This page intentionally left blank.
# Table of Contents

- Introduction ........................................................................................................................................................................ 7
- System Requirements .............................................................................................................................................................. 7
- Eclipse Plug-ins ..................................................................................................................................................................... 7
- Installation ............................................................................................................................................................................. 9
- Product Description ............................................................................................................................................................. 18
- User Interface Description ...................................................................................................................................................... 19
- Directories Tab Description .................................................................................................................................................. 19
- Plug-ins Tab Description ....................................................................................................................................................... 21
- Tools Tab Description ............................................................................................................................................................ 23
- Configuration ........................................................................................................................................................................ 25
- Using the Eclipse.cfg File for Publishing ............................................................................................................................ 25
- Eclipse.cfg Schema ............................................................................................................................................................... 27
- Configuring Source and Destination Directories .................................................................................................................. 29
- Starting and Stopping Services ............................................................................................................................................... 37
- Using Plug-Ins ........................................................................................................................................................................ 39
- HTML Plug-In ........................................................................................................................................................................ 39
- PDF Plug-In ............................................................................................................................................................................. 39
- Custom Plug-Ins ................................................................................................................................................................. 39
- Publishing To Multiple Formats .............................................................................................................................................. 42
- Publishing Multiple Files of the Same Format ...................................................................................................................... 44
- XInclude Plug-In ................................................................................................................................................................. 44
- Including Multiple Documents to a Single Document ......................................................................................................... 46
- Including Segments of Multiple Documents into a Single Document .................................................................................. 46
- Auto Indexing with the Eclipse Plus and the X3 Indexer ...................................................................................................... 49
- How To Auto-Index Using Eclipse ....................................................................................................................................... 50
- Frequently Asked Questions/Troubleshooting .................................................................................................................... 57
- Still having problems? .............................................................................................................................................................
Introduction

The Eclipse Autopublisher automatically outputs to multiple formats such as HTML, SGML, PDF, WML, and to other custom or proprietary formats through eXtensible Style Language Transformations from one, single-source XML document. Eclipse will maintain proper formatting throughout the process and output the files to a specified destination directory. Using Eclipse greatly decreases server load for organizations that use source XML and parse to HTML on the server per request.

![Figure 1 - Single-Source XML Transformed to Multiple Media Versions](image)

Using an XInclude plug-in, Eclipse can include 'segments' of selected XML documents, or even entire documents, to create an entirely new published output document! An example: to output a single PDF document and use only selected segments of 6 different XML documents, create a single XML document that 'includes' specific identifiers with the 6 XML documents, and once copied to the source directory, it will output the single PDF, essentially creating a completely new document.
This page intentionally left blank.
System Requirements

Please review the system requirements for Eclipse to ensure optimum application performance and operability.

- Working knowledge of XML and associated XSL
- Windows NT4 SP4 / 2000™ SP2 or XP™ Professional
- Minimum 128M RAM (the more the better)
- Microsoft Data Access Components 2.5 SP2 (for Windows NT 4.0 users)
- Windows Scripting Host 5.
- MSXML Parser™ 3.

Eclipse Plug-ins

Eclipse has the capability of using DocSoft-created or third-party plug-ins to allow publishing to just about any electronic format available. Below is a list of available plug-ins created by DocSoft:

- HTML Plug-in - comes standard with Eclipse installation package
- PDF Plug-in - comes standard with Eclipse installation package
- WML Plug-in - available as an add-on
- SVG Plug-in - available as an add-on
- XInclude Plug-in - available as an add-on
- X3 XML Search Engine Indexer Plug-in - available to X3 Search Engine owners only
- CHM - coming soon
- Word (*.doc) - coming soon
This page intentionally left blank.
Installation

Eclipse can be installed either from CD or through an ESD file from the DocSoft website.

NOTE: If you plan to use Eclipse in a network environment, you will need to have your organization's System Administrator setup account access to enable Eclipse to access and publish to network or webserver directories.

It is highly recommended you have this accomplished prior to installation as the installation process will ask you for account information.

1. Double-click the file named "eclipseInstall.exe" to begin the installation process.
2. Once initiated, a dialog will appear as show in the following figure. Click "Next" to begin.

![Figure 2](image)

*Continued on next page.*
3. Read through the "Eclipse Autopublisher" licensing agreement. If you agree with the terms, click on the "I accept the terms in the license agreement" radio button. Click next in order to continue the installation process.

Figure 3

Continued on next page.
4. Type a "User Name" and the "Organization" in the appropriate text fields. Click "Next" in order to continue.

Figure 4

Continued on next page.
5. Choose the set up type. Click next in order to continue in installation sequence. We recommend using the “Complete” setup type for most users.

Figure 5

Continued on next page.
6. Specify Eclipse account information by choosing one of the following radio buttons.

![Eclipse Autopublisher Setup](image)

**Eclipse Account Information**

Please specify Eclipse account information

- **Local System Account**
- **User Account**

**Account Name:**

\`:\eclipse``

**Password:**

``************``

**Password Confirmation:**

``************``

7. The "Local System Account" is set up to run on a single machine without network or server access. You do not need to setup account information with this option.

8. The "User Account" allows Eclipse to run from and/or publish to a Network or Web Server. Please contact your System Administrator to set up a user account for this type of operation. Once the account is set up, choose the "User Account" radio button and enter the account information as supplied by your system administrator.

9. Click "Next" to continue the installation process.

*Continued on next page.*
10. Click the "Install" button to begin installation. If you want to review or change any of your installation settings, click the "<Back" button. Click "Cancel" to exit setup.

Figure 7

Continued on next page.
11. During the installation process, the "Status" bar will show the status of files currently being installed.

Figure 8

Continued on next page.
12. If the installation is successful, the following dialog will appear.

![Completed Eclipse Autopublisher Setup Wizard](image)

**Figure 9**

13. Click the "Finish" button to complete the installation.
This page intentionally left blank.
Product Description

Eclipse runs as a service on the host computer and runs "in the background" formatting files in accordance with associated eXtensible Style Language Transformations (XSLTs) created by your organization or by DocSoft on a consulting basis. A user interface is provided in the Control Panel to allow the user to control different aspects of service and provide a means to add plugins for more functionality.

The following information is provided to assist the user in understanding the Control Panel interface.

Continued on next page.
User Interface Description

Eclipse runs as a service and starts as soon as the host computer is booted. There is a user interface in the Control Panel that provides the ability to choose source and destination directories, add or delete plug-ins and/or start/restart the service.

Directories Tab Description

The Directories tab provides the option to choose "Source" and "Destination" directories.

![Eclipse Autopublisher Settings](image)

Figure 10
This page intentionally left blank.
Plug-ins Tab Description

The "Plug-ins" tab displays what plug-ins are available, and provides the capability to add and delete plug-ins created by DocSoft to output to different formats.

Figure 11

NOTE: Additional custom plug-ins can be created by DocSoft to output a format to meet your specific needs. For more information on this service, please send an email to sales@docsoft.com.
This page intentionally left blank.
Tools Tab Description

The "Tools" tab contains controls to "Stop", and "Start" the service, or republish the entire source directory through the "Renew" button.

Figure 12
Configuration

The following information provides details on how to configure Eclipse and your XML files for publishing.

Using the Eclipse.cfg File for Publishing

The Eclipse.cfg configuration file provides a method of adding more functionality to publishing your documents. It allows the developer to specify how to publish, what types of files to publish, what eXtensible Style Language Transformation (XSLT) to use, and also provides custom indexing modifications for Eclipse Plus. This is specially useful if you do have or want to add extra processing instructions at the top of each XML document to be published. If processing instructions are not used, then the XML files will be published in accordance with the Eclipse.cfg file.

The Eclipse.cfg file needs to be placed in the root of the source directory(ies) as created by the Eclipse Autopublisher Configuration interface. If placed in the root source directory, it will be valid throughout all subdirectories.

A sample Eclipse.cfg file is provided below:

```xml
<?xml version="1.0" ?>
<default-settings xmlns="http://www.docsoft.com/Schemas/Eclipse/Settings">
 <publish method="htm" href="template.xsl">
  <except>*.pdf.xml</except>
 </publish>
 <publish method="wml" href="t_wml.xsl">
  <except>*.pdf.xml</except>
 </publish>
 <publish method="pdf" href="prodpdf.xsl">
  <filter>*.pdf.xml</filter>
 </publish>
 <index>
  <filter>jg*.xml</filter>
  <except>jg*_pdf.xml</except>
  <base href="/" />
  <context-id>5</context-id>
 </index>
 <plugin method="pdf" progid="Eclipse.PDFPlugin" />
</default-settings>
```

As the above example shows, there are 3 top-level elements in an Eclipse.cfg file: publish, index and plugin.

Available functionality is based solely upon the version of Eclipse you have (Lite, Standard or Plus). As shown in the example above, first publish element uses a method of "htm" (gives the output file an extension of *.htm) and uses "template.xsl" as the XSL stylesheet. The except element limits the publishing to all files except those ending in "_pdf.xml".

The third publish method in the example above uses an element of "filter", which provides a method of filtering using and asterisk (*) as a wildcard. The example shown will publish using the PDF method for all XML files ending with "_.pdf.xml".
Eclipse.cfg Schema

Below is the XSD Schema for Eclipse.cfg:

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    elementFormDefault = "qualified">
    <xsd:element name = "unnamed">
        <xsd:complexType/>
    </xsd:element>
    <xsd:element name = "default-settings">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "publish" maxOccurs = "unbounded"/>
                <xsd:element ref = "index"/>
                <xsd:element ref = "plugin"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "publish">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:choice>
                    <xsd:element ref = "filter"/>
                    <xsd:element ref = "except"/>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name = "href" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "method" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "except">
        <xsd:complexType mixed = "true">
            <xsd:choice/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "filter">
        <xsd:complexType mixed = "true">
            <xsd:choice/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "index">
        <xsd:complexType mixed = "true">
            <xsd:choice>
                <xsd:element ref = "filter"/>
                <xsd:element ref = "except"/>
                <xsd:element ref = "base"/>
                <xsd:element ref = "context-id"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "base">
        <xsd:complexType>
            <xsd:attribute name = "href" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "context-id">
        <xsd:complexType mixed = "true">
            <xsd:choice/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "plugin">
        <xsd:complexType>
            <xsd:attribute name = "progid" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "method" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```
Configuring Source and Destination Directories

The "Source" directory is the directory which is auto-scanned continuously for changes. When files are saved to, copied to, or updated and saved in this directory, this will initiate the publishing process according to the processing instructions contained within each file.

The "Destination" directory is the output directory the newly created or updated files are output to.

1. To configure what "Source" and "Destination" directories will be used in the publishing process, click "Add" under the Directory tab, as shown in the following figure.

![Eclipse Autopublisher Settings](image)

Figure 13

*Continued on next page.*
2. A dialog will appear with the option to type in the path to both directories or "Browse..." to locate the directories.

Figure 14

Continued on next page.
3. Choosing "Browse" will display the "Browse for Computer" dialog. Navigate to the directory you wish to select as the "Source" directory, the press "OK".

![Browse for Computer dialog]

**Figure 15**

4. Follow the same procedure for selecting the "Destination" directory.

*Continued on next page.*
5. After choosing the source and destination directories, the path to each appears in the respective fields, as shown in the following figure.

![Add Directory dialog box](image)

**Figure 16**

6. Selecting the "**Recursive**" checkbox will allow Eclipse to publish documents using a specific directory structure. So if a specific directory structure is required to maintain proper relational linking, copy the entire directory structure to the source directory. Eclipse will then re-create the directory structure to the selected destination directory, with each subdirectory containing the resultant formatted files.

*Continued on next page.*
7. To edit the Source and Destination directories, choose the "Edit" button on the Eclipse Autopublisher Settings interface, as shown in the following figure.

![Eclipse Autopublisher Settings](image)

**Figure 17**

*Continued on next page.*
8. The "Edit Directory" dialog appears. This displays what directories are currently in use as well as the text fields to edit which directories are chosen for publishing purposes. Selecting the "Browse" buttons allow for easy selection of each directory.

Figure 18

Continued on next page.
9. Click "OK" to make changes. When prompted by the "Eclipse Autopublisher Settings" interface, you may be required to finalize any changes in the "Tools" area by stopping and restarting the service.

Figure 19
This page intentionally left blank.
Starting and Stopping Services

The following information is provided to help the user understand the Tools tab on the "Eclipse Autopublisher Settings" interface.

Click on the "Start" tab and then click "OK" in order to finalize changes made to the original directory configuration.

![Eclipse Autopublisher Settings](image)

Figure 20
This page intentionally left blank.
Using Plug-Ins

The following information is provided to aid the end-user on how to properly create and include processing instructions in the source XML for proper formatting of resultant output files.

Eclipse reads these processing instructions to choose what output formats to publish to and to select which eXtensible Style Language (XSL) sheet(s) is/are required to perform the action. The processing instructions also allow a single-source XML file to be output to a myriad of different formats.

Eclipse creates resultant format files using the source XML file's original name. For instance, if you have a source XML file named "source.xml" and want to publish HTML and PDF documents, each resultant format file's name will be "source", shown below:

    source.htm
    source.pdf

HTML Plug-In

Using the HyperText Markup Language (HTML) plug-in, Eclipse can output an HTML document from a single source XML document or output a single HTML document (with the XInclude plug-in) from multiple source XML documents. (See XInclude for more information).

Eclipse uses XSL to parse source XML to HTML. In order to properly output resultant HTML, you must ensure that a stylesheet reference is placed in the source XML documents you wish to publish.

NOTE: The installation program includes a sample XML file and XSL to convert to HTML. You may use this as a reference as well.

Here is an example of the code needed inside an XML document to call to an eXtensible Style Language sheet. Since most XML documents are parsed to HTML for to display in a web browser, you may include a standard stylesheet processing instruction as shown below:

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="path/html_stylesheet_name.xsl"?>
```

You may also include a processing instruction in the source XML to accomplish this task as well. Sample code is provided below:

```xml
<?xml version="1.0"?>
<?xml-publish method="htm" href="path/html_stylesheet_name.xsl"?>
```

PDF Plug-In

When the Portable Document Format (PDF) plug-in is installed, Eclipse can output a PDF file from a single source XML document or output a single PDF file (with the XInclude plug-in) from multiple source XML documents. (See XInclude for more information).

NOTE: If you have purchased the PDF plug-in, sample XML and XSL files are provided to output a PDF document. The XSL is a valuable resource in understanding how to develop and XSL file to convert XML to the PDF format.

DocSoft can develop a custom XSL stylesheet for your organization on a consulting basis. Please contact sales@docsoft.com for more information.
**Custom Plug-Ins**

Eclipse can accept custom plug-ins developed by DocSoft or third parties to convert source XML to just about any electronic format available. The processing instructions required to initiate a custom plug-in is the same of using a standard plug-in, except the method attribute value must reflect the plug-in name, and a stylesheet must be developed to convert the source XML to the custom format. Please contact DocSoft Support for more information.
This page intentionally left blank.
Publishing To Multiple Formats

Eclipse can output to multiple formats from a single source XML document through the proper use of processing instructions. The following is provided to aid the developer in implementing and using multiple processing instructions to output to multiple formats.

Each processing instruction points to a specific eXtensible Style Language sheet and outputs a different file format for each format type.

Below is some sample code needed to output an HTML and PDF version of the source XML document:

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="path/html_stylesheet_name.xsl"?>
<?xml-publish method="pdf" href="path/pdf_stylesheet_name.xsl"?>
```

In the above example, the second processing instruction is a standard reference to a stylesheet. In this case, the stylesheet reference points to a stylesheet that parses the XML to HTML.

The third processing instruction is a standard Eclipse processing instruction. Notice the method tells Eclipse to publish the source XML to PDF in addition to HTML. The href attribute points to the associated XSL stylesheet that converts the source XML to the PDF format.

**NOTE:** The xml-stylesheet reference is not required. You may use only xml-publish processing instructions if you wish.

Below is sample code to publish to three different formats, HTML, PDF and WML:

```xml
<?xml version="1.0"?>
<?xml-publish method="htm" href="path/htm_stylesheet_name.xsl"?>
<?xml-publish method="pdf" href="path/pdf_stylesheet_name.xsl"?>
<?xml-publish method="wml" href="path/wml_stylesheet_name.xsl"?>
```
**Publishing Multiple Files of the Same Format**

Eclipse also offers a method of output multiple files of the same format, but with different filenames. Let's say that you have a source XML file named "mySource.xml" that you wish to publish to HTML, but the same XML file is used to create a Table of Contents. How do you accomplish this?

Using separate xml-publish processing instruction provides this capability. Just add processing instructions similar to below:

```xml
<?xml version="1.0"?>
<?xml-publish method="htm" href="path/htm_stylesheet_name.xsl"?>
<?xml-publish method="toc.htm" href="path/htm_toc_stylesheet_name.xsl"?>
```

This will create the following resultant HTML files:

- MySource.htm
- MySource.toc.htm

The results are 2 different HTML files, using one source XML file and 2 different eXtensible Style Language stylesheets.

**XInclude Plug-In**

XInclude is a W3C (World Wide Web Consortium) working draft (at publication date). The Eclipse XInclude plug-in allows developers the ability to make use of a good portion of the draft. You may view the latest version at http://www.w3.org/TR/xinclude/.

Inclusion is accomplished by merging a number of XML Infosets into a single composite Infoset. Specification of the XML documents (infosets) to be merged and control over the merging process is expressed in XML-friendly syntax (elements, attributes, URI References).

The following is provided to aid developers in understanding how to use the XInclude plug-in. Sample XML and XSL files have been included in the installation for those who have purchased this plug-in. These files can be a valuable resource for developers.
This page intentionally left blank.
Including Multiple Documents to a Single Document

To include multiple documents into a single output file, simply use the xml-publish processing instruction and the XInclude property and point to each file you want to include. Below is some sample code you may modify to suit your needs:

```xml
<?xml version="1.0"?>
<?xml-publish method="pdf" href="pdf.xsl"?>
<docs xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include href="document1.xml" />
  <xi:include href="document2.xml" />
  <xi:include href="document3.xml" />
  <xi:include href="document4.xml" />
  <xi:include href="document5.xml" />
</docs>
```

The above code includes 5 separate XML files into a single Portable Document Format (PDF) file.

Including Segments of Multiple Documents into a Single Document

The Eclipse XInclude plug-in allows developers to "pull" portions of XML documents into a separate, single resultant output file. By pointing to a specific, unique identifier within an XML file will use only the data. This is important when using "Boiler Plate" data, or data that is used several times throughout your document or series of documents. Warnings and Cautions used in technical documentation is a good example of "Boiler Plate" data. Using this method to publish documents greatly decreases upkeep on re-usable data.

This method requires an identifier be described in the associated DTD or Schema, so ensure your XML DTD or Schema defines a unique identifier, such as an id attribute, for proper operation.

To use this method, simply use the XInclude property to point to a specific identifier and an XML tag. Below is some sample code:

Source.xml:

```xml
<?xml version="1.0"?>
<?xml-publish method="pdf" href="pdf.xsl"?>
<docs xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include href="document1.xml#i2217" />
  <xi:include href="document1.xml#i457" />
  <xi:include href="document2.xml#i789" />
</docs>
```

Or you may use the following to perform the same action:

```xml
<?xml version="1.0"?>
<?xml-publish method="pdf" href="pdf.xsl"?>
<docs xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:include href="document1.xml#xpointer(/doc/table[7])" />
  <xi:include href="document1.xml#xpointer(/doc/para[22])" />
  <xi:include href="document2.xml#xpointer(/doc/table[3])" />
</docs>
```

In the example directly above, the xpointer property is used with XPath to point to a specific element within document1.xml and document2.xml. The elements is declared according to its position with like elements within the external documents, for instance, /doc/para[22] points to the 22nd para element within document1.xml.

The resultant PDF file will only use the data between the tags of the selected identifier or element. For example, let's use the following XML files for reference:
Document 1:

```xml
<?xml version="1.0"?>
<doc>
<body>
<famous_quotes>
<quote id="i456">
<person>George Orwell</person>
<para id="i00124">"Men are only as good as their technical development allows them to be."</para>
</quote>
<quote id="i457">
<person>Ralph Waldo Emerson</person>
<para id="i00125">"The world is upheld by the veracity of good men: they make the earth wholesome. They who lived with them found life glad and nutritious. Life is sweet and tolerable only in our belief in such society."</para>
</quote>
</famous_quotes>
</body>
</doc>
```

Document 2:

```xml
<?xml version="1.0"?>
<doc>
<body>
<famous_quotes>
<quote id="i789">
<person>Henry David Thoreau</person>
<para id="i01227">"Good religious men, with the love of men in their hearts, and the means to pay their toll in their pockets."</para>
</quote>
<quote id="i790">
<person>William Shakespeare</person>
<para id="i01228">"Let men say we be men of good government, being governed, as the sea is, by our noble and chaste mistress the moon, under whose countenance we steal."</para>
</quote>
</famous_quotes>
</body>
</doc>
```

Using source.xml and pdf.xsl, the output PDF file will contain the following data:

**Compiled Quotes**

<table>
<thead>
<tr>
<th>Quote</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Men are only as good as their technical development allows them to be.'</td>
<td>George Orwell</td>
</tr>
<tr>
<td>'The world is upheld by the veracity of good men: they make the earth wholesome. They who lived with them found life glad and nutritious. Life is sweet and tolerable only in our belief in such society.'</td>
<td>Ralph Waldo Emerson</td>
</tr>
<tr>
<td>'Good religious men, with the love of men in their hearts, and the means to pay their toll in their pockets.'</td>
<td>Henry David Thoreau</td>
</tr>
</tbody>
</table>

The information in red is data pulled from document1.xml. The information in green is from document2.xml.
This page intentionally left blank.
Auto Indexing with the Eclipse Plus and the X3 Indexer

The following information is provided to aid the developer in understanding how to auto-index files using the X3 Search Engine's Indexer.

Eclipse can interact with the X3 Search Engine in that it can auto-index documents according to Rules created by the user utilizing the X3 Indexer's Tag Editor. Using this functionality requires the X3 Indexer installed on the same system as Eclipse.

Using Eclipse with the X3 Search Engine provides organizations an all-inclusive publishing and searching solution. Eclipse will autopublish any documents that are copied to, or updated in the Source Directory, then auto-index those documents to keep the X3 Search Engine's index up-to-date.

Continued on next page.
How To Auto-Index Using Eclipse

To auto-index your source XML documents to the X3 index, you must add a processing instruction that points to the ID number given to a specific Region created by the X3 Indexer. To locate a specific Region ID, perform the following:

1. Launch the X3 Indexer.

![X3 XML Indexer (demo version) - DocSoft LLC](image)

**Figure 21**

*Continued on next page.*
2. Select the database in which the index resides. It is best to select the ODBC Datasource created.

Figure 22

Continued on next page.
3. If selecting via ODBC Data Source, choose "Machine Data Source" and select the DSN name. In this case, it is called "IndexerSQL.

Figure 23

Continued on next page.
4. After selecting the proper data source, login using the Use Trusted Connection option, as shown below:

![SQL Server Login](image)

Figure 24

*Continued on next page.*
5. After database has been properly select, you may then click on the "Tag Editor" button to launch the X3 Indexer Tag Editor.

![Image of X3 Indexer Tag Editor interface]

**Figure 25**

*Continued on next page.*
6. The Tag Editor interface appears. Please select the specific Region you wish to auto-index by highlighting the specific Region. Once highlighted, the Region ID appears under the "Region ID:" field on the top-right side of the dialog.

![Figure 26](image_url)

Now that you have your Region ID, you may add the processing instruction to the Region-specific source XML files. Sample code may look like the following:

```xml
<?xml version="1.0"?>
<?xml-publish method="htm" href="template.xsl" ?>
<?xml-index base="/" context-id="102"/>
```

The "base" attribute contains the URL in which the XML data to be accessed is located. In this case, our entire website is located in the root directory, therefore the value of the attribute is "/". This can also contain an absolute URL reference.

The context-id attribute's value contains the Region ID we found through the Indexer's Tag Editor interface.

Once the processing instruction is added, either save the file in the Source Directory, or copy the source XML file to the Source Directory. The source XML file will then be published according to the xml-publish processing instruction(s), then be indexed or re-indexed to the X3 Search Engine's index.
This page intentionally left blank.
Frequently Asked Questions/Troubleshooting

The following is the short list of common problems associated with the use of Eclipse.

1. Will the Eclipse Autopublisher work on Windows 95/98/Me/XP Home editions?

No. Eclipse runs as a "service" on Windows NT4/2K/XP Pro, so it needs NT Services to operate.

2. During installation, I get an error during XInclude or PDF Plug-in installation. What is wrong?

This is probably due to one of the following:

1. Security settings on the machine will not allow installation of these plug-ins
2. A feature of your anti-virus software such as "script-blocking" is not allowing installation of the software.

Please check the security and/or anti-virus settings on the host system. If the problem is anti-virus software, you may have to disable script blocking. If not sure, check with your System Administrator on what is best to do.

3. Everything has installed okay, and I have created source and destination directories, but when I copy over XML files into the source directory, nothing happens. What causes this?

Check the following items in order:

1. Ensure that the XSL stylesheet used to transform your XML exists in the source directory(ies) BEFORE you copy and/or update your XML files.
2. Ensure you have the correct link path to your XSL sheetstyle in each XML document.
3. Check to see if a "Script Blocking" feature on your host system is enabled. You may have to disable this feature on the host machine in order to have Eclipse run properly. Version 1.1 of Eclipse takes care of this problem. Please visit http://www.docsoft.com/store to download this version.

The Event Viewer under Control Panel | Administrative Tools | Application is a very helpful tool in determining problems with Eclipse. It usually describes what the problem is. When it doesn't give a specific problem, and it gives an error similar to "Run-time error [IDispatch error #3092]", then the problem is probably somewhere in your XSLT.

4. How can I develop a plug-in for Eclipse to output to a different format than those provided?

For the time being, you may use DocSoft to develop custom plug-ins for this purpose. We hope to have an SDK available later this year that will enable third party developers to develop custom plug-ins for Eclipse. Please contact sales@docsoft.com for more information.
This page intentionally left blank.
Still having problems?

Please visit our tech support page at http://www.docsoft.com/productq.asp or call us at 1.405.236.2466.