place to put these files, change directories to

```
cd  /usr/local/Jakarta-tomcat-5.0.28/webapps
```

If you created a recursive ZIP file on the Docserv machine, copy the ZIP file to this directory (the webapps directory). Then unzip the file. It should create an iWIP directory and put all the files under it. Look inside the iWIP directory to make sure your files made it.

That completes the installation of the JSP and ASP files and their related sub directories.
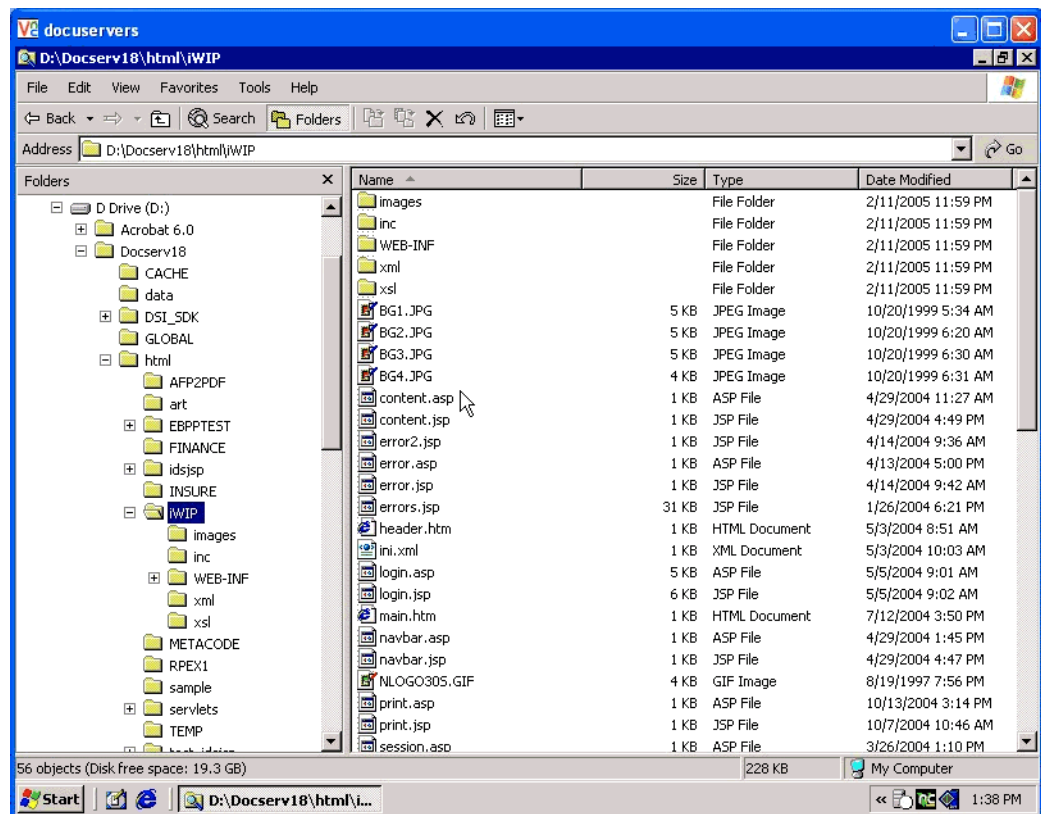
Figure 16: Docserv iWIP Directory

COPYING THE JAR FILES

Figure 17 shows the Docserv directory where the WIP Edit jar files are located. These can be found under this directory:

```
Docserv\DSI_SDK\JAVA\jars
```

With one exception, you will want to copy all of these jar files over to the Linux machine. The one exception is a choice between DSIJAVA.jar and DSIJAVAMsg.jar. Both of these files do perform the same function. One uses Java and the other uses C++/JNI. You only need to use one of these files. If both are in the directory, the application will not work.

The easy way to move these files is to zip them up and copy them over to the Linux machine. Unzip the jar files into:

```
/usr/local/Jakarta-tomcat-5.0.28/common/lib
```

NOTE: On a UNIX install, the jar files are located in resource/DSI_SDK/JAVA/jars.

Once these jar files are in that location, this aspect of the configuration is complete.

Figure 17: Docserv iWIP Jar Files Directory

COPYING THE PROPERTIES FILE

Figure 18 shows the location of the dsimsgclient.properties file. This file must be copied over to the Linux machine and placed in the following directory:
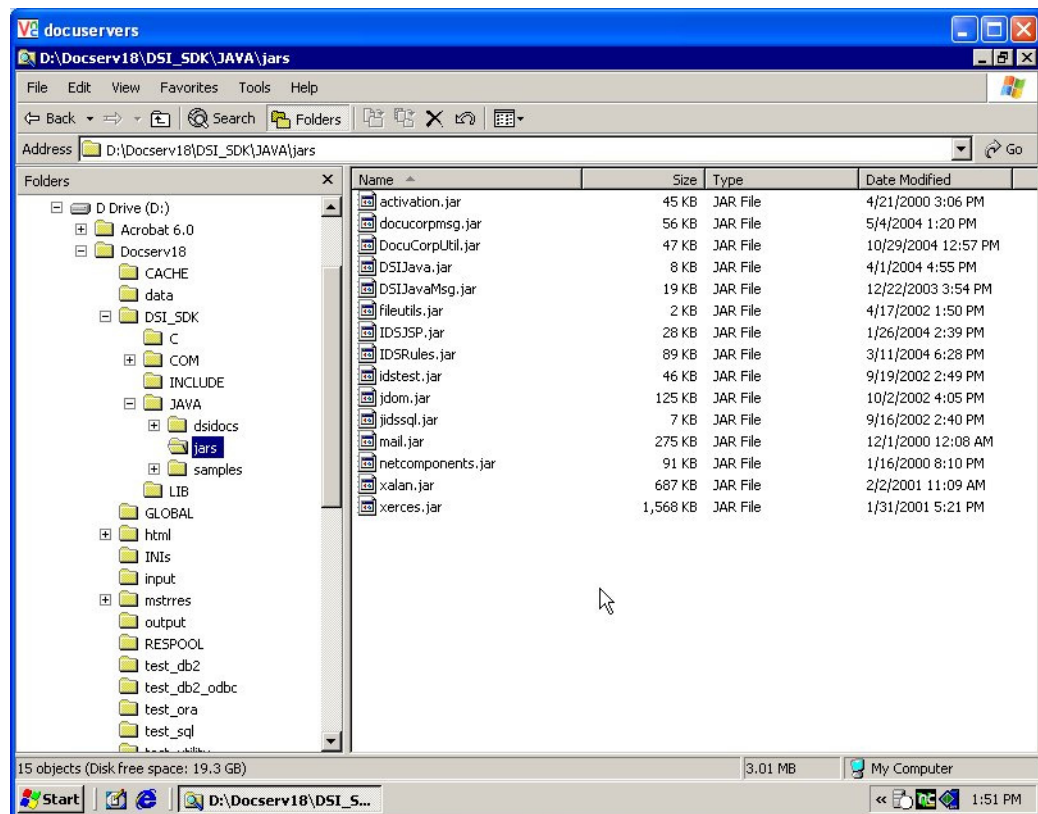
```
/usr/local/Jakarta-tomcat-5.0.28/bin
```

Next, edit the Tomcat startup file to point to this property file.
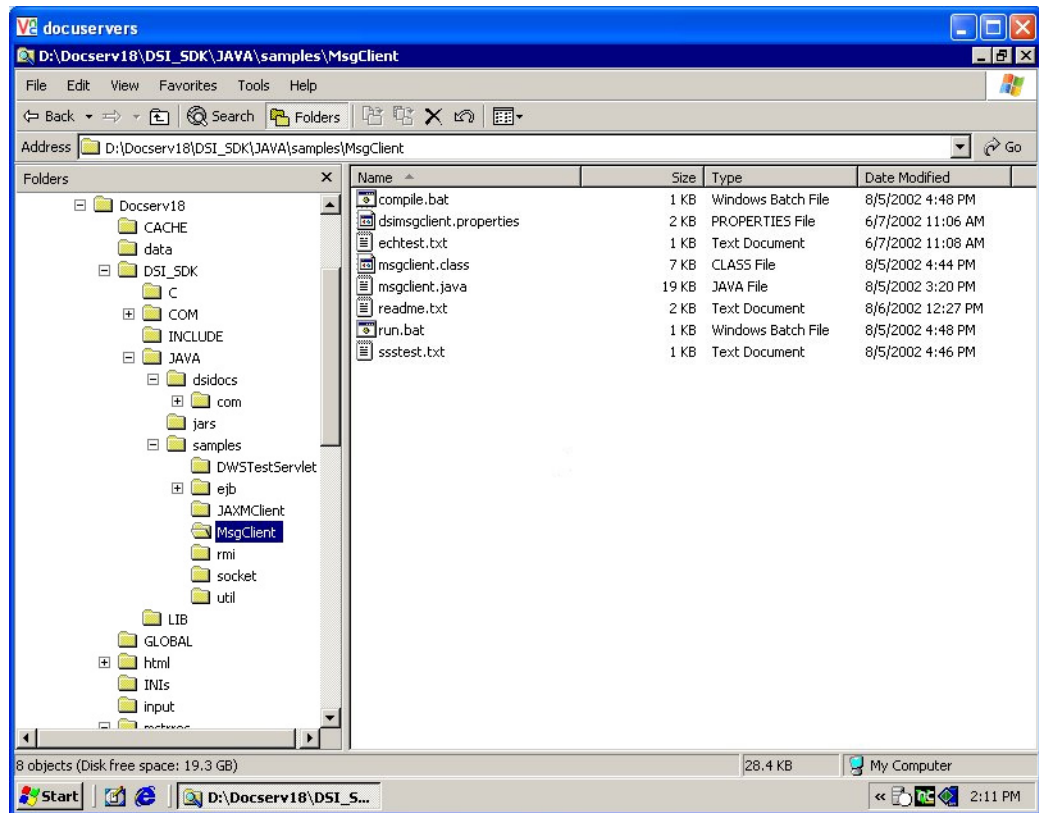
Figure 18: dsimsgclient.properties file location

CREATING SCRIPTS TO SET ENVIRONMENT VARIABLES

Several environment variables must be set for Tomcat to work properly. The easiest way to do this is to create a script file that sets them. On the Linux machine, change directories to:

```
/usr/local/jakarta-tomcat-5.0.28/bin
```

Check to see if a file called *setenv.sh* exists in the directory. If it doesn't, using a Linux/ UNIX editor, create the file and add these lines to it:

```
#!/bin/sh
TOMCAT_HOME=/usr/local/jakarta-tomcat-5.0.28
CATALINA_HOME=/usr/local/jakarta-tomcat-5.0.28
J2SE_HOME=$JAVA_HOME
export TOMCAT_HOME  CATALINA_HOME  J2SE_HOME
```

The first line is specific to the shell that is running on the Linux machine. In this case, the Bash shell. If you are running on Solaris or AIX, you will have to change this line to reflect your operating system's way of identifying the shell in script files. The $JAVA_HOME variable should have been set up when you installed your JVM.

You will not have to manually run this file. You can edit the Tomcat startup script to do this automatically. However, you will have to change its mode to executable. Enter the following on the command line:

```
chmod 755 setenv.sh
```

This makes the file read/write/executable for root, and read/executable for everyone else.

EDITING THE TOMCAT STARTUP SCRIPT

Tomcat ships with a startup script file in the bin directory, named *catalina.sh*. This file is called by the startup.sh script. Edit this file to make sure:

- The dsimsgclient.properties file location is correctly identified.

- Tomcat starts with the proper option to use the dsimsgclient.properties file.

Editing the dsimsgclient.properties File

The dsimsgclient.properties file contains information about the location and names of the queues you have defined. This information is read by the JSP scripts on the mid-tier. When a request is posted to the mid-tier, it posts a message on the request queue. If you do not have this file properly configured or if it is missing, when the client side makes a request a timeout interval will occur. The timeout message tells you the mid-tier could not be reached and timed out. It also instructs you to make sure IDS is running.

The following shows how the dsimsgclient.properties file was configured for WebSphere MQ. You will need to replace the entries for your specific system information. You will need to edit the following entries:

Option	Set to the
mq.queue.manager	Name of your queue manager.
mq.queue.channel	Name of your channel. In the listing below, the default channel is used.
mq.tcpip.host	IP address of the machine that is running your queue manager.
mq.tcpip.port	Port your queue manager is using.
mq.outputqueue.name	Request queue name that you defined within your queue manager.
mq.inputqueue.name	Response queue name that you defined within your queue manager.

The remaining options can remain set to their default values.

```
# The class that implements queuing, in this case MQSeries
queuefactory.class=com.docucorp.messaging.mqseries.DSIMQMessageQueu
eFactory
# The class that formats the DSIMessage to send to a IDS server.
Uncomment
# the following line to communicate with IDS version 1.6, leave
commented for
# subsequent versions.
#marshaller.class=com.docucorp.messaging.data.marshaller.LegacyByte
ArrayDSIMessageMarshaller

# Here are some sample MQSeries parameters
# Queue manager for system hosting MQSeries
mq.queue.manager=IDSQM

# MQSeries channel that the messaging client and IDS server
communicate through
mq.queue.channel=SYSTEM.DEF.SVRCONN
```

```
# MQSeries communication can be either in 'bindings' mode (the
program is running
# on the same machine as the MQSeries server) or 'client' mode (the
program is
# running on a different machine and communicates with the MQSeries
server through
# TCP/IP). If the setting 'mq.tcpip.host' is defined then we use
client mode else
# we use bindings mode.
mq.tcpip.host=10.1.13.185

# TCP/IP port that the MQSeries server is listening to, 1414 is most
commonly used.
mq.tcpip.port=1414

# A client program sends requests out and gets results in
mq.outputqueue.name=REQ1
mq.inputqueue.name=RES1

# How long, in seconds, that the MQSeries Server will keep a message
in the queue
# if a program doesn't get it.
mq.outputqueue.expiry=120
```

Listing 23: The dsimsgclient.properties File

SETTING THE LOCATION OF THE DSIMSGCLIENT.PROPERTIES FILE

Once again, using your favorite editor, load up the catalina.sh file located in the Tomcat bin directory. Scroll down through the file until you see where the CATALINA_HOME environment variable is set. The script code will look like this:

```
# Only set CATALINA_HOME if not already set
[ -z "$CATALINA_HOME" ] && CATALINA_HOME=`cd "$PRGDIR/.." ; pwd`

if [ -r "$CATALINA_HOME"/bin/setenv.sh ]; then
    . "$CATALINA_HOME"/bin/setenv.sh
fi
```

This code...

- Makes sure the CATALINA_HOME environment variable is set. (We actually override this in our setenv.sh file.) If not, it sets it for us.

- Checks to see if the setenv.sh file is in the bin directory. If it is, it will automatically be run.

NOTE: The line inside the IF statement is difficult to read, but there is one key thing to remember in the Bash shell: to execute a script file, you enter a period (.) followed by a space, followed by the script file name. If you try to run the script with ./setenv.sh (dot/setenv.sh) it would not run in the Bash shell (even after you have made it executable). You can verify this by running it with the dot/ method first and typing **env | less** on the command line to see if the variables actually got set.

Right after this block, we can be assured the CATALINA_HOME environment variable is set. So now we can define the location of the dsimsgclient.properties file. Right after the block of code discussed above, add the following:

```
PROPSFILE=$CATALINA_HOME/bin/dsimsgclient.properties
export PROPSFILE
```

This defines a variable within the catalina.sh file that will be used to set the location of the properties file.

Making Tomcat Use the Properties File at Startup

This is the hardest part of the Tomcat configuration. The reason is there are several blocks of code within the script that can start the servlet container. Which block of code executes depends on the logic of the various IF blocks. Therefore, the startup option that tells Tomcat to use the properties file must be put in various places.

If you scroll through the catalina.sh script, you will see blocks of code that resemble this:

```
        if [ "$1" = "-security" ] ; then
         echo "Using Security Manager"
         shift
         exec "$_RUNJDB" $JAVA_OPTS $CATALINA_OPTS \
            -Djava.endorsed.dirs="$JAVA_ENDORSED_DIRS" -classpath
"$CLASSPATH" \
```

```
                -sourcepath "$CATALINA_HOME"/../../jakarta-tomcat-catalina/
catalina/src/share \
            -Djava.security.manager \
        -Djava.security.policy=="$CATALINA_BASE"/conf/catalina.policy
\
            -Dcatalina.base="$CATALINA_BASE" \
            -Dcatalina.home="$CATALINA_HOME" \
            -Djava.io.tmpdir="$CATALINA_TMPDIR" \
            org.apache.catalina.startup.Bootstrap "$@" start
        else
          exec "$_RUNJDB" $JAVA_OPTS $CATALINA_OPTS \
            -Djava.endorsed.dirs="$JAVA_ENDORSED_DIRS" -classpath
"$CLASSPATH" \
            -sourcepath "$CATALINA_HOME"/../../jakarta-tomcat-catalina/
catalina/src/share \
            -Dcatalina.base="$CATALINA_BASE" \
            -Dcatalina.home="$CATALINA_HOME" \
            -Djava.io.tmpdir="$CATALINA_TMPDIR" \
            org.apache.catalina.startup.Bootstrap "$@" start
        fi
```

Based on the logic tested, different blocks can execute. The options on the command line all begin with *-Doption_name*. The last part of each block starts Tomcat:

```
"org.apache.catalina.startup.Bootstrap "$@" start"
```

Just before that line in each block, add the following:

```
                -Ddsimessage.properties="$PROPSFILE" \
```

So the modified block of code would look like this:

```
        if [ "$1" = "-security" ] ; then
          echo "Using Security Manager"
          shift
          exec "$_RUNJDB" $JAVA_OPTS $CATALINA_OPTS \
            -Djava.endorsed.dirs="$JAVA_ENDORSED_DIRS" -classpath
"$CLASSPATH" \
            -sourcepath "$CATALINA_HOME"/../../jakarta-tomcat-catalina/
catalina/src/share \
            -Djava.security.manager \
        -Djava.security.policy=="$CATALINA_BASE"/conf/catalina.policy
\
            -Dcatalina.base="$CATALINA_BASE" \
            -Dcatalina.home="$CATALINA_HOME" \
            -Djava.io.tmpdir="$CATALINA_TMPDIR" \
            -Ddsimessage.properties="$PROPSFILE" \
            org.apache.catalina.startup.Bootstrap "$@" start
        else
          exec "$_RUNJDB" $JAVA_OPTS $CATALINA_OPTS \
            -Djava.endorsed.dirs="$JAVA_ENDORSED_DIRS" -classpath
"$CLASSPATH" \
            -sourcepath "$CATALINA_HOME"/../../jakarta-tomcat-catalina/
catalina/src/share \
            -Dcatalina.base="$CATALINA_BASE" \
            -Dcatalina.home="$CATALINA_HOME" \
            -Djava.io.tmpdir="$CATALINA_TMPDIR" \
            -Ddsimessage.properties="$PROPSFILE" \
            org.apache.catalina.startup.Bootstrap "$@" start
        fi
```

Make sure you go through the entire file and put this line in each block that can execute the code. In the version of Tomcat used in this example, that line was inserted in seven locations. Once you finish editing the file, save it and exit to the command line.

STARTING
TOMCAT

To start Tomcat, make sure you are in the /usr/local/jakarta-tomcat-5.0.28/bin directory and type the following:

```
startup.sh
```
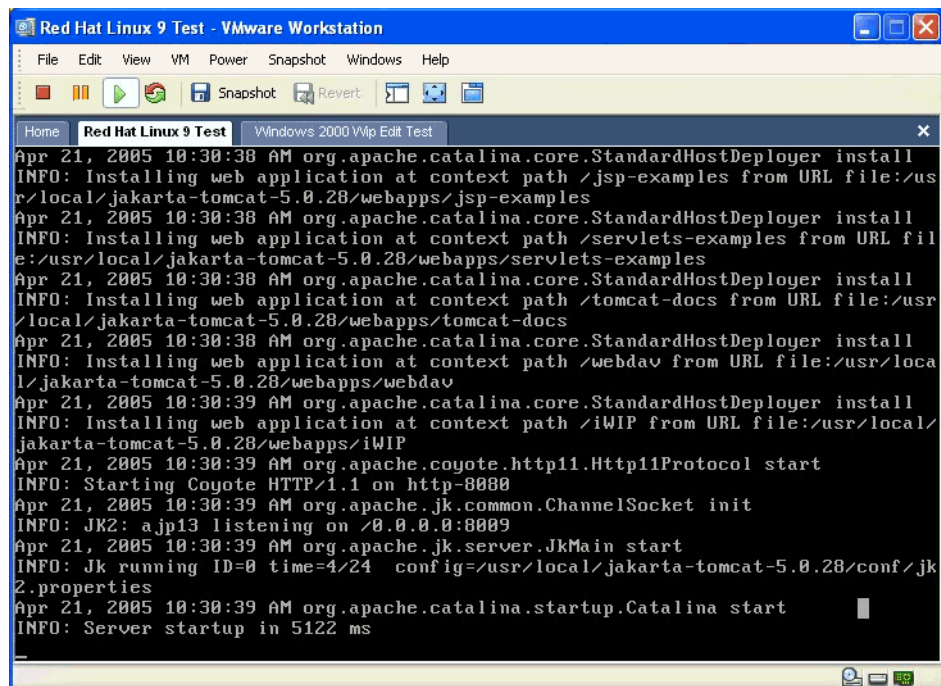
Tomcat will start and begin to write information to the console window (STDOUT), shown below. You will see lines that begin with *INFO*. Look for a line that says something like this:

```
INFO:  Installing web application at context path /iWIP from URL
file:/usr/local/jakarta-tomcat-5.0.28/webapps/iWIP
```

This shows that Tomcat successfully loaded the iWIP web application. At the bottom of the window you will see something like:

```
INFO:   Server startup in 5122 ms
```

This indicates that Tomcat is ready and waiting for a connection. If you get these kinds of results, you have successfully deployed the iWIP web application to your servlet container.

Figure 19: Tomcat Startup

Setting Up Print Preview with WebSphere 5.1 as the Mid-tier

Instead of using the Tomcat server on Linux (or Windows), another alternative is to use the WebSphere application server. For this appendix, WebSphere 5.1 Application Server (WAS) was installed on a Windows 2000 machine. IBM can change the user interface to the Administration Console between version numbers. So if you are deploying to a different, the screen examples may be different. However, the process describe should be similar between versions.

This appendix guides you through the steps necessary to deploy the mid-tier on WebSphere. It is more complicated than Tomcat, so there are screen examples to guide you through the process. For each screen examples, there will be a comment telling you what to do on that screen.

This appendix includes these topics:

CREATING THE WAR FILE

To deploy an application to WAS you have to have the application packaged in a WAR file or an EAR file. WAR is defined as Web Application Archive file, and EAR is Enterprise Application Archive. These files have a *jar* file extension. Essentially, they are like ZIP files and can have a hierarchy inside them. One of the important directories that appear inside one of these files is WEB-INF. This directory holds information about the web application.

In the Tomcat discussion in this document, we found the Sampco web application JSP files (and other related things) on IDS as shown in . For Tomcat, we made a ZIP file, copied that over to the mid-tier machine, created an iWIP directory under the webapps directory, and unzipped the files there. The process will be different for a WAS deployment.

The first step is to create a WAR file with the web application. To do that, follow these steps.

1   On the back-end server, located the iWIP directory under the Docserv\html\iWIP. As in the case of the Tomcat example, zip these files up into *iWIP.ZIP* (make sure you do this recursively and grab all the contents under this directory).

2   Copy the iWIP.ZIP file to your WebSphere box. Make a working directory, like c:\temp\iWIP, and unzip this file in that location.

3   Look under the iWIP directory and you should see a directory called WEB-INF. If it does not have a subdirectory called *lib*, create that directory.

4   On the back-end server, open a copy of Explorer and find the jar files as shown in . Make a ZIP file of these jar files and copy that to the WebSphere box temporary directory you created in step 2.

5   Unzip the jar files into the iWIP\WEB-INF\lib directory.

6   On the WebSphere server, open a console (DOS) window and change directories to:

```
C:\temp\iWIP
```

This assumes you created your temporary directory in that location. If not, change to where you unzipped the iWIP.ZIP file.

7   Make sure that the Java/bin directory is in your path. You can set this in My Computer, Environment Variables.

8   From within the iWIP directory (make sure you see WEB-INF in that directory), type the following:

```
jar cvf ../iWIP.war *
```

The jar utility ships with the Java Runtime Environment. It is very similar to the ZIP utility, however, the command line switches like *cvf* are taken from the UNIX tar utility program. When you issue this command you are telling the jar utility to create a file in the directory above where you are sitting and call it *iWIP.war*. The asterisk (*) at the end of the line tells it to gather everything in the current directory and all subdirectories and include them in the WAR file. If you get errors when you run this command, you probably don't have the Java/bin directory set correctly in your path.

9   Once the iWIP.war file is created, copy it to the following directory:

```
C:\Program Files\WebSphere\AppServer\InstallableApps
```

This directory is a staging area for applications that you install into WebSphere.

10 Now locate the dsimsgclient.properties file on the back-end server, as shown in .

11 Make a directory on the WAS machine called Sampco and copy this file into that location. Note: It is recommended that you make this directory directly under the root of one of your drives. Also, it is recommended that you avoid putting spaces in directory and names.

12 If you haven't edited the dsimsgclient.properties file to point to your queue manager, follow the instructions in

Now you are ready to deploy the application using the WAS Administration console.

DEPLOYING THE WAR FILE WITH THE WAS ADMINISTRATIVE CONSOLE

On your WebSphere 5.1 server, bring up a browser window and type in this URL:

```
http://localhost:9090/admin
```

NOTE: Your administrator can change ports during the install. If the default installation is performed, port 9090 will be assigned to the Administrative Console.

You will be presented with a login screen as shown in Figure 20. Unless your administrator has setup the security server, or has required logins via LDAP, operating system, or some other method of authentication, you can enter anything you want in the user ID edit control. Try logging in as *admin*. If that doesn't work, contact your administrator to find out the login ID you need to use.
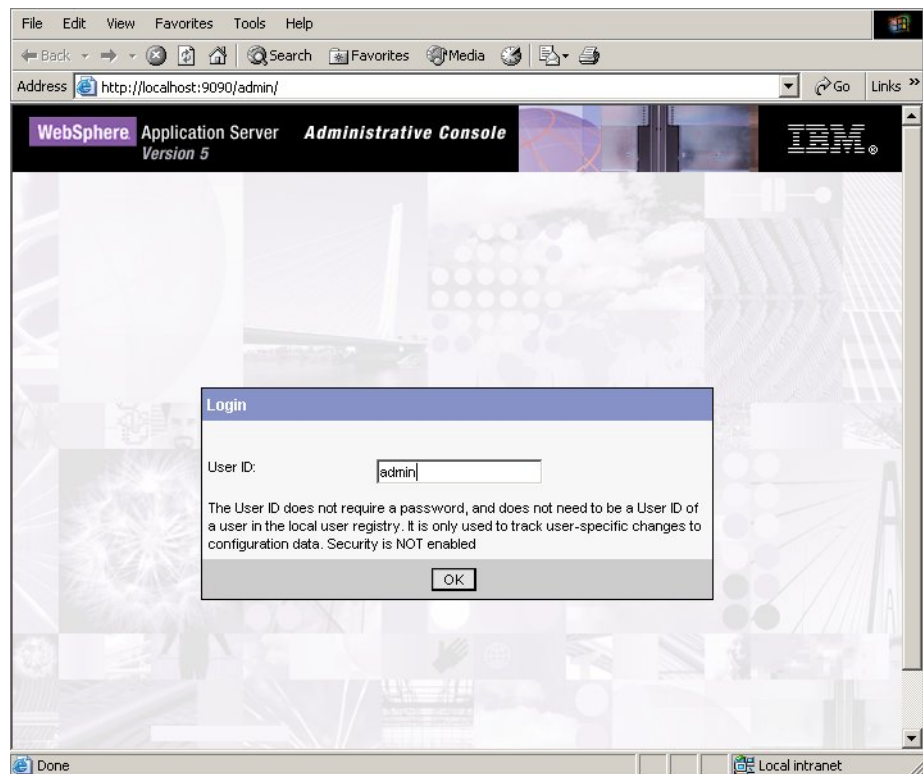
Figure 20: WAS Login Screen

If your login ID is valid, you will see the WAS main screen, as shown in Figure 21. On the left hand side, there are dynamic menu items. Click the Applications item to expand the menu tree. You will see an item called *Install new Application*. Click on that and the screen shown in Figure 1.22 on page 105 appears.
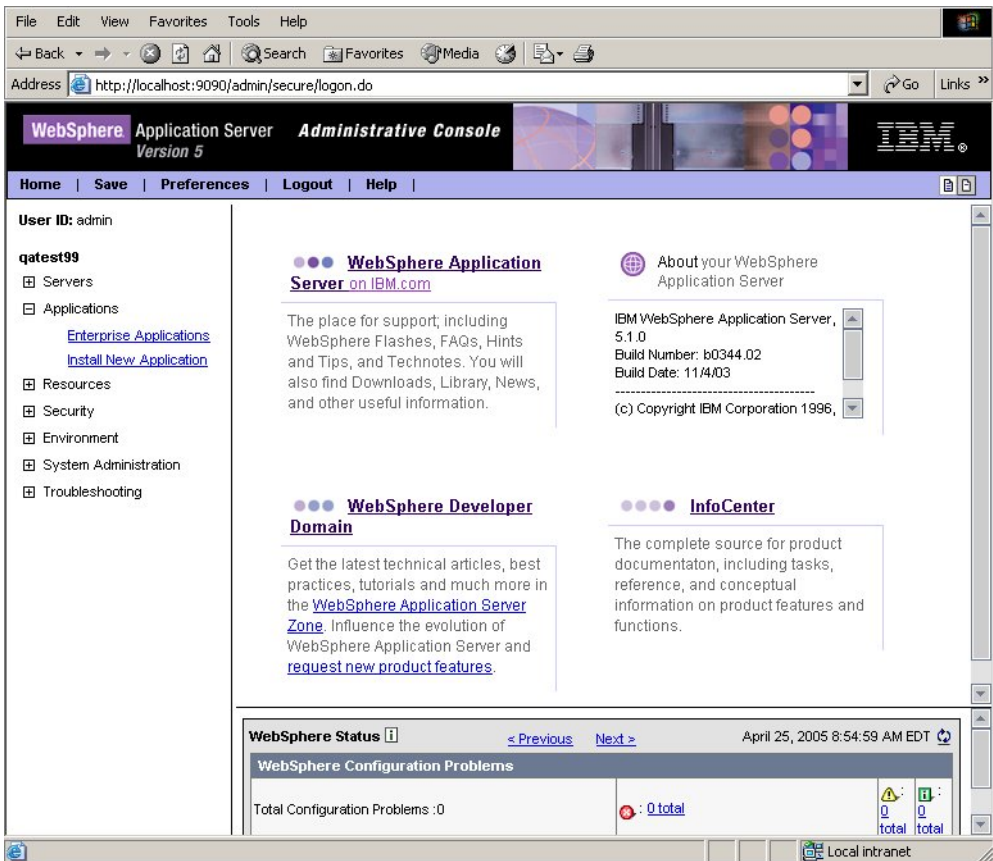
Figure 21: WAS Main Screen

In Figure 22, you enter the path where the iWIP.war file is located. In this case, we copied it to the following directory in the previous topic:

```
C:\Program Files\WebSphere\AppServer\InstallableApps
```

You can click Browse to set this path.

The other setting is for the *context root*. This tells WebSphere the name you will use in the URL to identify this application. The URL to access the iWIP application is shown in Figure 22. Enter **iWIP** in the Context Root field and click Next.
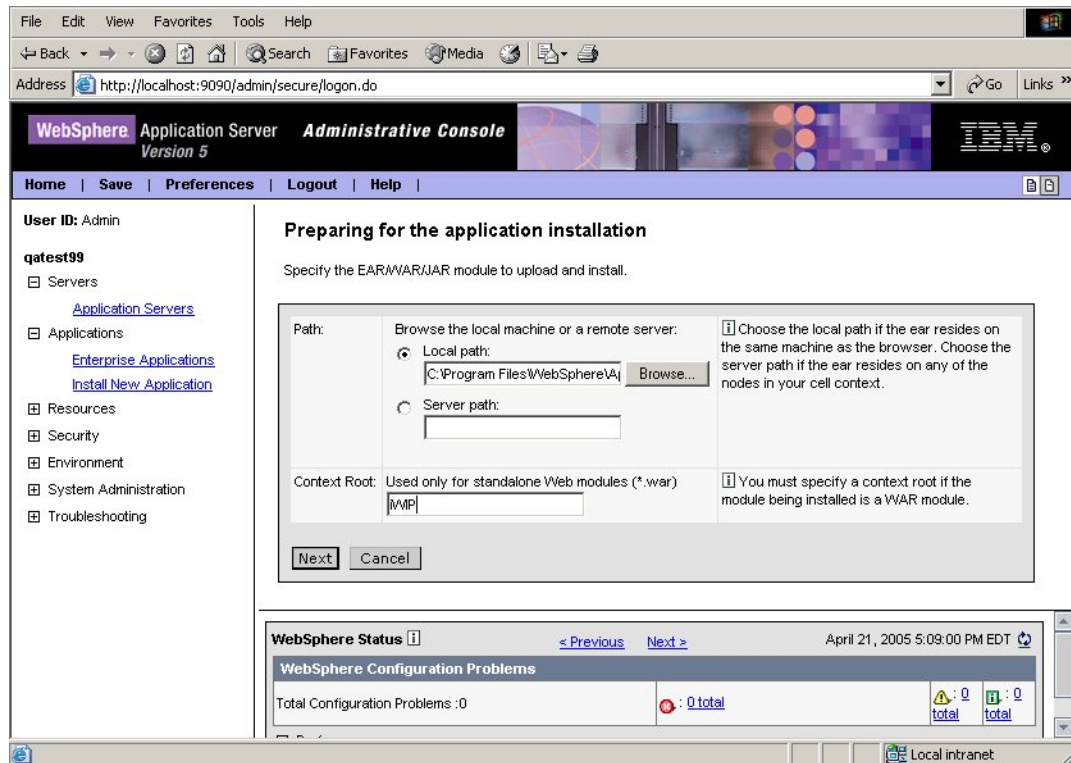
Figure 22: Install a New Application

Figure 23 shows the next screen that lets you bind a host to the application. WebSphere lets you set up virtual hosts. Consult the IBM documentation for more information on how to do this. In this example, use the default host. Click Next to continue.
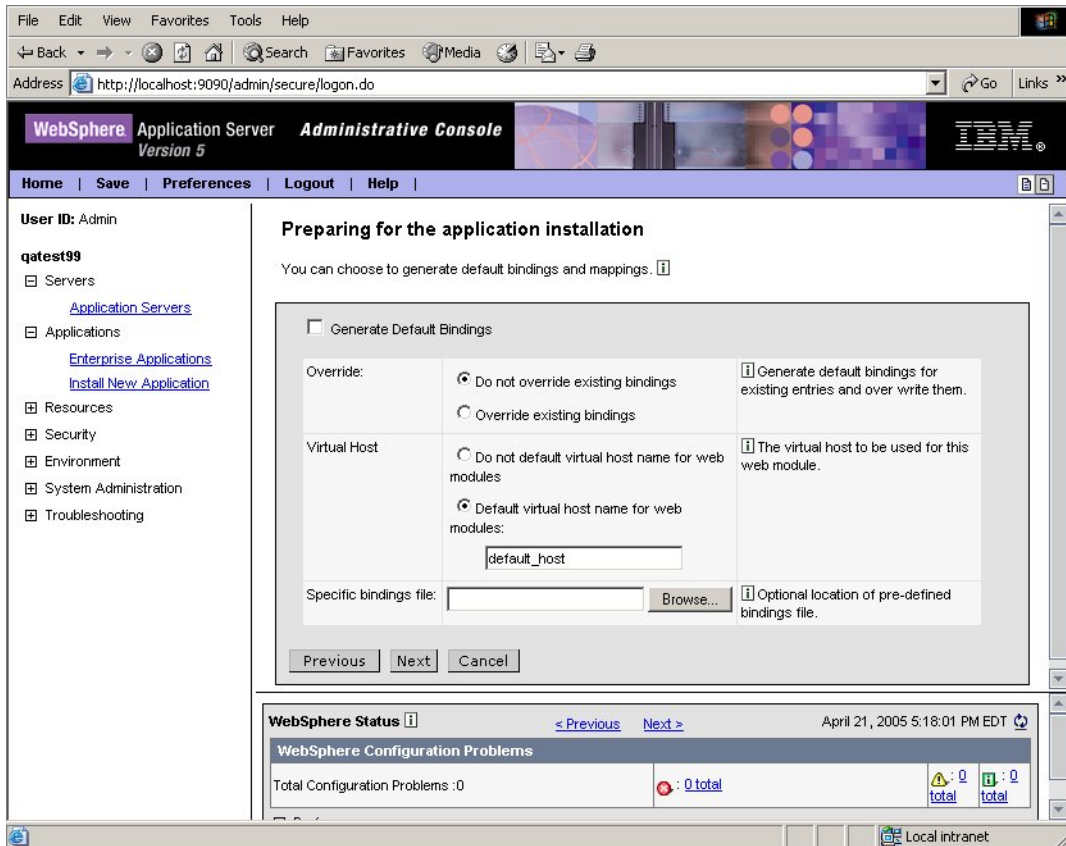
Figure 23: WAS Host Bindings

Figure 24 shows the Application Security Warnings page. In a production environment, you will want to take a look at these and address each item. Consult the IBM documentation for more information on WebSphere security settings. For this document, the defaults are acceptable. Click Next to continue.
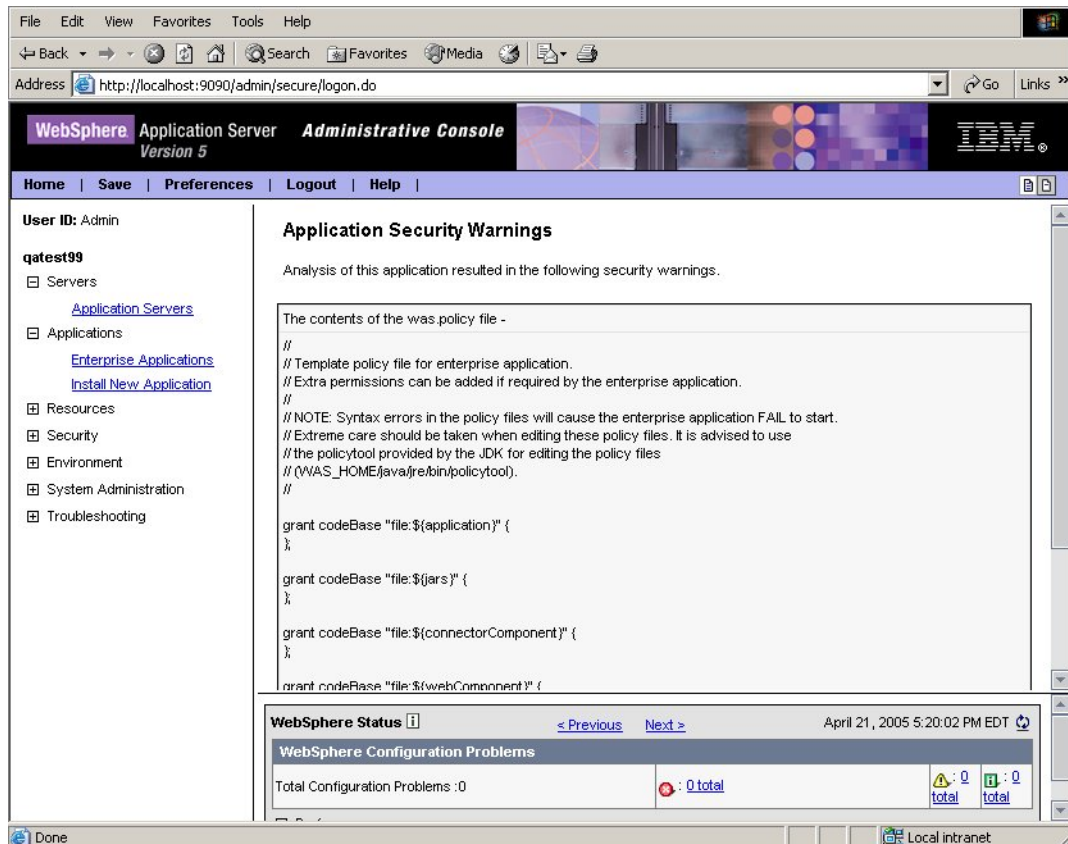
Figure 24: WAS Security Warnings

Figure 25 shows the next screen that lets you set options for the install. WebSphere automatically filled out the Application Name setting. That value was set when the Context Root was defined. In a production environment, you can set items like Pre-compile JSP. For now, however, make your settings look like those shown in the figure. We will not try any optimizations. Once you've filled out this form, click Next to continue.
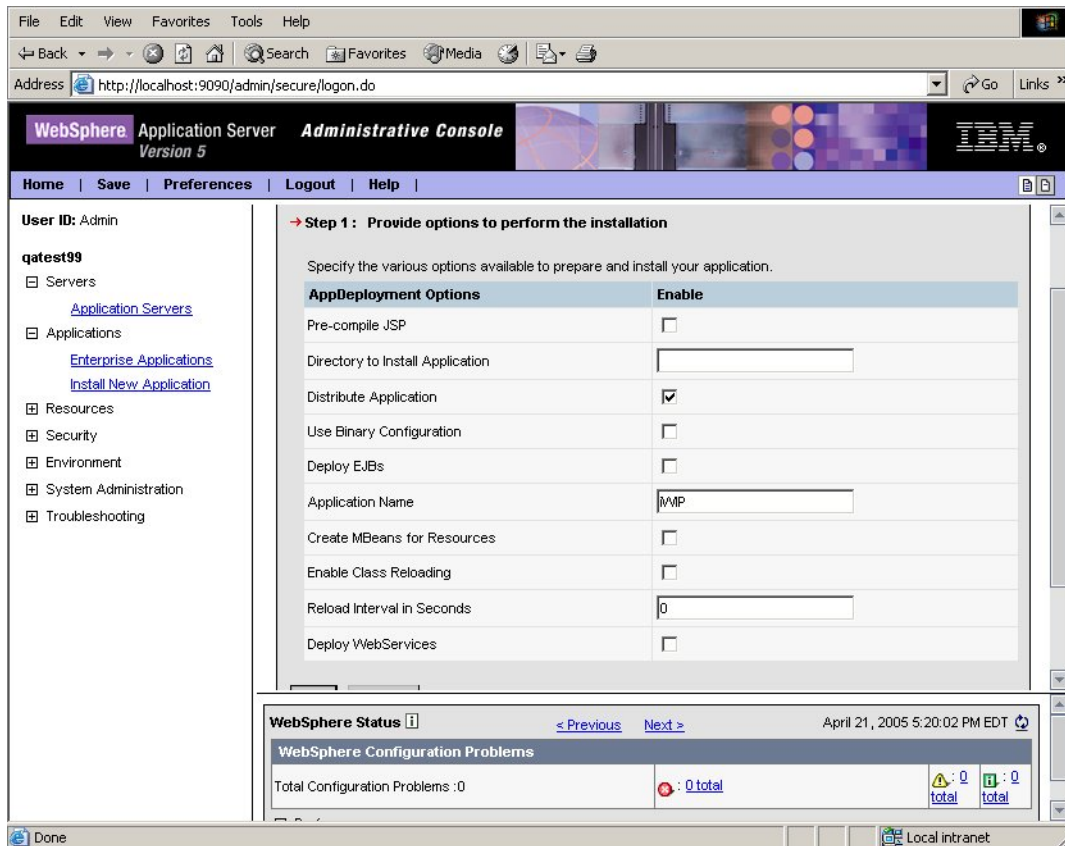
Figure 25: WAS Installation Options (Step 1)

Figure 26 shows the WAS screen that lets you map the virtual host to use for the application. The default value will be used. Click Next to continue.
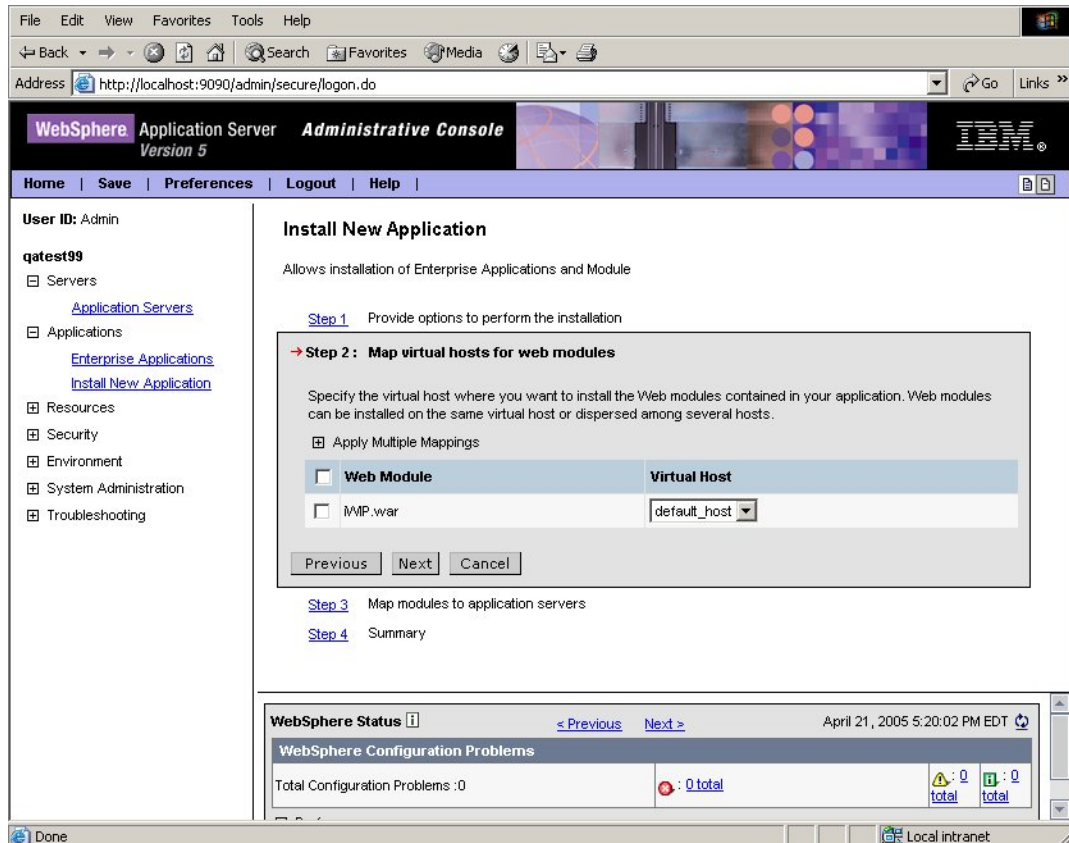
Figure 26: WAS Virtual Host Settings (Step 2)

Figure 27 shows the next screen where the WAR file module is mapped to the application
server. The default value will be used. Click Next to continue.
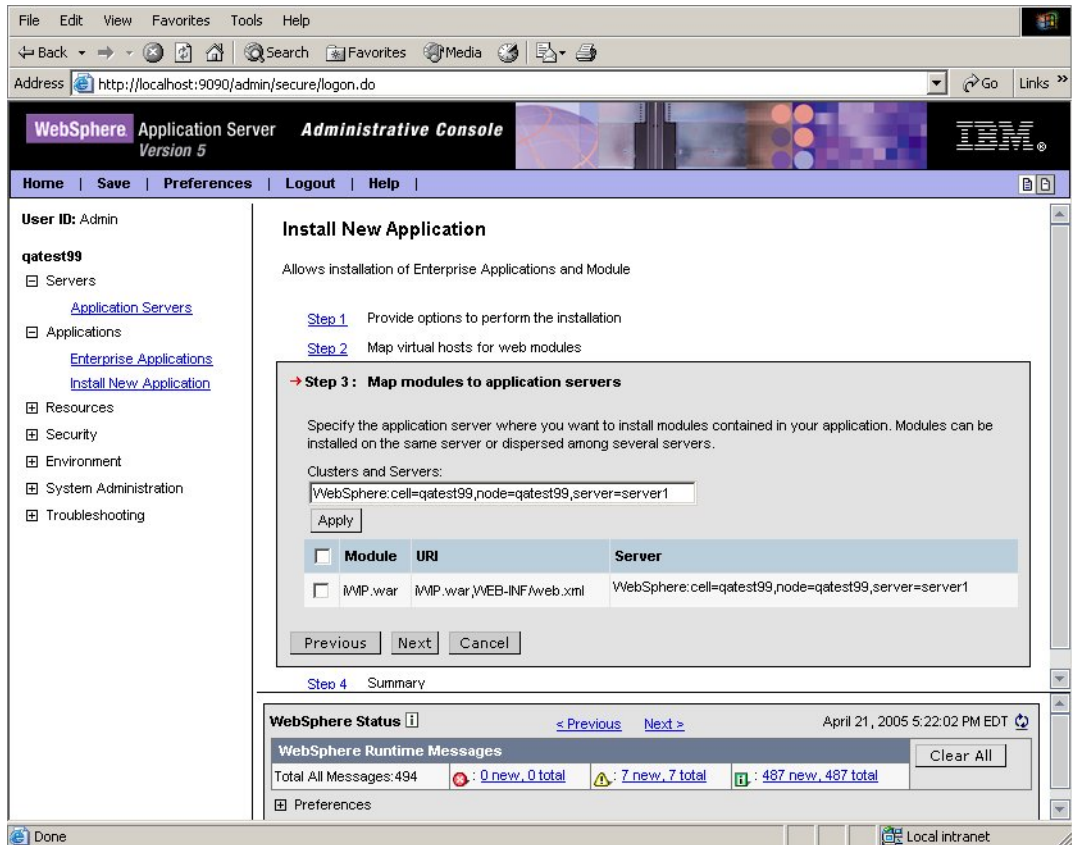
Figure 27: Map Modules to Application Servers

Figure 28 shows the WebSphere summary page for the application installation. You can review the settings you made during the installation on this page. If your screen looks like the figure, then you are ready to deploy the application to WebSphere. Click Finish to complete the installation.
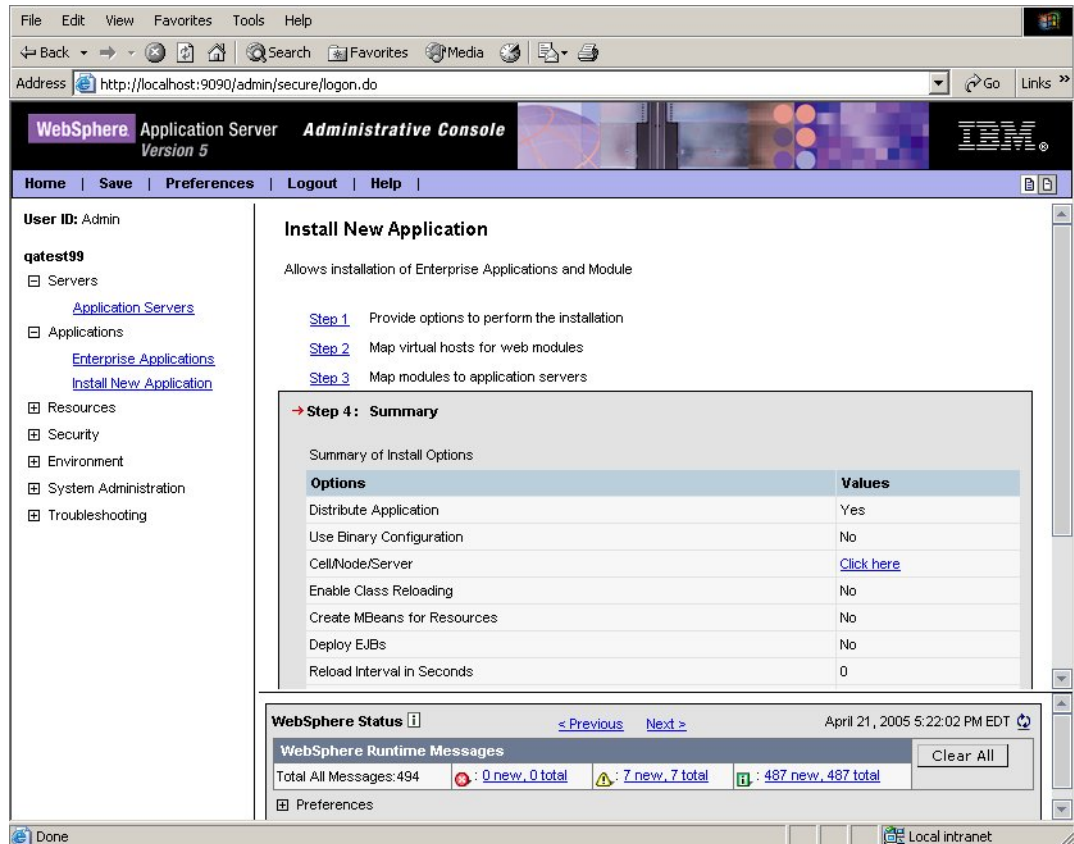
Figure 28: WAS Install Application Summary Page

Figure 29 shows the WebSphere screen that appears when the application is installed. You will be informed on the success of the install. In this instance, the installation was successful.

The installation must be saved to the Master Configuration file. Click Save to Master Configuration to continue.
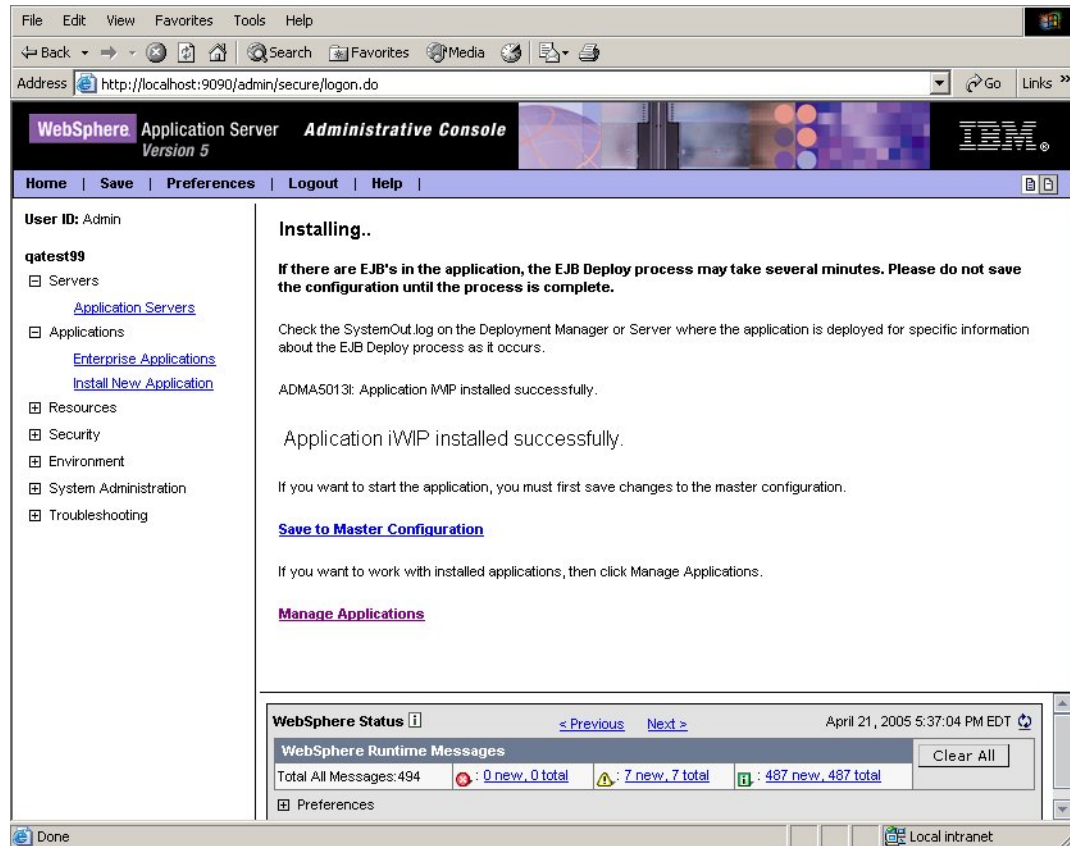
Figure 29: Installation Screen

Figure 30 shows the application server Master Configuration save screen. Even though you clicked on the Save to Master Configuration link, the deployment information has not been written to the WebSphere configuration files. You need to click Save near the top of the window (the "Message(s)" area) in Figure 30 to really save the configuration information.
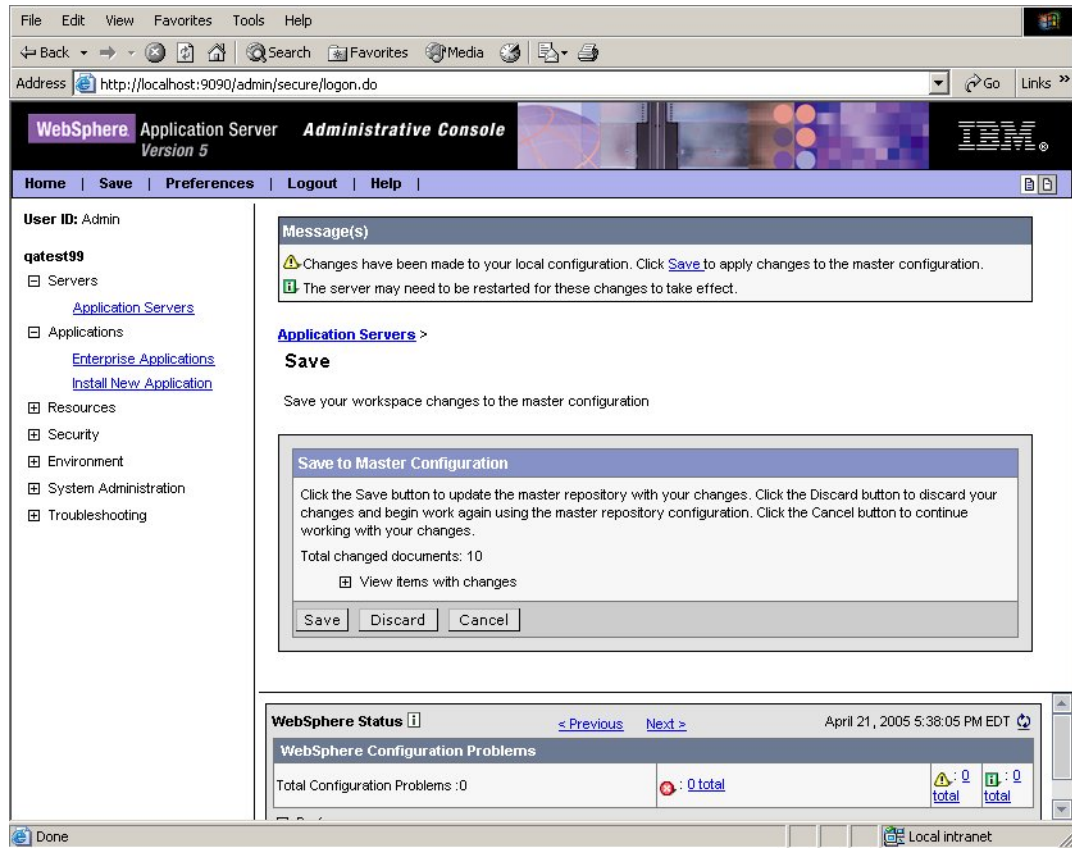
Figure 30: WAS Save Screen

Now that the deployed application has been saved to the WebSphere Master Configuration File, you can check on the status by clicking the Enterprise Application hyperlink shown in Figure 31. You will see a list of applications that have been deployed to this server. In this case, there are many example files that IBM shipped with the application server. The icons under the Status column show the running state of each application.

Before you start the iWIP application, there are two more major steps in the configuration you must perform.
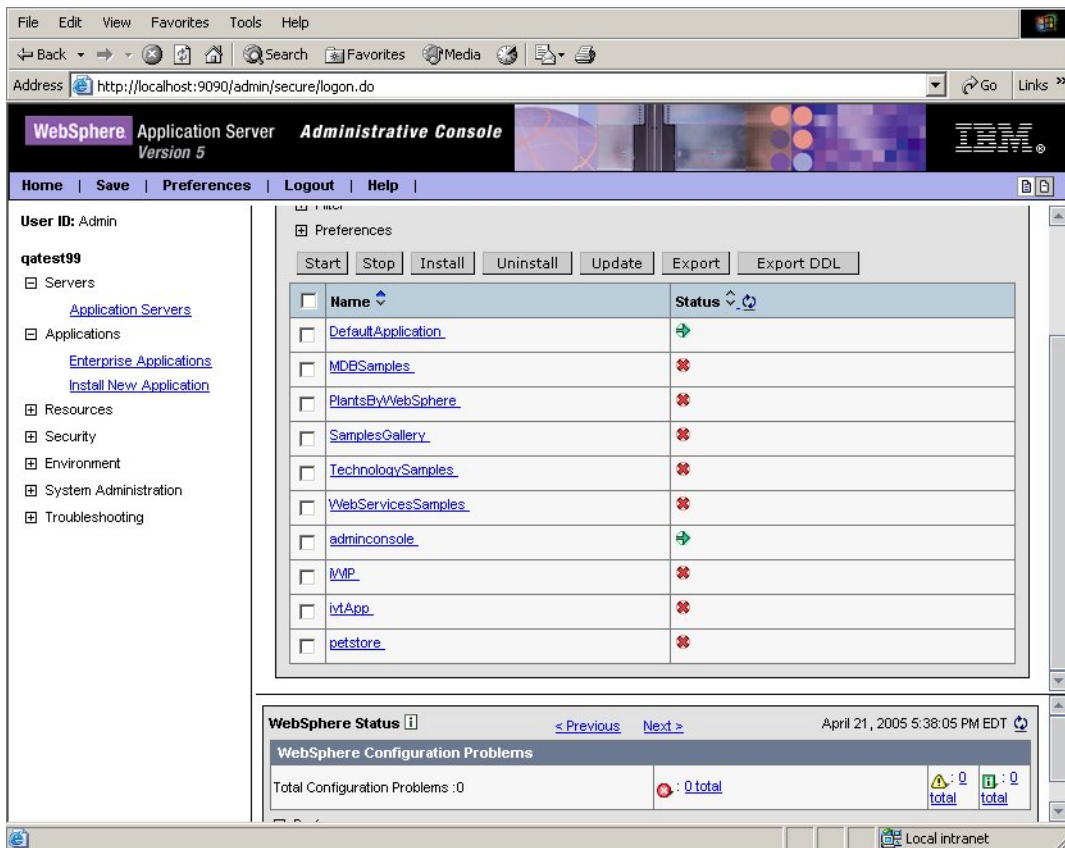
Figure 31: WAS Applications | Enterprise Applications Screen

WebSphere ships with an HTTP server (their own version of the Apache web server). The HTTP server does not automatically know how to route requests to the Application Server. IBM requires a plug-in for the web server that binds URL requests to the application server.

Click the Environment hyperlink on the left hand side of the WebSphere main admin window to expand the menu tree. Click Update Web Server Plugin and you will see the screen shown in Figure 32.

Click Ok to generate the plug-in file. WebSphere will generate the plug-in file and automatically transfer it to the web server if it is installed on the same machine as the application server.

For this appendix, we do have the built-in web server installed on the same machine. This lets you test the iWIP application once the configuration settings are complete. In a production environment, your HTTP server will probably be located on another machine. Consult the IBM documentation to see where you need to install the plug-in in that instance.
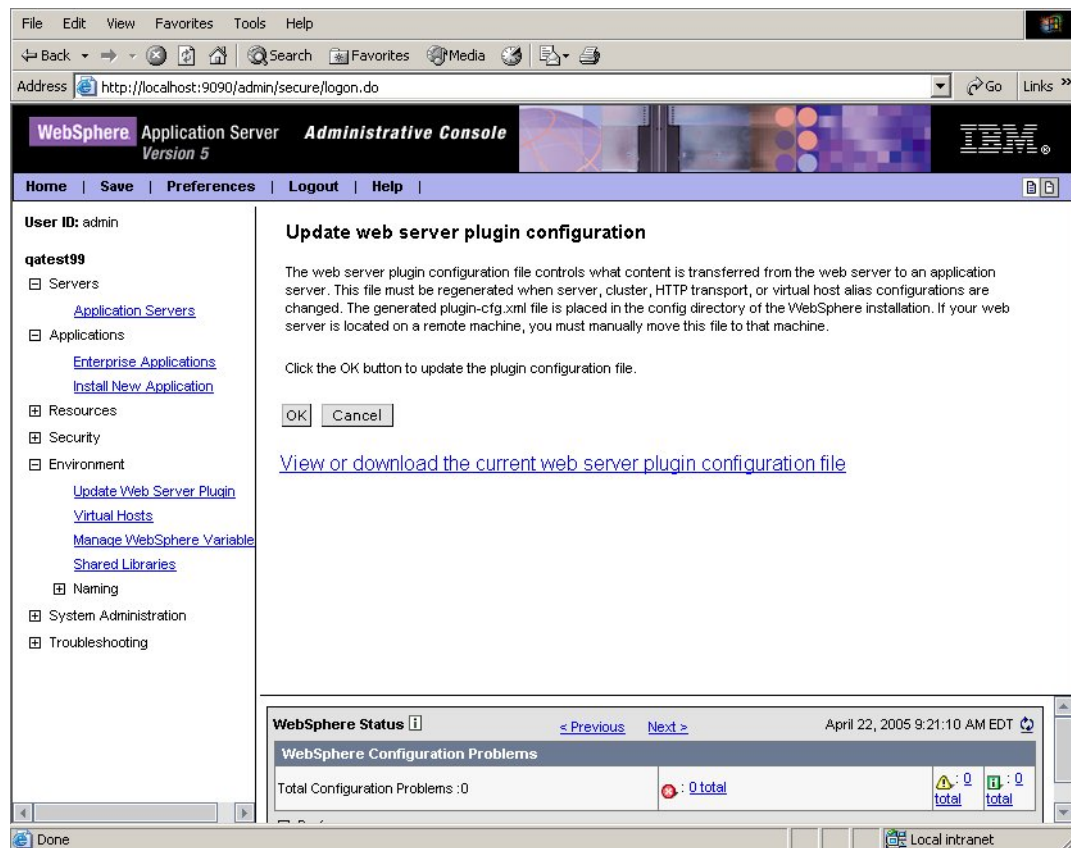
Figure 32: WAS Server Plugin Configuration Page

The final part of the configuration consists of telling the JVM that is running the iWIP application to load the dsimsgclient.properties file when it starts up. This file defines the configuration parameters for the queue manager that is used with IDS and the iWIP application.

To configure the JVM, do the following:

1   From the WebSphere Main Admin Console page (Figure 21) click on the Servers item on the left hand side.

2   The menu will expand to show a Application Servers link. Click this link.

3   The screen that appears has two tabs: Runtime and Configuration. Click the Configuration tab to select it.

4   Scroll down this window. In a section called Additional Properties you will see a link called Process Definition as shown in Figure 33. Click this link.
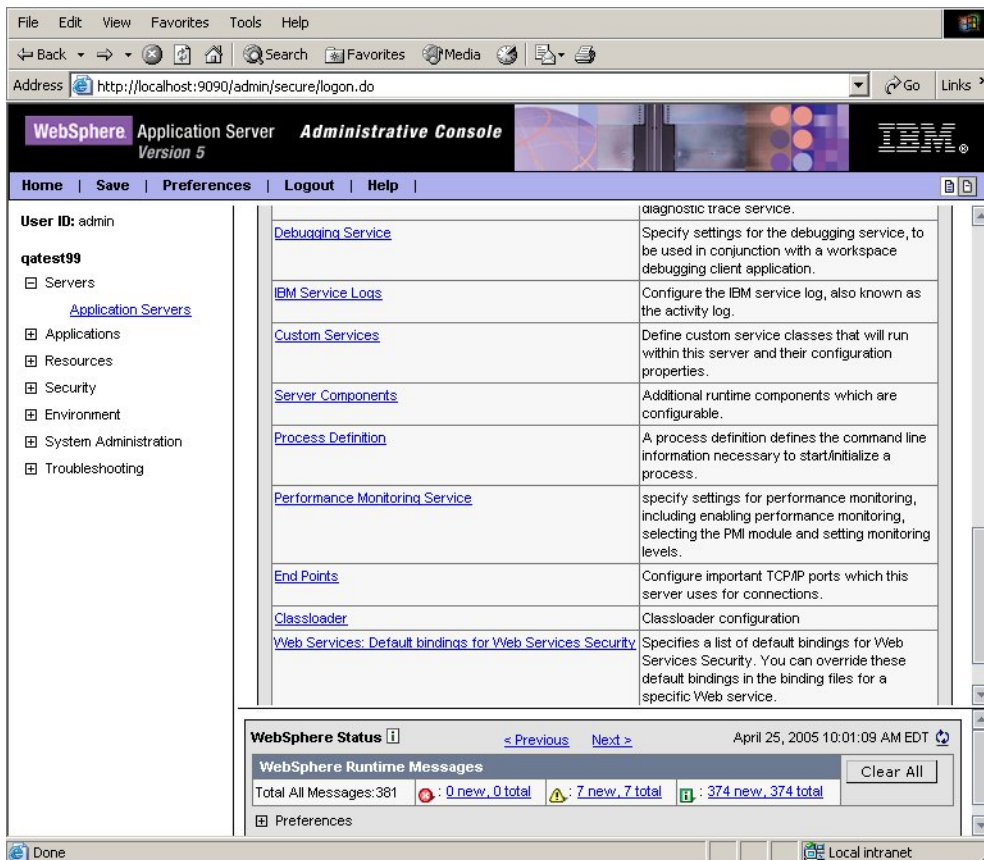
Figure 33: Application Server Configuration

116

The screen shown in Figure 34 appears. Under the Additional Properties section there is a Java Virtual Machine hyperlink. This is where you can set parameters for the specific JVM that is running your web server application. Each server in WebSphere has its own JVM. The settings will apply to the JVM, so any applications deployed to that server will inherit these settings as well. Click the Java Virtual Machine hyperlink.
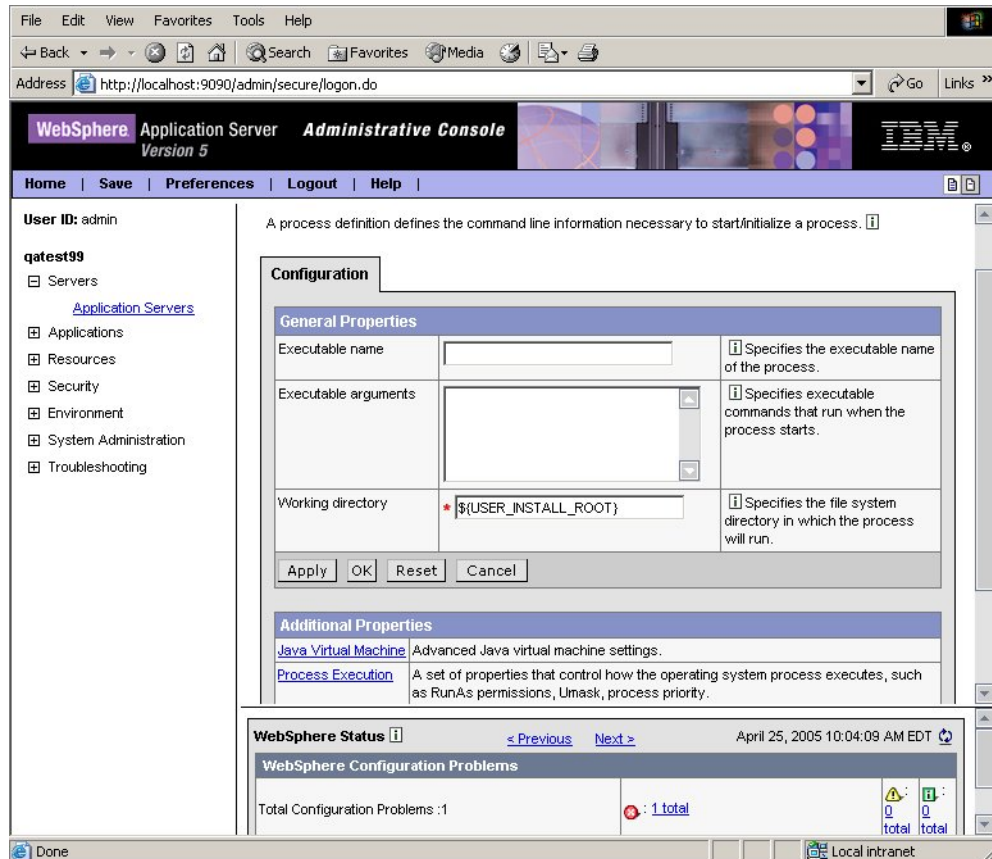
Figure 34: JVM Configuration

Scroll down the page shown in Figure 35 and you will see a text box entry labeled *Generic JVM arguments.* Earlier you created a directory called Sampco off your root drive and copied the dsimsgclient.properties file to that location. Now enter that information into the Generic JVM arguments edit control. Assuming the Sampco directory is on the C:\ drive, enter this in the edit control:

    -Ddsimessage.properties=C:\Sampco\dsimsgclient.properties

The -D flag tells the JVM to define a property called *simessage.properties* and set the value equal to *C:\Sampco\dsimsgclient.properties.* The next time the server starts, the JVM loads this value into memory where it can be accessed by the iWIP application.

NOTE: The entry in the text box is case sensitive (except for the drive and directory name). Make sure you enter it exactly as shown.
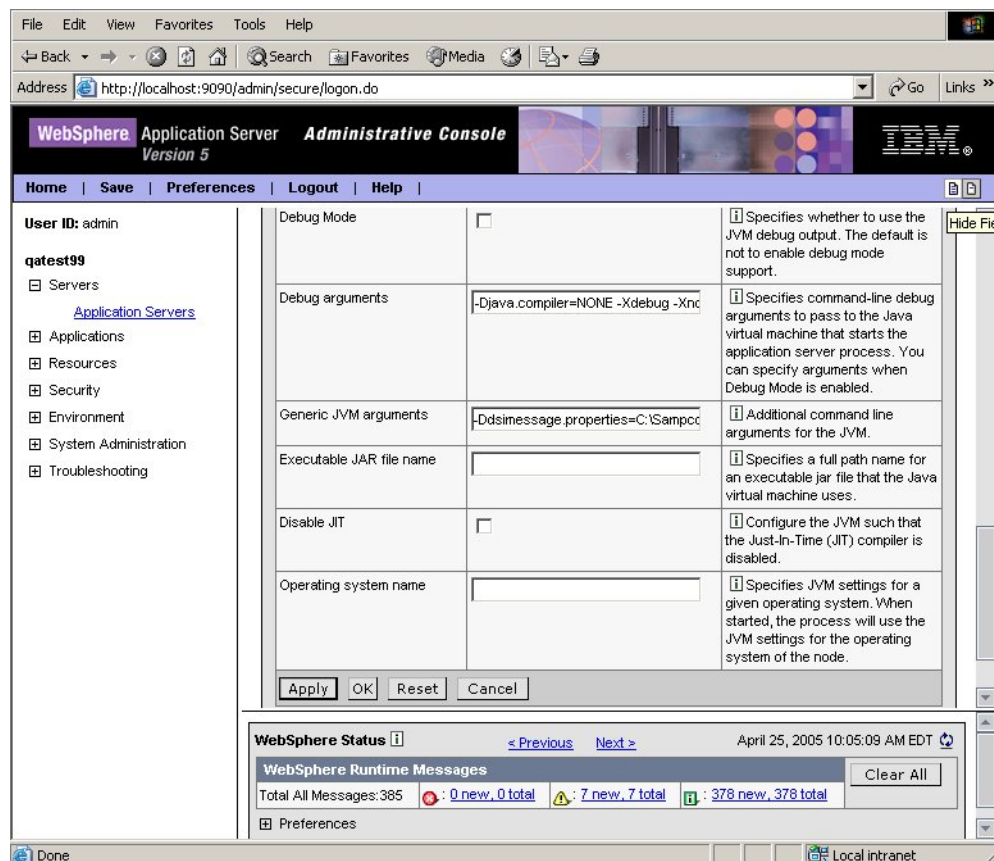
Figure 35: JVM Additional Arguments

Now you must re-start the WebSphere application server for this value to be loaded into memory by the JVM. The easiest way to start and stop the server is through the First Steps application IBM ships with the application server. It should be on the Start button, Programs, IBM WebSphere, Application Server v5.1 menu.

Before launching it, log out of the Admin Console browser window — there is a log out option near the top of the window. Launch the First Steps application and click on the Stop Server hyperlink. Once it confirms that the server status is stopped, click the Start Server hyperlink. When you see this message in the console window, the server is ready:

```
open for e-business
```

TESTING THE WEBSPHERE INSTALLATION

Use the web server that is built into WebSphere to test WIP Edit. The default port for this web server is 9080. You use that port when you launch a browser on the same machine that hosts the WebSphere application server.

We will test the client side on the WebSphere machine. This provides an added feature of eliminating any communications or access issues that may be associated with your environment (such as firewalls, and so on). By testing the client side on the WebSphere box, you can tell whether the installation was successful.

Install the client side WIP Edit on the machine that is running WebSphere. Refer to Setting Up the WIP Edit Client on page 13 for more information. After you install WIP Edit, you have to configure Internet Explorer for ActiveX controls.

You have to use a different URL for WebSphere (compared to Tomcat). Once the browser has been configured, enter the following URL:

```
http://localhost:9080/iWIP/login.jsp
```

Once you get the login screen, you can proceed to Configuring Internet Explorer for ActiveX controls on page 16 and pick up with step 2.

Setting Up Print Preview with WebSphere 5.1 as the Mid-tier

Appendix C

Modifying the GLOBAL.XML File

This appendix identifies the modifications required in the GLOBAL.XML file to use the WIP Edit ActiveX component with iPPS. The assumption is that you already have a working iPPS system that generates HTML.

The basics of setting up a GLOBAL.XML file from scratch are not covered here. However, a complete example of a GLOBAL.XML file is included.

This appendix discusses these topics:

MODIFYING THE SYSTEM ELEMENT IN GLOBAL.XML

The COM+ implementation of iPPS supports caching to application object when the MRL resources are obtained from IDS.

To enable caching, in the System element, under the ApplicationData, ActiveServer node, add the following:

```
<OBJECTCACHE Active="YES" EXPIRETIME="" MAXSIZE=""
TYPE="0101000001101" DESTINATION="" SERIALIZE=""/>
```

MODIFYING THE MRL ELEMENT IN GLOBAL.XML

This section will use the Amergen MRL as an example. To use the WIP Edit ActiveX control to display forms, put the following in your file:

1 Under the Amergen element, ApplicationData, ActiveServer add this entry:

```
<!--Use this node to define HTML location if not using MRL database
PROTOCOL Attribute valid values:
DBMS, IDS, URL, PLUGIN
-->
<IFORMS PROTOCOL="PLUGIN" DOMAIN="" HOST="$HOST" ROOT="$ROOT"
PORT="21" PASSWORD="guest" USERID="anonymous">$FORMSLOC</IFORMS>
```

IFORMS is a new element to support WIP Edit. The PLUGIN setting for Protocol tells iPPS to send the forms to WIP Edit on the client side.

If you have the <IFORMS> element node uncommented and you set the value of PROTOCOL to *DBMS*, then the value for $FORMSLOC must match the value you have entered in the DBASE, TABLE, NAME attribute. For instance, if the entry looks like this:

```
<DBASE>
<TABLE NAME="iMRLDB">
[Other entries omitted]
</DBASE>
```

The value of $FORMSLOC will be *iMRLDB*.

If you set PROTOCOL to *IDS* then the value set in $FORMSLOC is not used. If you set PROTOCOL equal to *URL* then the value of $FORMSLOC must provide a valid URL location to the forms.

2 In the DBASE element, under Table there is a DSN element. Edit the MRLSOURCE field. Valid entries are DBMS or IDS. If you enter *IDS* the Internet Document Server generates resources in real time.

3 The PUBLISHEXPORT element specifies the export type to use when calling Documaker Server for the print engine. It tells the system to just export the combined NA/POL file when calling RunRP.

```
<PUBLISHEXPORT RPEXPORT="CMBNA">XML</PUBLISHEXPORT>
```

4 You can specify where the form navigation will go when you choose to work on a WIP item. In the PPS element add the following:

```
<!--
 Valid WIPMODES
        SERVICE_ENTRY_NEW_TRANSACTION (comp_lob.asp)
        SERVICE_ENTRY_FORMS_Selection (formset.asp)
        SERVICE_ROUTE_PROCESSINGOPTIONS (control.asp)
        SERVICE_ENTRY_WORK_ON_FORM (form.asp)


 -->
 <WIPMODE>SERVICE_ENTRY_NEW_TRANSACTION</WIPMODE>
```

5 You can configure the system to provide a WIP search screen. These settings must be in place:

```
Searchable = Yes
Key = Yes
```

Under the Amergen, Services, Service Name ="WIP" section, set the Searchable parameter to Yes, as shown here:

```
<SERVICE NAME="WIP" TYPE="iWip" SEARCHABLE="YES">
```

In Amergen, Services, Service Name="WIP" | "WIPKeys", add a Key = Yes entry, as shown here:

```
<WIPKEYS>
<!--  Base 23 Keys. The keys match the base wip.dfd
  -->
  <KEY1 KEY="YES" NAME="KEY1" DOCSETHEADINGS="Company" DISPLAY="YES"
/>
  <KEY2 KEY="YES" NAME="KEY2" DOCSETHEADINGS="Line of Business"
DISPLAY="YES" />
  <KEYID KEY="YES" NAME="KEYID" DOCSETHEADINGS="Policy Number"
DISPLAY="YES" />
…
```

6 If you want to be able to mail a completed form to someone, you need to configure a mail server in the GLOBAL.XML file. In the System configuration element, add the following and fill out the entries for your mail server:

```
<MAIL TYPE="SMTP" HOST="$EMAILHOST" DOMAIN="$EMAILDOMAIN"
PASSWORD="" USERID="" PORT="25"/>
```

7 Under the MRL data application section there is an element called Cache. Make sure the Root attribute points to the CachePath. Active must be set to True. For example, on Windows it would look like this:

```
<CACHE ROOT="c:\inetpub\wwwwroot\_ipps311\cache"
CLEARONSTARTUP="TRUE" ACTIVE="TRUE" />
```

8 In the i_PluginInit request type, the PUTURL setting should be either LocalHost or the IP address of the server. Here is an example:

```
<VAR NAME="PUTURL">localhost</VAR>
```

CHANGING THE REQUEST TYPES

There are several types of changes to request types you must make in the GLOBAL.XML file:

- LOW.LEVEL.SERVICE attribute changes

- New request types to support WIP Edit functionality

- Updated request types

The LOW.LEVEL.SERVICE binds a numeric value to the request type. This number is used internally by iPPS. The changes you need to make are for going forward with the product evolution.

LOW.LEVEL.SERVICE CHANGES

```
<REQTYPE DESCRIPTION="" NAME="i_Print" LOW.LEVEL.SERVICE="5180">
<REQTYPE DESCRIPTION="" NAME="i_PrintFormsetXML"
LOW.LEVEL.SERVICE="5181">
<REQTYPE DESCRIPTION="" NAME="i_Proof" LOW.LEVEL.SERVICE="5191">
<REQTYPE DESCRIPTION="" NAME="i_CheckPolicy"
LOW.LEVEL.SERVICE="2175">
<REQTYPE DESCRIPTION="" NAME="i_DPRModifyWIPData"
LOW.LEVEL.SERVICE="1012">
<REQTYPE DESCRIPTION="" NAME="i_DPRAssignWIP"
LOW.LEVEL.SERVICE="1005">
<REQTYPE DESCRIPTION="" NAME="i_ArchiveFormset"
LOW.LEVEL.SERVICE="3070">
```

ADDING REQUEST TYPES

The new request types should be added to the System element, REQTYPES section of the GLOBAL.XML file. This is a multi-page listing. If you are upgrading an existing installation, the best way to edit your files is to copy the sections from the sample GLOBAL.XML file that ships with iPPS into your existing GLOBAL.XML file. One particular request type to look at is i_PluginInit. Make sure you edit any field with a $Variable placeholder.

```
<REQTYPE DESCRIPTION="Returns groups from IDS"
NAME="i_GetMRLResource" LOW.LEVEL.SERVICE="7000">
        <INPUT>
            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
            <VAR NAME="DOCUMENTSTREAM" SOURCE=""
RETURNVIAQUEUE="YES" ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
            <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
        </INPUT>
        <OUTPUT>
            <VAR NAME="RESULTS"/>
        </OUTPUT>
    </REQTYPE>
    <REQTYPE DESCRIPTION="Returns forms list from IDS" NAME="
i_GetMRLResource " LOW.LEVEL.SERVICE="7001">
        <INPUT>
```

```
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="IMPORTFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="XMLIMPORT"/>
                            <VAR NAME="DOCUMENTSTREAM" SOURCE=""
RETURNVIAQUEUE="YES" ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
                            <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                            <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Returns formset shell from IDS" NAME="
i_GetMRLResource " LOW.LEVEL.SERVICE="7002">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="IMPORTFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="XMLIMPORT"/>
                            <VAR NAME="DOCUMENTSTREAM" SOURCE=""
RETURNVIAQUEUE="YES" ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
                            <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                            <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Returns html from IDS" NAME="
i_GetMRLResource " LOW.LEVEL.SERVICE="7003">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="IMPORTFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="XMLIMPORT"/>
                            <VAR NAME="DOCUMENTSTREAM" SOURCE=""
RETURNVIAQUEUE="YES" ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
                            <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                            <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Performs a spell check" NAME="SPELL"
LOW.LEVEL.SERVICE="2135">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="IMPORTFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="XMLIMPORT"/>
                            <VAR NAME="EXPORTFILE" SOURCE="" RETURNVIAQUEUE="YES"
ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
```

```
                                <VAR NAME="USERDICT" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="LANGUAGEOPT" SOURCE=""/>
                                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                                <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Returns the user dictionary"
NAME="RETUSERDICT" LOW.LEVEL.SERVICE="2136">
                    <INPUT>
                                <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                                <VAR NAME="EDITFILE" SOURCE="" RETURNVIAQUEUE="YES"
ATTACHMENTPREFIX="DOCUMENTSTREAM"/>
                                <VAR NAME="USERDICT" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="LANGUAGEOPT" SOURCE=""/>
                                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                                <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Edits the user dictionary"
NAME="EDTUSERDICT" LOW.LEVEL.SERVICE="2137">
                    <INPUT>
                                <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                                <VAR NAME="EDITFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="XMLIMPORT"/>
                                <VAR NAME="USERDICT" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="LANGUAGEOPT" SOURCE=""/>
                                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                                <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Pulls Data from IDS for Tables"
NAME="i_Tbllkup" LOW.LEVEL.SERVICE="2165">
                    <INPUT>
                                <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                                <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                                <VAR NAME="TABLEFILE" SOURCE=""/>
                                <VAR NAME="TABLEID" SOURCE=""/>
                                <VAR NAME="TABLERETURNS" SOURCE="">DESCRIPTION</VAR>
                                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
```

```xml
                            <VAR NAME="RESULTS"/>
                            <VAR NAME="RECORDS" RETURNROWSET="YES"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="" NAME="i_PluginInit"
LOW.LEVEL.SERVICE="7006">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="PASSWORD" SOURCE="">DEMO1</VAR>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="PRTTYPE" SOURCE="">DPW</VAR>
                            <VAR NAME="RF_POSTFILE" SOURCE=""
RETURNVIAQUEUE="YES" ATTACHMENTPREFIX="RF_POSTFILE"/>
                            <VAR NAME="RECNUM" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/WIPKEYS/RECNUM"/>
                            <VAR NAME="HTTPQUERYSTRING">1</VAR>
                            <VAR NAME="HTTPQUERYSTRING1.NAME">SessionID</VAR>
                            <VAR NAME="HTTPQUERYSTRING1.VALUE" SOURCE=""
XPATHSOURCE="1" DATAXPATH="DOCUMENT/SESSION/ID"/>
                            <VAR NAME="SAVE_REQTYPE">i_PluginSave</VAR>>
                    <VAR NAME="SCRIPT">ipps3.2/wipsave.asp</VAR>
                    <VAR NAME="GETSCRIPT">ipps3.2/wipdownload.asp</VAR>
                    <VAR NAME="PUTURL">$HOST</VAR>
                    </INPUT>
                    <OUTPUT>
                            <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Saves Plugin data back to wip"
NAME="i_PluginSave" LOW.LEVEL.SERVICE="7007">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="RF_POSTFILE" SOURCE="" SENDVIAQUEUE="YES"
ATTACHMENTPREFIX="RF_POSTFILE"/>
                            <VAR NAME="DPWRECNUM" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/WIPKEYS/RECNUM"/>
                            <VAR NAME="RF_POSTFILE.FILENAME" SOURCE=""/>
                            <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
                    </INPUT>
                    <OUTPUT>
                            <VAR NAME="RESULTS"/>
                    </OUTPUT>
            </REQTYPE>
            <REQTYPE DESCRIPTION="Retrieves resource from IDS"
NAME="i_PluginGetResource" LOW.LEVEL.SERVICE="7008">
                    <INPUT>
                            <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                            <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                            <VAR NAME="RETURNFILE" SOURCE="" RETURNVIAQUEUE="YES"
ATTACHMENTPREFIX="RETURNFILE"/>
                            <VAR NAME="RESOURCENAME" SOURCE=""/>
```

```
                <VAR NAME="RESOURCETYPE" SOURCE=""/>
                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
            </INPUT>
            <OUTPUT>
                <VAR NAME="RESULTS"/>
            </OUTPUT>
        </REQTYPE>
<REQTYPE DESCRIPTION="This request type performs a wip filter"
NAME="i_DPRFindWipRecord" LOW.LEVEL.SERVICE="1108">
<INPUT>
    <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
    <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
    <VAR NAME="STARTRECORD" SOURCE="">1</VAR>
    <VAR NAME="MAXRECORDS" SOURCE=""/>
    <VAR NAME="" SOURCE="SERVICES/SERVICE/WIPKEYS/
node()[@KEY='YES']"/>
    <VAR NAME="" SOURCE="SERVICES/SERVICE/WIPKEYS/CUSTOMKEYS/
KEY[@KEY='YES']"/>
    <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
</INPUT>
<OUTPUT>
    <VAR NAME="RESULTS"/>
    <VAR NAME="MORERECORDS"/>
    <VAR NAME="RECORDS" RETURNROWSET="YES"/>
</OUTPUT>
</REQTYPE>
```

UPDATING REQUEST TYPES

This update to the i_PrintWIPFormset request type lets you put a watermark graphic in the PDF file when you click the Proof button in WIP Edit.

This element is found in the System, REQTYPE section of the GLOBAL.XML file:

```
<REQTYPE DESCRIPTION="" NAME="i_PrintWIPFormset"
LOW.LEVEL.SERVICE="1010">
            <INPUT>
                <VAR NAME="USERID" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/SESSION/USERID"/>
                <VAR NAME="CONFIG" SOURCE="" XPATHSOURCE="1"
DATAXPATH="DOCUMENT/DOCSET/LIBRARY/@CONFIG"/>
                <VAR NAME="RECNUM" SOURCE=""/>
                <VAR NAME="PRINTFILE" SOURCE=""/>
                <VAR NAME="PRTTYPE" SOURCE="">PDF</VAR>
                <VAR NAME="ALLRECIPIENTS" SOURCE=""/>
                <VAR NAME="DPRPROOFLOGO">NO</VAR>
                <VAR NAME="DSITIMEOUT" SOURCE="">30000</VAR>
            </INPUT>
            <OUTPUT>
                <VAR NAME="RESULTS"/>
            </OUTPUT>
        </REQTYPE>
```

130

CHANGING THE WIPEDIT.INI FILE

There are some minor changes required for the WIPEDIT.INI file, located in the Docserv directory. This listing shows how this file should look:

```
< WIPSave >
    OverridePrompt = Yes
    SaveOnExit      = Yes
< WIPUnlock >
    UnlockOnExit    = No
< Control ?
    NoLoadFormList = Yes
< WIPEdit >
    TrapEvents      = No
```

Listing 24: WIPEDIT.INI in the Docserv directory

Appendix D

Optimizing Performance

Tuning web applications for maximum response times is usually approached by first knowing the target workflows/use-cases where tuning is desired in relation to how they interact with the web application server and the back-end services such as DBMS and message queuing systems. This appendix can assist in initial configuration of the web application server, web application and the back-end services it uses to get initial acceptable performance for response times.

The settings and parameters described in this appendix are based on load tests conducted in our environment on our hardware. Since your environment differs, keep in mind that your test results will need to be analyzed, and subsequent retuning must occur based on those results. These settings are mere guidelines.

This appendix includes information on optimizing these components:

WEBSPHERE APPLICATION SERVER (WAS)

STATIC CONTENT SERVING

All content that can be served up by the web server should be served up by the web server and not by WAS's internal HTTP server. The fileServingEnabled property is the key to controlling static file serving. This setting is contained in the ibm-web-ext.xmi file in the WAR. It should be set to False so static content is served up by the web server and not by the web applications internal HTTP server.

Web application servers' internal HTTP servers are typically provided to ease setup and configuration for development purposes and should not be used in production. Even PDF files created dynamically by Oracle Insurance products can and usually should be served up by the web server instead of being piped through MQ if at all possible.

WEB SERVER TUNING

Generally you should set up the maximum number of processes so the web server passes performance tests.

NOTE: For IBM's HTTP Server (IHS) or Apache's Web Server v2.x these are called *threads*. Typically, the number of processes should be set to 1000 for MaxClient in IHS v1.3.x.

The KeepAliveTimeout, MinSpareServers, and MaxSpareServers should usually be left at the defaults. However, some people prefer to set the MinSpareServers and MaxSpareServers to equal values. This will keep forking to a minimum which will reduce CPU utilization if that becomes a concern.

WEBSPHERE TUNING PARAMETERS

Tuning the inbound queue for WAS depends on a number of factors:

•   Number of concurrent users to support

•   Memory requirements of the web application

•   Amount of database access from the web application

In Oracle Insurance performance tests, we are able to push up to 200 concurrent users for our hardware for Linux, AIX, and HP-UX platforms. Because these machines are robust in terms of memory, the web container's maximum thread size can be set fairly higher then the default value of 50. For most testing, this number should be half the number of virtual users in the load test. If we determine after initial testing that the application has heap space and CPU time to spare, then the maximum thread size can be increased. If there is plenty of room, then set the flag to allow this number to grow beyond the maximum.

JVM HEAP SIZE

For most web applications set the following:

• Minimum heap size to 512MB

• Maximum heap size to 1024MB/1GB.

Analysis of JVM performance will show if garbage collection is a performance bottleneck. This can occasionally be remedied by setting the maximum/minimum heap size to equal values (such as 1024MB each). Keep in mind that these settings depend on the amount of RAM on the system and how much is available.

It is helpful to use LoadRunner monitor or other tools to collect system resource usage values to determine RAM usage statistics. If too much physical RAM is being used, the JVM heap uses the less-efficient virtual RAM, which can cause excessive swapping to occur. Here are some statistics to monitor:

• Rstat service statistics

• Swap-in

• Page-in

• Disk transfer rate

WEBSPHERE 5 CLASSLOADER MODE SETTING

There is a known memory leak in JAXP1.1 that comes with WebSphere AS v5. iPPSj is distributed with Sun's endorsed JAXP1.2. This version is newer than the version that ships with J2EE1.3. To guarantee that WAS uses the version of JAXP included in iPPSj a configuration parameter must be set for the WebModule. This configuration parameter is **_Classloader_** mode and should be set to PARENT_LAST.

NOTE: There are a few other Classloader Mode settings in WAS. These can be left at their default values.

JNDI/JDBC PROVIDER SETUP

The Connection Pool settings for the DataSource should be tuned to allow the maximum number of connections that the application may need for performance. Typically this should be set relatively low so that the database is not overwhelmed by the application. Our performance testing yields optimum results these starting points:

```
minimum= 1
maximum=30
```

You can tune up from there based on your hardware and configuration.

WAS's Resource Analyzer or the perfServlet EAR can be used to collect statistics during initial load testing through the performance testing tool, such as LoadRunner with the appropriate monitor for WAS. The statistics related to the data source connection pooling (Pool Size, Percentage Used, Concurrent Waiters) can be collected and used to determine the best settings for the minimum and maximum connection pool values.

You should understand that the WebSphere architecture is divided into several logical queues. These queues are used to hold work units until resources are available to process them. Typically you want to configure the queues to get *smaller* as the work units progress through the WebSphere queue system. Normally it is less expensive to hold a work unit in an upper level queue than a lower level queue. WebSphere queues are shown below in order of largest to smallest.

- Web (HTTP)

- Web Container

- EJB

- Database

FOR ADDITIONAL INFORMATION

This is a good document on understanding tuning parameters in WAS:

http://www-306.ibm.com/software/webservers/appserv/doc/v40/ae/
infocenter/was/0901.html#b206

DATABASE

It is always best during initial performance tests of the web application to have a qualified DBA monitor the DBMSs during testing for any bottlenecks such as:

- Full table scans

- Maximum connection utilization

Additionally, the DBA should collect statistics on database queries executed to assist in tuning the tables and indexes for maximum performance.

WAS v4 also uses DBMS tables for its housekeeping, so setup and tuning of this database may be required if a bottleneck is evident, however this is a rare occurrence.

Use monitoring tools such as LoadRunner's monitors to track DBMS usage values. LoadRunner comes with various monitors for several popular DBMS systems. With some monitors, even full table scans can be detected.

As a rule of thumb it is typically better to locate the database on a dedicated server. For internal WAS tables, the same node/server is best.

Also, see related DBMS information in the Oracle Insurance components list below.

INDEXES

Appropriate indexes must be in place to avoid full table scans especially on tables that are fairly large. See details on indexes necessary for the various back-end services such as Documaker Server and Docupresentment below. Additionally, the Documanage DBA Guide contains a list of default indexes that should be implemented in the various Documanage housekeeping tables.

MAXIMUM CONNECTIONS

Various database servers have different means of setting the maximum connections and related attributes. In DB2 there is the MaxAgents setting. Depending on the number of virtual users in the performance test, this attribute might need to be increased.

WEBSPHERE MQ

MQ CLIENT MODE (BINDING OR TCP/IP CHANNELS)

There are two methods that an MQ client can connect to an MQ server. Better performance is gained in the queuing layer between Oracle Insurance components if the MQ server can be accessed locally through *binding mode.* This means having the MQSeries server on the same node as the client application. The trade-off here is that not all MQ clients will be running on the same node, and running MQSeries server will consume system resources.

The other option is to connect through the network over an MQ channel via TCP/IP, which is the typical scenario if MQSeries server is on a dedicated node.

The difference between the two methods is diminished if the network is tuned and bandwidth is high between the MQ client and the MQ server.

MAXIMUM CHANNELS

MQSeries server has a maximum number of concurrent connections ("channels") that can be open. Since MQ clients (such as Oracle Insurance components, client applications, and so on) connect via TCP/IP channels, high levels of concurrent users may exceed the maximum number of channels available.

In WebSphere MQ each queue manager has a MaxChannels attribute, if not specified the default is typically 100. For performance testing this should be set relatively high, to at least four times the number of virtual users.

This setting is in the QM.INI file:

```
< Channels: >
   MaxChannels = 800
```

MAX QUEUE DEPTH

The default MAXDEPTH of 5000 for the local queues used in MQ for communications between Oracle Insurance components is typically more than enough. However, based on use-cases, some performance tests may require this number to be increased at least on the request queue (to accommodate the backlog of messages). This use-case would be one where the number of virtual users (or requestors) is relatively high (600-1000) and the response time is relatively slow due to the nature of the request.

Some configurations of Docupresentment use asynchronous processing. There are two methods of asynchronous processing, either in request, response, or both. In this setup, many requests may be put into a queue and held for any period of time — in other words, they are not immediately processed. Similarly, many responses may be put into a queue and held for any period of time. If any asynchronous Docupresentment setup is being used, the MAXDEPTH setting should be relatively high on the queues that are affected.

DOCUPRESENTMENT (IDS)

The IDS server designed as a request/response engine with *bridges* to other Oracle Insurance products, such as Documanage and Documaker Server. Each Bridge has its own services that it exercises from Docupresentment, so some performance tuning parameters are specific to the bridges being used, and some are general to the IDS server.

IDS INSTANCES

Depending on what use-case is being exercised, some requests are CPU intensive and some are I/O intensive. If the requests in the use-case are CPU intensive, some tuning can be applied to lessen the CPU usage, such as lowering the PDF compression level. If the CPU resource(s) is maximized to the point of causing contention, then the number of IDS instances should be reduced. If the requests in the use-case are disk I/O intensive or network I/O intensive, the number of instances should also be reduced to lessen the effect of contention for OS resources.

Finding the optimal number of IDS instances is therefore specific to the use-case and to the hardware that is hosting the IDS servers. There is a "sweet spot" that represents the optimal number of instances – too few or too many will result in poor response time. The results of performance testing can be plotted as a 2-dimensional graph that will have a bell-curve appearance to illustrate this effect.

An initial setting of 6-8 IDS instances per CPU is a good start. From there, tune the number of instances with repeated load testing.

SAR REQUEST

IDS has a built in request named SAR used for house-keeping tasks such as file cache maintenance and queue maintenance. Each IDS instance calls this request to itself on a regular interval and it is processed by the main request processing thread of each IDS instance. Some bridges such as Documanage Bridge also use this request to run their own house-keeping tasks by inserting functions/rules into this Request Type in the IDS configuration file.

The interval for the SAR request can be controlled which can be useful for performance tuning. This is an IDS instance that is running a housekeeping task is unavailable for processing incoming requests.

To determine if the SAR request is causing bottlenecks, collect minimum logging (TransactionTime and RuleTime) from IDS during initial performance testing. If the duration of the SAR request is excessively high or extremely low, then the interval between these requests can be increased. This will increase throughput by causing fewer interruptions in IDS instance availability.

The IDS configuration file (usually DOCSERV.INI) contains an option to set the interval between housekeeping requests. This value is in seconds.

```
< DOCSRVR >
   AutoRunInterval = 1000
```

LOGGING

IDS logging should be minimized during performance testing. Turning on TransactionTime and RuleTime has a minimal affect on performance but all other logging should be turned off. Verify during the initial testing that these settings to not affect performance.

Additionally, the default request types defined in the IDS configuration file include the rules ATCLogTransaction. This rule writes data to the srvlog.dbf and srverr.dbf. These files are usually not used, and so this rule can be commented-out of all request types. An example:

```
< ReqType:MTC>
    ; function = atcw32->ATCLogTransaction
```

DOCUMAKER SERVER, DOCUMAKER SERVER SHARED OBJECT (BRIDGE) TO DOCUPRESENTMENT

INDEXES FOR ARCHIVE

Documaker Server makes requests to Documanage using query criteria on the Line-of-Business (LOB) table (such as folder filters) as well as the OT_DOCS table (such as document filters). These criteria are used when archiving to and retrieving from Documanage. Tuning the indexes on these tables is necessary to avoid full table scans. Without the appropriate indexes, performance is affected in a very negative manner.

These filters cause WHERE clauses to be used in the Documanage Server queries to the DBMS. The LOB table should include indexes on what is defined in the Documaker Server configuration files for the FolderBy option (see below) as well as the POWER_TAG column.

```
< DMIA:cabinetname >
    FolderBy = field1[,fieldn…]
```

Also, any searches allowed by end-users on the column values generated by search interfaces should be coordinated with appropriate indexes in the LOB table.

The OT_DOCS table should contain indexes on these columns:

- ObjectClass
- DocID
- RecordID
- RecordID + DocId
- ObjectClass + DocId

Additionally, all default indexes recommended in the Documanage DBA Guide should be implemented.

INDEXES FOR WIP

Documaker Server interfaces with its WIP index table and makes queries based on how the users search for item in WIP. The appropriate indexes should exist on this table to avoid full table scans based on the way that the WIP index table is accessed. A DBA should be used to gather query statistics on this table and create the appropriate indexes. Usually an index is necessary based on the Key1, Key2, and KeyID used in WIP.

LOGGING

Logged should be greatly reduced or turned off when performance testing with the rules processor (Documaker Server and the Documaker Bridge). There should be no DPRTRC.LOG file output from Docupresentment and no trace file output in the working directory for RunRP/GenData. If any of these logging files appear the cause of the logging should be determined and turned off.

BITMAP SIZES

If MRL resources contain LOG files or other graphics, they should be checked to make sure that excessive and unnecessary settings that affect size of the graphic are not used. For example, 24-bit color bitmaps should not be used if the actual bitmap has few or no colors used. The size of the resulting output files such as print files will slow down performance, in some cases significantly.

PDF COMPRESSION

Documaker Server and other bridges can produce PDF files. The settings for these PDF files in regards to compression can affect performance. High levels of compression can use up considerable amounts of CPU resources. Lower settings on compression can cause fairly large PDF sizes.

The default for the Compression option in the PrtType:PDF control group is 2. A lower number results in less compression and larger PDF files. Conversely, higher compression yields smaller PDF files. If the average size of the PDF files is small, less than 50K, then compression 0 will increase performance. If the average size is very large, 500K or higher then a value of 2 or 3 should work. If they are mixed it is best to find a setting that provides the best performance through repeated performance tests. Possible settings are:

> 0 – no compression

> 1 – best speed

> 2 – default compression

> 3 – best compression

NOTE: The PrtType:PDF control group can appear in a number of places, such as the DAP.INI, CONFIG.INI, FSIUSER.INI, or FSISYS.INI files.

DOCUMANAGE AND DOCUMANAGE BRIDGE TO DOCUPRESENTMENT

DBMS CONNECTIONS

Documanage server maintains a pool of database connections. The default is one per DSN defined in Documanage. Under heavy load more connections are necessary. Any DSN that is defined in Documanage and is used by the Documanage Bridge or Documaker Bridge/Documanage Interface should be tuned.

Generally, the number of database connections should be set to the number of IDS instances hosting the Documanage Bridge, plus the number of IDS instances hosting Documaker Bridge using the Documanage interface, plus 1. This setting is configured in the POFFICE.INI file:

```
< DS_{DatasSurceName} >
    Connections = n
```

Additionally a DBA can monitor the database connections on the database server during load testing to determine if connections are being saturated by Documanage activity. If the connections are saturated, then additional connections can be configured in the POFFICE.INI file.

INSTANCES

Multiple instances of Documanage Servers can increase throughput if Documanage has been determined to be a bottleneck in performance.

INDEXES

See Documaker Server section on specific indexes needed for Documaker's interface to Documanage for Archive.

Most other indexes that Documanage Server needs for its internal housekeeping tables are typically created during the installation phase. However, Documanage Bridge can be used for search queries into the Documanage LOB and OT_DOCS tables, and appropriate indexes should be determined by what columns are allowed to be searched upon and what typical user searches are performed. Monitoring the tables during performance testing should catch any full table scans and collecting the queries made to the DBMS server by qualified DBAs is the best resource for fine tuning the table indexes.

iPPSJ   Most iPPSj tuning and configuration is controlled in the global.xml file. iPPSj has many options for configuring DBMS access, session management and caching. Here are the highlights of the configuration settings in the current iPPSj version that have the most affect on performance.

SESSION/STATE MANAGEMENT

iPPSj can store session data in several repositories. When using enterprise-level web application servers to host iPPSj applications, the best performance option is to use internal session management system provided by the application server. Other options such as file-system based and DBMS-based are not optimal for performance, but have their advantages. In some DBMS, using BLOB fields in DBMS-based solutions can be very costly for performance.

The global.xml configuration node for this is listed below. Setting the value to SESSION enables the use of the web application server's internal session management. Other parameters in this node are not necessary unless using other methods for session management.

```
<STATEMGT DOMAIN="docucorp.com" PORT="21" HOST="???" ROOT="????"
PASSWORD="***" USERID="????">SESSION</STATEMGT>
```

TABLE INDEXES

The tables created during the initial installation of the iPPSj base system should include default indexes needed. Normally the data in these tables does not change, so by default iPPSj caches query results from these tables.

If additional tables are added to iPPSj from further customization and the tables contain more than a few hundred rows, then appropriate indexes should be added. A DBA should be able to analyze query statements used to determine the appropriate indexes to be added.

For example, if a table named AGENT is queried on the columns FULLNAME, AGENCYCODE, OFFICEID, NAME, and AGENTCODEID then the appropriate index on these columns should be created in the database.

SENDING/RECEIVING FILES VIA QUEUES

iPPSj can send and receive files to and from IDS through the queue layer. If large files are being sent through the queue performance will be affected negatively. Ensure that all REQTYPES defined in the global.xml explicitly disable sending files via queues if this functionality is not required. This is accomplished by setting the SENDFILE and GETFILE attributes to NONE as shown below:

```
<REQTYPE DESCRIPTION="" NAME="i_AssignWIP" LOW.LEVEL.SERVICE="1005"
SENDFILE="NONE" GETFILE="NONE">
```

JNDI/JDBC PROVIDERS AND IPPSJ CACHING OPTIONS

iPPSj provides the ability to cache a number of different data entities. Some are illustrated below:

• Database query results

• XSLT transformation results

• HTML entry forms

• Static resources (such as JPG/GIF files)

Caching should never be disabled in a production environment. If custom code components of an iPPSj implementation make use of iPPSj database objects for querying tables other than those included in the base implementation, caching must be disabled at the API level.

JNDI should always be used for database access to avoid DBMS connection overhead for all tables that the web application accesses. The global.xml configuration file will tell iPPSj to use JNDI if the DATASOURCE parameter in the DSN node contains a value. Other parameters are not used when there is a value in the DATASOURCE parameter.

Here is an example:

```
<TABLE NAME="iMRLDB">
<DSN PORT="50000" HOST="youdbserver" USERID="yourdbuser"
PASSWORD="****" CLASS="com.ibm.db2.jcc.DB2Driver" DATASOURCE="jdbc/
MemoDB">jdbc:db2://yourdbserver:50000/YOURDATABASE
</DSN>
</TABLE>
```

HTML LOCATION IN THE IMAGE_VERSION TABLE

The IMAGE_VERSION table contains the IMAGE_DATA column. This column is used to indicate the location of the HTML version of a FAP file (section). The column can contain two types of data:

• Text data that references a file on the file system (using the file:// prefix), or a URL (http:// prefix) to the file location.

• BLOB data that contains the HTML form in the database

iPPSj performs better using the text referencing method instead of BLOB method.

TRANSLETS VS. XSLT

iPPSj uses XSLT templates to render the HTML presentation layer. These templates can be compiled into translets contained in JAR files if they conform to the compiler's standards. CPU usage is significantly reduced if translets are used instead of the standard text-based XSLT templates.

Translet compilation can be performed on-the-fly by the iPPSj application. This is controlled by an attribute setting for each <PRESENTATIONLAYER> node in the GLOBAL.XML file. The sample shown below creates a translet in ~/WEB-INF/classes/com/DocuCorp/jeds/translets/system:

```
<XSLT>
    <PRESENTATIONLAYER NAME="SERVICE_DEFAULT_MRLSelection"
DOTRANSLET="YES">4000</PRESENTATIONLAYER>
</XSLT>
```

This tells iPPSj to create a translet in the com/DocuCorp/jeds/translets/system directory under WEB-INF/classes. Note that the on-the-fly compilation feature is available in iPPSj version 3.x, patch 024 and higher. For prior versions, you will need to run the compilation manually.

COMPRESSION

iPPSj uses varying levels of compression that are controlled via the global.xml. iPPSj will apply the selected compression level to session state management, collection state management and cached resources. You will get optimal speed when the setting is zero (0), however, this setting requires more RAM resources. Possible settings are:

For	Enter
No serialization and no compression	0 (zero)
Serialization, but no compression	1
Serialization and compression	2 (the default)

These settings are applied to the STATE_TABLE, COLLECTION_TABLE and OBJECT_TABLE nodes using the attribute *COMPRESS="n"* as shown below:

```
<STATE_TABLE COLNAME="state_id" SQLTYPE="-4" COMPRESS="2"
OWNER="">iSession</STATE_TABLE>
```

iPPSj compression is also applied in the screens.xml configuration file to cache page templates. Caching the page templates reduces file I/O and results in faster performance, however it requires more RAM resources. A sample setting is shown below:

```
<screen id="/mrlselection" tcache="true" description="Agent Access">
```

Depending on this setting, iPPSj will cache the page template for later use. Normally in a production environment, you want to use this setting:

```
TCache=True
```

NETWORK

An inefficient and untested network can affect performance results of any web application. Back-end services such as DBMS and MQ should be on a separate network from other unrelated business processes. This will prevent unrelated network access and use from saturating bandwidth and hampering the performance of communications between the web application and the back-end services. This architecture also allows security services such as firewalls and secure socket layers to be avoided so that they do not affect performance.

Load testing tools have monitors that can also collect statistics from firewalls and network devices if necessary. Load tests can be run both inside and outside the autonomous network to compare network performance from the web client. Some monitoring tools are available that can be used to monitor web application performance and network performance in real-time and alert administrators if there is a problem detected.