avenue.quark TUTORIAL

Table of Contents

Introduction

In this tutorial, you will:

1 Open a sample QuarkXPress Passport™ document.

2 Use avenue.quark™, along with a supplied DTD (Document Type Definition), to tag most of the content in the sample document.

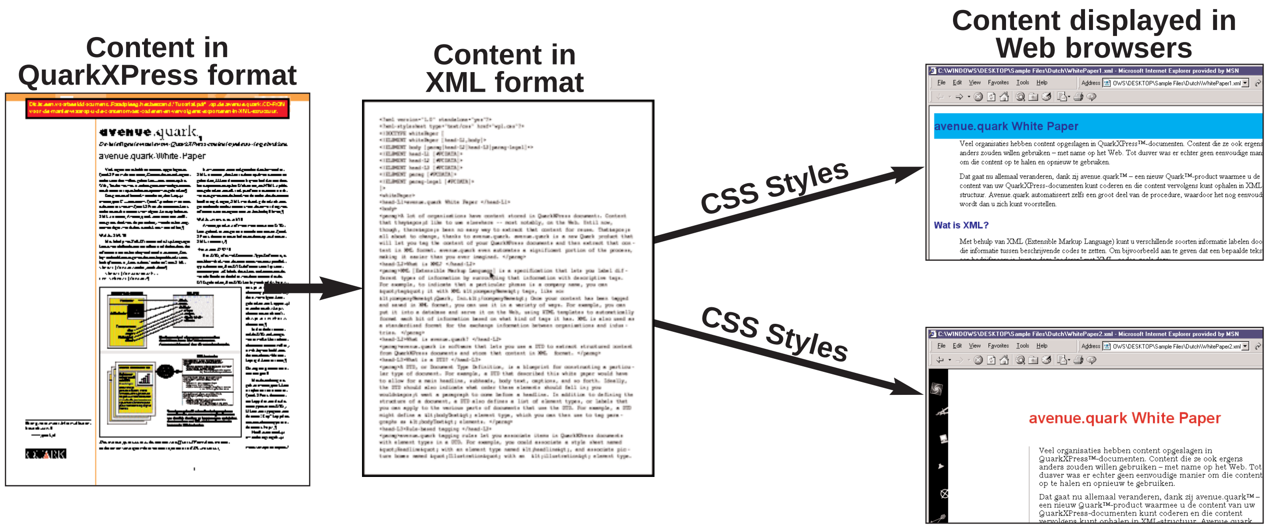
3 Save the tagged content in XML (Extensible Markup Language) format.

4 View the content in a Web browser using Cascading Style Sheets (Windows only).

5 Streamline the tagging process by first updating the tagging rule set to tag more types of content, and then combining several text boxes into a *sequence* that can be tagged as a whole.

6 Tag a picture and caption.

The advantage of avenue.quark is that it lets you tag content that is currently stored in QuarkXPress Passport format, export that content in industry-standard XML format, and then reuse that content in a variety of ways. For example, in this tutorial we will take our original sample document (on the left below), export its content in XML format using avenue.quark, and then view that content in a Web browser using two completely different sets of CSS (Cascading Style Sheet) styles (on the right below).



QuarkXPress Passport content saved as XML and viewed in a Web browser with CSS

!!! For an in-depth introduction to XML, DTDs, and avenue.quark, see Chapter 1, “avenue.quark Basics,” in “A Guide to avenue.quark” (“Guide to avenue.quark.pdf”). For more information about tagging rule sets, see Chapter 5, “Tagging Rule Sets,” in the same file.

Preparing to Use avenue.quark

Before you begin this tutorial, please take the following steps:

1. Verify that you have QuarkXPress Passport 4.04 or later.

2. Verify that the avenue.quark QuarkXTensions™ module and the Item Sequence QuarkXTensions module are in the "XTension" folder in your QuarkXPress Passport application folder. For installation instructions, see the avenue.quark installation instruction booklet or the "Installation Instructions.pdf" file.

3. Verify that the file named "WhitePaper - English.xmt" is in the "Templates" folder in your QuarkXPress Passport application folder.

4. Copy the folder "avenue.quark Tutorial" to the desktop of your computer so you can get to it quickly and easily.

You may want to print this tutorial, so that you can easily refer to it while you are going through the steps.

Using Rule-Based Tagging

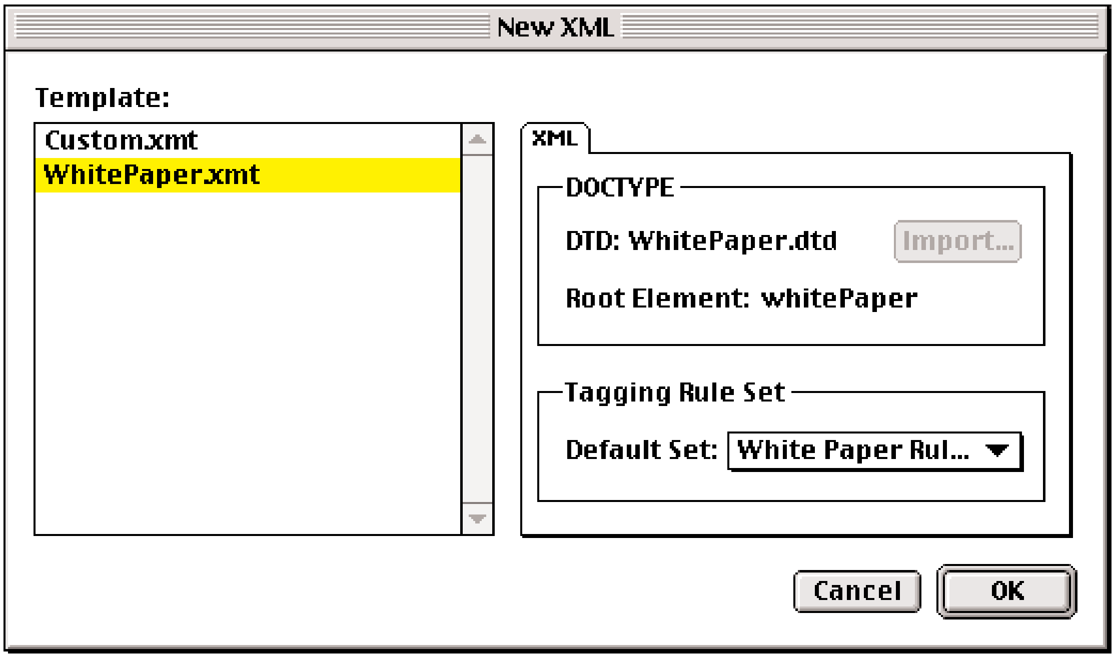
This section shows you how to use a feature called *rule-based tagging* to automate much of the XML tagging process. Rule-based tagging relies on a *tagging rule set,* which is a list of guidelines for automatically tagging QuarkXPress Passport content based on the content's QuarkXPress Passport style sheets and other formatting attributes.

To use a previously created tagging rule set to tag the contents of a text box in a sample document:

1 Launch QuarkXPress Passport.

2 Choose **File ➝ Open;** locate the file named "SampleDoc.qxt" in the "avenue.quark Tutorial" folder on your desktop.

3 Choose **File ➝ New ➝ XML** to create a new XML document. The **New XML** dialog box is displayed.


New XML

Template:
Custom.xmt
WhitePaper.xmt

XML

DOCTYPE
DTD: WhitePaper.dtd
Root Element: whitePaper

Tagging Rule Set
Default Set: White Paper Rul...

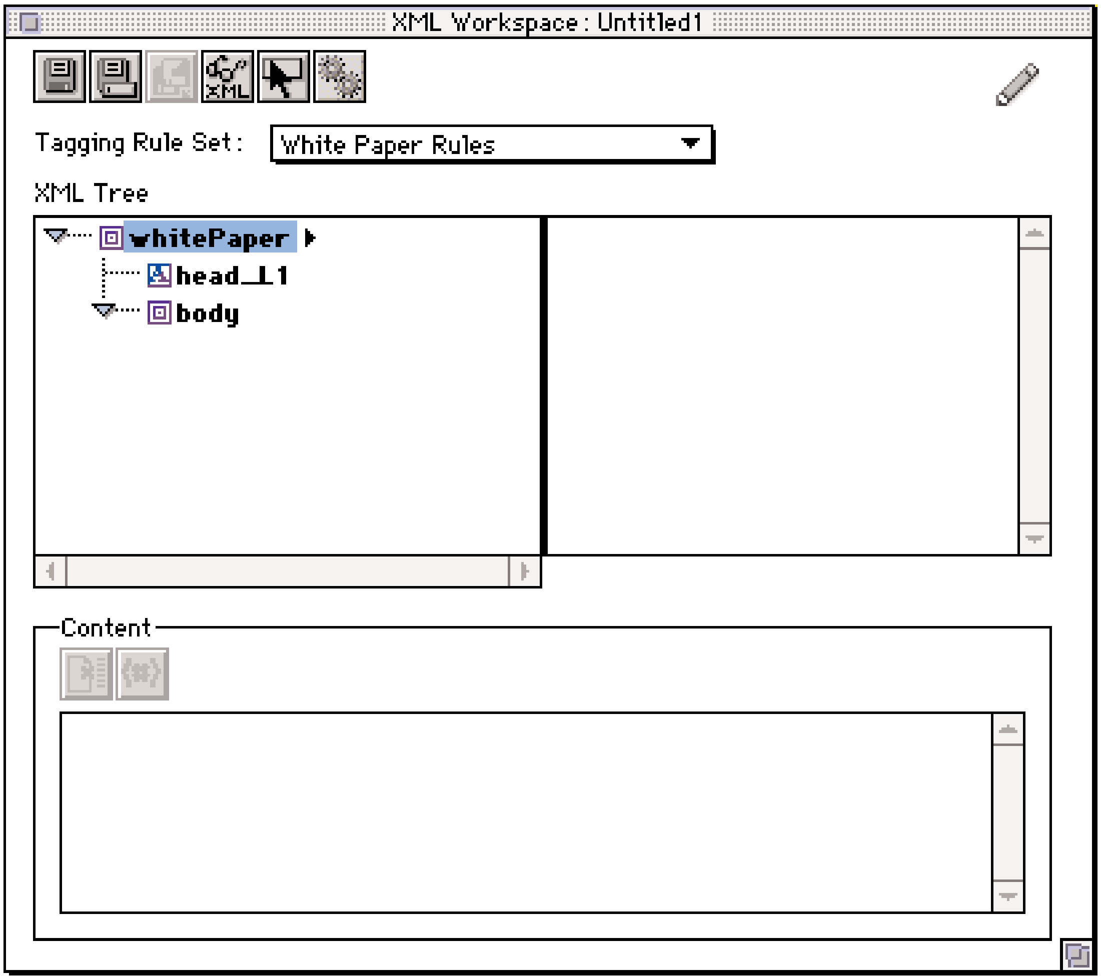
New XML dialog box

!!! If the **XML** item is not shown on the **File** ➛ **New** submenu, avenue.quark may not be correctly installed. See “Installation Instructions.pdf” for instructions on how to install the software.

4 Select **WhitePaper - English.xmt** in the **Template** list to indicate that you want to base the new XML document on the template named “WhitePaper - English.xmt.” This template specifies that you want to use a DTD named “WhitePaper.dtd”, with `<whitePaper>` as the root element, and that you want to use a preconfigured tagging rule set named “White Paper Rules.”

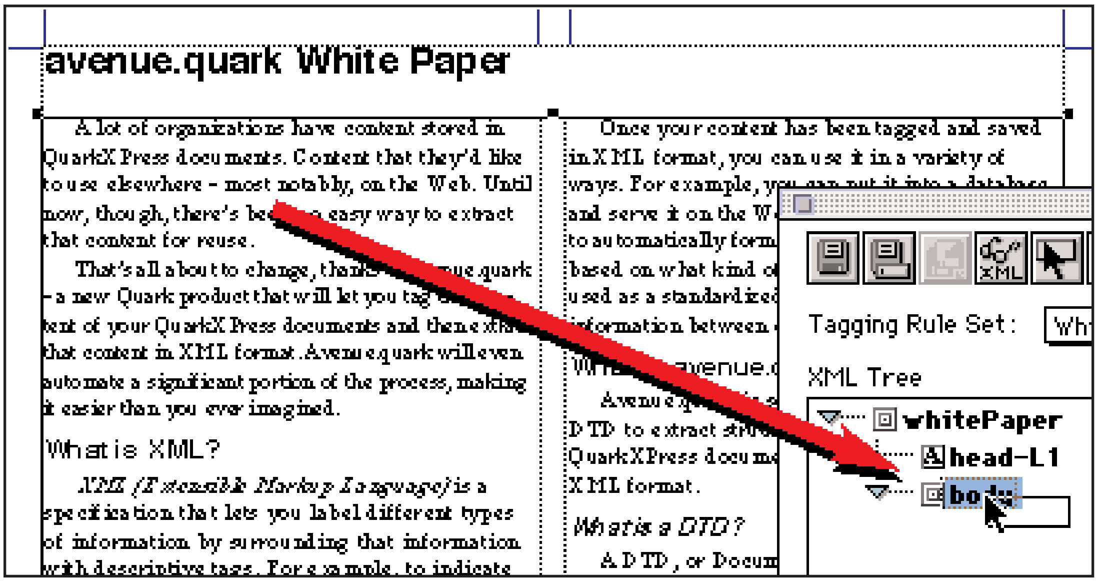
What is a DTD? A DTD, or Document Type Definition, defines a set of element types (tag definitions) and structure rules for a particular type of document. For example, a DTD for a news story might specify that each story must have exactly one section of text tagged as a `<headline>` element, that each story may or may not have a `<byline>` element, that each illustration must be immediately followed by exactly one `<caption>` element, and so forth. For more information about DTDs, see Chapter 1, “avenue.quark Basics,” in “Guide to avenue.quark.pdf.”

5 Click **OK.** The **XML Workspace** palette is displayed, and any mandatory elements in the WhitePaper DTD (in this case, `<whitePaper>`, `<head_L1>`, and at least one `<body>`) are automatically added to the **XML Tree** list.



XML Workspace palette

6 Select the text box that begins with "A lot of organizations have content...." Then press ⌘ (Mac OS) or Ctrl (Windows) and drag the text box to the <body> element in the **XML Tree** list. (Element names are displayed as bold names in the **XML Tree** list, like this: **body**.) Avenue.quark automatically tags the content in the text box, creating new elements as necessary.



avenue.quark White Paper

A lot of organizations have content stored in QuarkXPress documents. Content that they'd like to use elsewhere – most notably, on the Web. Until now, though, there's be ... easy way to extract that content for reuse.

That's all about to change, thank ... ue.quark – a new Quark product that will let you tag ... tent of your QuarkXPress documents and then ... that content in XML format. Avenue.quark will even automate a significant portion of the process, making it easier than you ever imagined.

What is XML?

XML (Extensible Markup Language) is a specification that lets you label different types of information by surrounding that information with descriptive tags. For example, to indicate

Once your content has been tagged and saved in XML format, you can use it in a variety of ways. For example, y ... and serve it on the W ... to automatically form ... based on what kind o ... used as a standardized ... information between ... Wh ... avenue.q ... Avenue.q ... DTD to extract str ... QuarkXPress docume ... XML format.

What is a DTD?

A DTD, or Docum

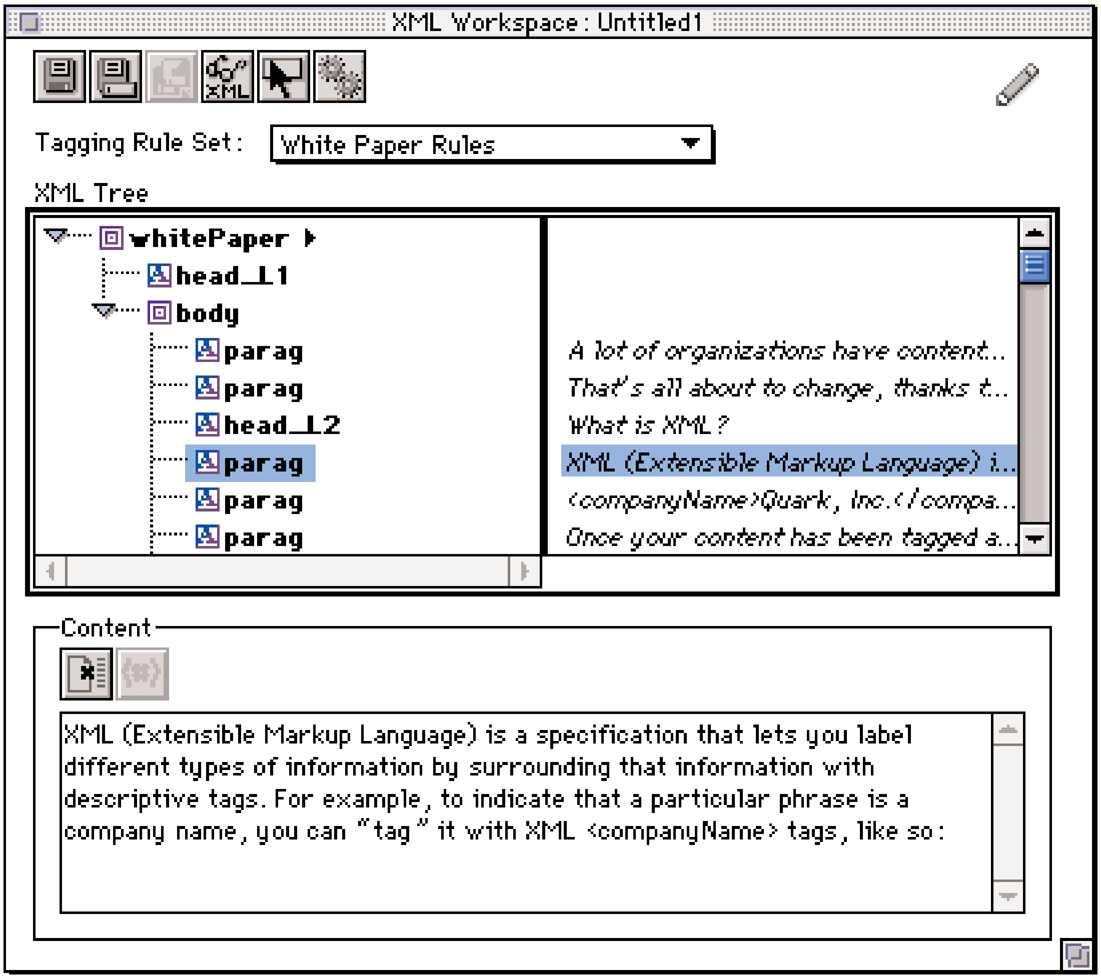
Rule-based tagging

What just happened? Avenue.quark just used the "White Paper Rules" tagging rule set to tag the content in the text box you dragged onto the <body> element. You'll learn more about tagging rule sets in the "Adding a Rule to a Tagging Rule Set" section of this tutorial.

Wait, it didn't work! If the **Choose Rule/Position** dialog box displays when you drag the text box to the <body> element, click **Stop.** Then choose **Edit** → **Preferences** → **avenue.quark**; make sure the **Always insert repeating elements at the end of the current branch** box is checked; click **OK**; and then close the **XML Workspace** palette and go back to Step 3.

If something else unexpected happened when you dragged the text box onto the `<body>` element in the **XML Tree** list, don't worry about it. Just close the **XML Workspace** palette and go back to Step 3. You may want to make sure the "WhitePaper - English.xmt" file from the "avenue.quark Tutorial" folder is in the "Templates" folder inside your QuarkXPress Passport application folder.

7 Scroll through the **XML Tree** list (in the **XML Workspace** palette) and note how the white paper's content has been tagged with the appropriate element types. Subheads have been tagged as `<head_L2>` elements, and body text paragraphs have been tagged as `<parag>` elements. You can see the full contents of an element by clicking the element's name and looking at the **Content** field.


XML Workspace : Untitled1
Tagging Rule Set :
White Paper Rules
XML Tree
whitePaper
head_L1
body
parag
parag
head_L2
parag
parag
parag
A lot of organizations have content...
That's all about to change, thanks t...
What is XML ?
XML (Extensible Markup Language) i...
<companyName>Quark, Inc.</compa...
Once your content has been tagged a...
Content

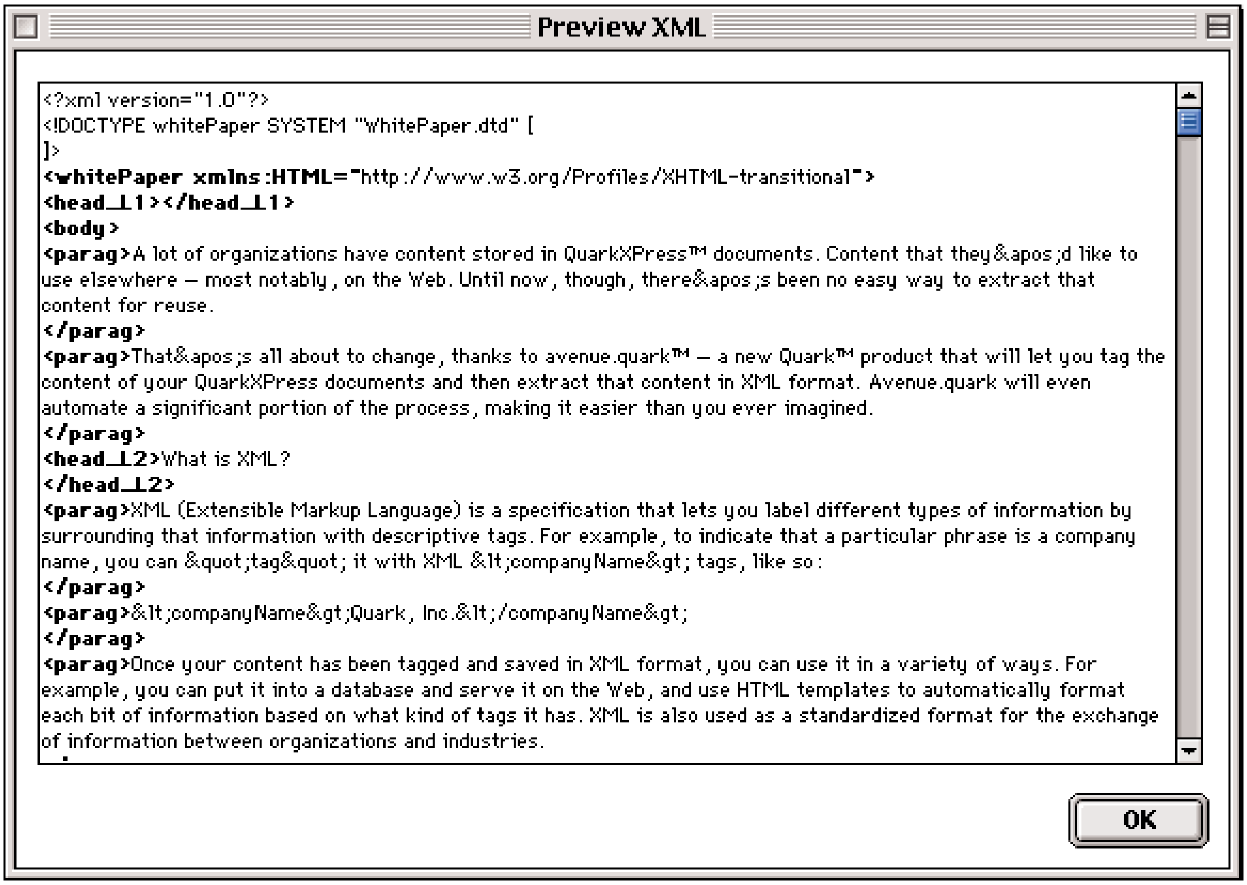
XML (Extensible Markup Language) is a specification that lets you label different types of information by surrounding that information with descriptive tags. For example, to indicate that a particular phrase is a company name, you can "tag" it with XML <companyName> tags, like so:

XML Workspace palette with **Content** field

!!! Four things that *didn't* get tagged are the headline ("avenue.quark White Paper"), the picture, the picture's caption, and the legal text that makes up the last two paragraphs on page two. You'll learn how to tag these pieces just as easily later in this tutorial.

8 Click the **Preview XML** button at the top of the **XML Workspace** palette to display the **Preview XML** dialog box. This lets you see what the exported XML file will look like.

Preview XML button



```
<?xml version="1.0"?>
<!DOCTYPE whitePaper SYSTEM "WhitePaper.dtd" [
]>
<whitePaper xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional">
<head_L1></head_L1>
<body>
<parag>A lot of organizations have content stored in QuarkXPress™ documents. Content that they&apos;d like to
use elsewhere — most notably, on the Web. Until now, though, there&apos;s been no easy way to extract that
content for reuse.
</parag>
<parag>That&apos;s all about to change, thanks to avenue.quark™ — a new Quark™ product that will let you tag the
content of your QuarkXPress documents and then extract that content in XML format. Avenue.quark will even
automate a significant portion of the process, making it easier than you ever imagined.
</parag>
<head_L2>What is XML?
</head_L2>
<parag>XML (Extensible Markup Language) is a specification that lets you label different types of information by
surrounding that information with descriptive tags. For example, to indicate that a particular phrase is a company
name, you can &quot;tag&quot; it with XML &lt;companyName&gt; tags, like so:
</parag>
<parag>&lt;companyName&gt;Quark, Inc.&lt;/companyName&gt;
</parag>
<parag>Once your content has been tagged and saved in XML format, you can use it in a variety of ways. For
example, you can put it into a database and serve it on the Web, and use HTML templates to automatically format
each bit of information based on what kind of tags it has. XML is also used as a standardized format for the exchange
of information between organizations and industries.
```

Preview XML dialog box

What am I looking at? The **Preview XML** dialog box shows you what your content looks like in XML format. Each chunk of information in an XML file is bracketed by tags that describe what that content is. For example, near the top of the screen shot above, you can see that the first paragraph in the document has been placed between an opening `<parag>` tag and a closing `</parag>` tag. This makes it easy for a wide variety of applications to identify the body copy and handle it appropriately — for example, by displaying it in a 12-point, serif font.

9 Click **OK** to close the **Preview XML** dialog box.

10 Click the **Save** button at the top of the **XML Workspace** palette, and save the document in the "avenue.quark Tutorial" folder as "MyDoc.xml." (Leave the **Encoding** pop-up menu set to **UTF8** and check **Save XML as Standalone.**) You will view this file in a Web browser in the next section, "Viewing Extracted XML Content — Windows only."

Save button

11 Click the close button on the **XML Workspace** palette to close the XML document.

Viewing Extracted XML Content

Once you've extracted content from a QuarkXPress Passport document and stored it in an XML file, you can do a wide variety of things with that content. For example, you can use Cascading Style Sheets (CSS) to view it in a Web browser as a formatted document. In this section, we'll first view a couple of sample XML files with CSS formatting, then show you how you can add CSS formatting to the XML document you saved in the previous section.

What is CSS? CSS is a way of applying style to tagged content. It was developed for use with HTML, but can be used with XML as well. It's useful because it lets you change the formatting of a large number of documents by updating a single CSS file.

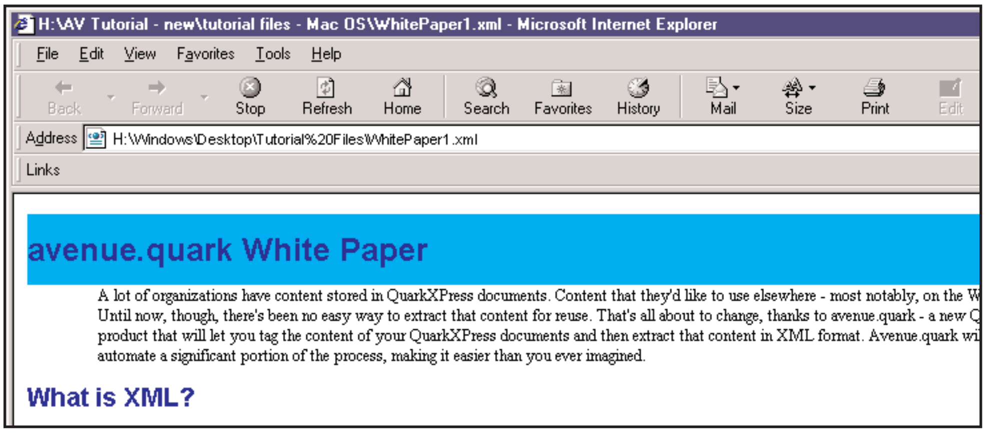
At the time of this writing, this section works only with Microsoft Internet Explorer 5.0, which is available only for Windows. By the time you read this, though, other Web browsers (such as Internet Explorer 5.0 for Mac OS) may be able to display XML with CSS styles. If you do not have a browser that can display XML with CSS styles, please skip to the next section, "Adding a Rule to a Tagging Rule Set."

Viewing an XML file with CSS style sheets

So that you can easily see what an XML file looks like when viewed with CSS, we've included two sample XML files with this tutorial. These files were generated from the same sample document you worked with in the previous section, and they're identical except for one thing: Each uses a different CSS file to provide its presentation. To view and compare the files:

1 Launch a Web browser that can display XML with CSS styles, such as Microsoft Internet Explorer 5.x.

2 Open the sample file named "WhitePaper1.xml." The browser displays the XML file using the CSS styles in the file named "wp.css." (Your exported XML file should appear similar when you view it with this CSS file in the next subsection.)


H:\AV Tutorial - new\tutorial files - Mac OS\WhitePaper1.xml - Microsoft Internet Explorer

File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit
Address
Links

H:\Windows\Desktop\Tutorial%20Files\WhitePaper1.xml

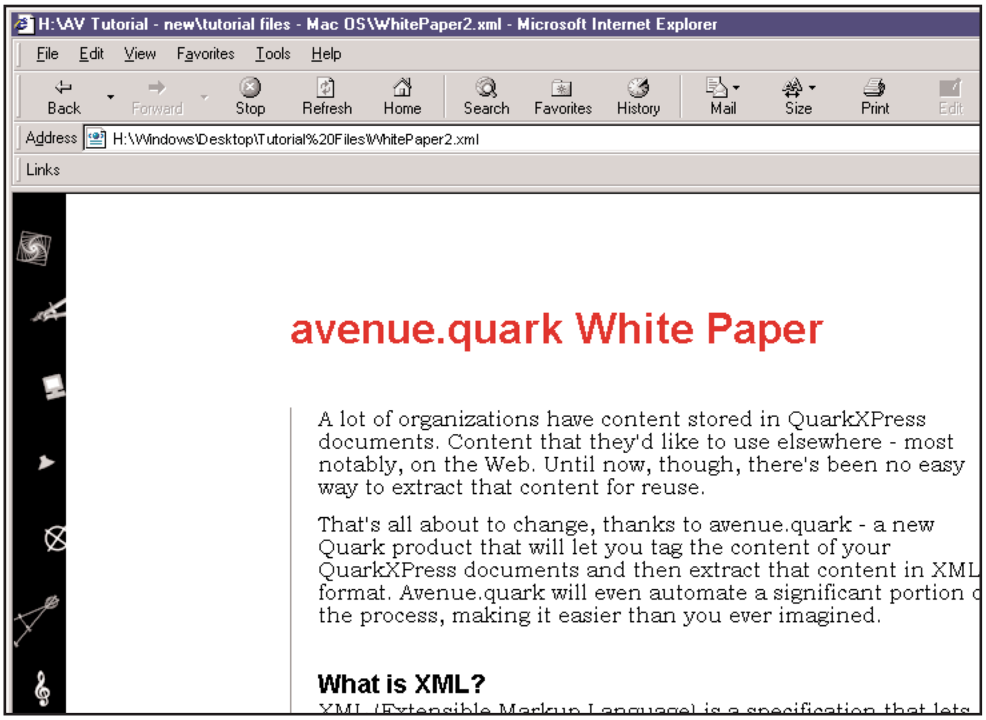
avenue.quark White Paper

A lot of organizations have content stored in QuarkXPress documents. Content that they'd like to use elsewhere - most notably, on the W
Until now, though, there's been no easy way to extract that content for reuse. That's all about to change, thanks to avenue.quark - a new Q
product that will let you tag the content of your QuarkXPress documents and then extract that content in XML format. Avenue.quark wi
automate a significant portion of the process, making it easier than you ever imagined.

What is XML?

The XML version of the white paper, displayed using the CSS styles in "wp.css"

3 Open the sample file named "WhitePaper2.xml" in the Web browser. The browser displays the same XML content, but this time using the CSS styles in the file named "wp2.css." (Your exported XML file should appear similar when you view it with this CSS file in the next subsection.)


H:\AV Tutorial - new\tutorial files - Mac OS\WhitePaper2.xml - Microsoft Internet Explorer

File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit
Address
Links

H:\Windows\Desktop\Tutorial%20Files\WhitePaper2.xml

avenue.quark White Paper

A lot of organizations have content stored in QuarkXPress documents. Content that they'd like to use elsewhere - most notably, on the Web. Until now, though, there's been no easy way to extract that content for reuse.

That's all about to change, thanks to avenue.quark - a new Quark product that will let you tag the content of your QuarkXPress documents and then extract that content in XML format. Avenue.quark will even automate a significant portion o the process, making it easier than you ever imagined.

What is XML?

XML (Extensible Markup Language) is a specification that lets

The same XML file, displayed using the CSS styles in "wp2.css"

Viewing your XML file with CSS styles

To view the XML file you created in the first section:

1 Using a text editor or a word processor other than Microsoft Word, open "MyDoc.xml" (the document you saved in Step 10 of the first section).

!!! **Why not use Word?** Some versions of Microsoft Word will automatically replace two hyphens in a row with an em dash. If possible, use a "plain text" word processor such as SimpleText (Mac OS) or WordPad (Windows). To open the file in WordPad: Launch WordPad; choose **File ➛ Open**; choose **All Documents** from the **Files of type** pop-up menu; select "MyDoc.xml"; and then click **Open**.

2 Add the following line, immediately after the first line of the XML file. This line indicates that the XML document should be viewed with the style sheet named "wp.css." Make sure you copy the line exactly.

```
<?xml-stylesheet type="text/css" href="wp.css"?>
```

3 Add the following line, immediately after the line you just added. This line indicates the beginning of a comment; we'll use it to "comment-out" the DTD embedded in the file. This is necessary because some Web browsers have trouble reading internal DTDs.

```
<!--
```

4 Add the following line, immediately after the line that reads "]>". This line indicates the end of the commented-out DTD.

```
-->
```

5 Save the file in ASCII (plain text) format. Make sure you don't change the document's name.

6 Make sure there is a copy of the “wp.css” file (provided with the avenue.quark Tutorial) in the same folder as “MyDoc.xml.”

7 Open “MyDoc.xml” in your Web browser. The CSS styles defined in “wp.css” are applied to the XML.

8 To see how the XML file looks when viewed with the CSS styles defined in “wp2.css,” reopen it in your text editor or word processor and change “wp.css” to “wp2.css,” like so:

```
<?xml-stylesheet type="text/css" href="wp2.css"?>
```

9 Save the file in ASCII (plain text) format. Make sure you don’t change the document’s name.

10 Make sure there is a copy of the “wp2.css” file (provided with the avenue.quark Tutorial) in the same folder as “MyDoc.xml.”

11 Click the Web browser’s refresh button to reload the XML file. The CSS styles defined in “wp2.css” are applied to the XML. Note the difference; see how the same XML file can have a radically different appearance depending on how it’s presented.

12 Close your Web browser and your text editor or word processor.

!!! If you look at the WhitePaper DTD in a text editor, and you’re somewhat familiar with how DTDs are constructed, you may notice that the `<whitePaper>` element has an attribute named “xmlns:HTML” with a fixed value of “http://www.w3.org/TR/REC-html40.” This attribute is automatically included in all XML files based on this DTD. Its purpose is to alert Microsoft Internet Explorer version 5.0 or later that the document uses the HTML namespace. Since the document uses the HTML namespace, the tag `<HTML:img>` is interpreted as an HTML `<IMG>` tag; thus, you’re able to display images referenced by the XML.

Adding a Rule to a Tagging Rule Set

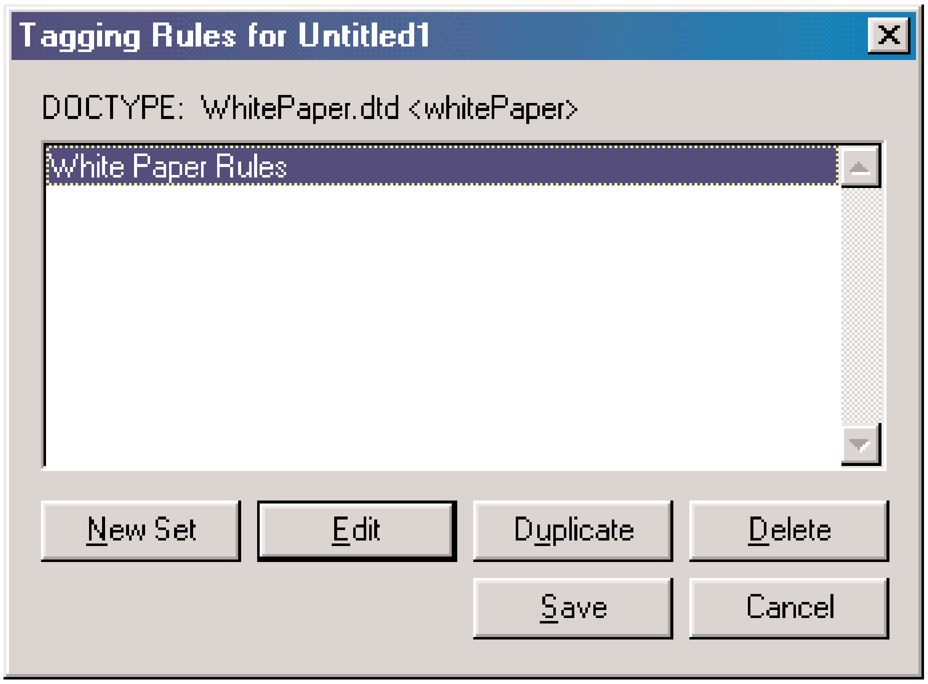
In the first section, we used an existing tagging rule set to automatically tag most of a sample document. However, the legal text at the end of the document was *not* tagged. One of the reasons is because the supplied tagging rule set does not contain a rule to deal with text that uses the "Legal Text" paragraph style sheet. This section shows you how to solve this problem by adding a new rule.

!!! When this tutorial mentions "paragraph style sheets," it's referring to QuarkXPress Passport style sheets, as opposed to CSS styles.

To add a new rule to the "White Paper Rules" tagging rule set:

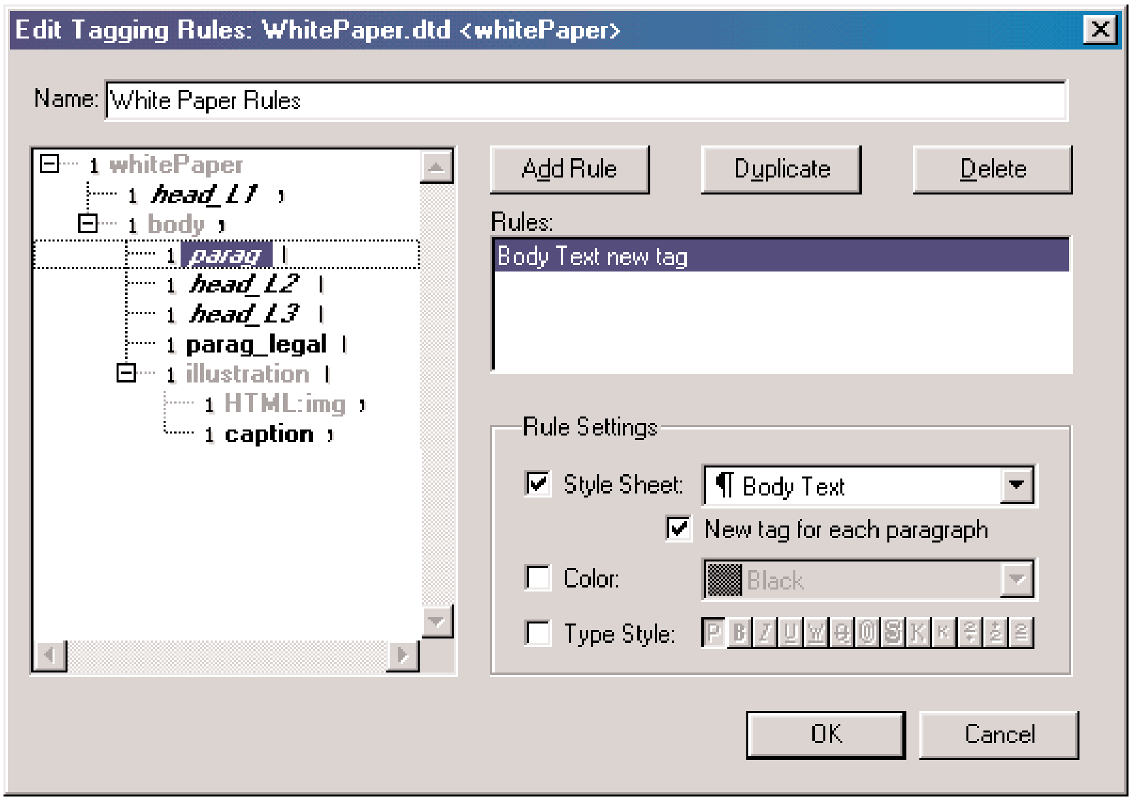
1. Launch QuarkXPress Passport. If QuarkXPress Passport is already running, close any open **XML Workspace** palettes.
2. Open "SampleDoc.qxt" if it is not already open.
3. Choose **File** ➝ **New** ➝ **XML** to create a new XML document. The **New XML** dialog box is displayed. Select **WhitePaper - English** in the **Template** list, then click **OK** to display the **XML Workspace** palette.

4 Choose **Edit** ➛ **Tagging Rules.** The **Tagging Rules** dialog box is displayed.


Tagging Rules for Untitled1
DOCTYPE: WhitePaper.dtd <whitePaper>
White Paper Rules
New Set
Edit
Duplicate
Delete
Save
Cancel

Tagging Rules dialog box

5 Select the tagging rule set named "White Paper Rules." (This is the tagging rule set we used in the first section.) Then click **Edit** to edit this tagging rule set using the **Edit Tagging Rules** dialog box.


Edit Tagging Rules: WhitePaper.dtd <whitePaper>

Name: White Paper Rules

```
⊟ 1 whitePaper
    1 head_L1
  ⊟ 1 body
      1 parag
      1 head_L2
      1 head_L3
      1 parag_legal
    ⊟ 1 illustration
        1 HTML:img
        1 caption
```

Rules: Body Text new tag

Rule Settings

- [✓] Style Sheet: ¶ Body Text
- [✓] New tag for each paragraph
- [ ] Color: Black
- [ ] Type Style:

Edit Tagging Rules dialog box

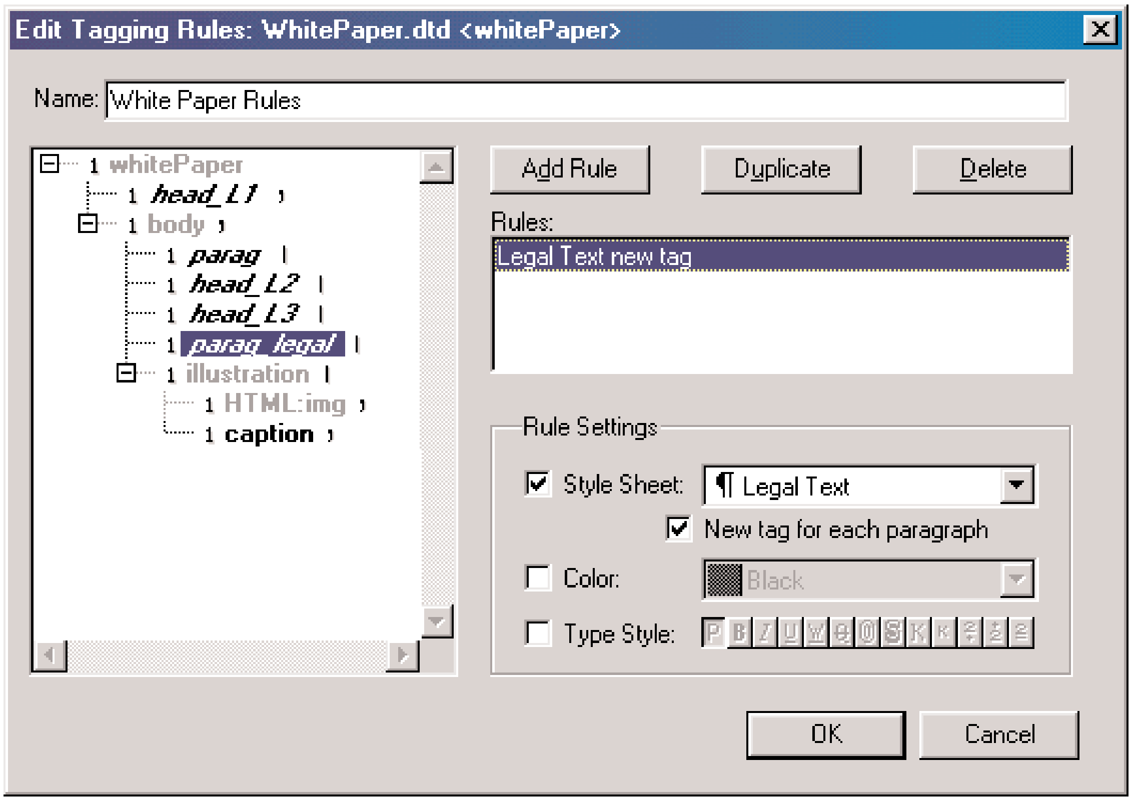
6 The list on the left side of the dialog box displays the WhitePaper DTD as a hierarchy. You can display and hide the contents of container elements and attributes by clicking the ▷ and ▽ icons (Mac OS) or the ⊞ and ⊟ icons (Windows). To view the rules that make up this tagging rule set, click the element names "head_L1," "parag," "head_L2," and "head_L3" in this list. The rules that apply to each element type are displayed in the **Rules** list on the upper right.

What do tagging rules do? Tagging rules let you tell avenue.quark that you want content that is formatted in a particular way to be tagged with a particular element type. For example, the rule displayed above indicates that paragraphs that use the "Body Text" paragraph style sheet should be tagged as `<parag>` elements.

7 Click the element named `<parag_legal>` in the list on the left, and then click **Add Rule** to create a new rule for this element type. This indicates that you want to create a new tagging rule for the `<parag_legal>` element type.

Why "parag_legal"? XML tags can be named as descriptively as you like (within certain limitations). In this DTD, a `<parag>` element is designed to contain a paragraph, and a `<parag_legal>` element is designed to contain a paragraph of legal information.

8 Check the **Style Sheet** check box and choose **Legal Text** from the **Style Sheet** pop-up menu. Check the **New tag for each paragraph** check box. This indicates that you want the `<parag_legal>` element type to be applied to paragraphs that use the "Legal Text" paragraph style sheet and that you want each paragraph that uses the "Legal Text" style sheet to be tagged as a separate element. Note that the rule in the **Rules** list (at the upper right) updates to display your selections.


Edit Tagging Rules: WhitePaper.dtd <whitePaper>

Name: White Paper Rules

```
⊟ 1 whitePaper
    1 head_L1
  ⊟ 1 body
      1 parag
      1 head_L2
      1 head_L3
      1 parag_legal
    ⊟ 1 illustration
        1 HTML:img
        1 caption
```

Rules: Legal Text new tag

Rule Settings
- [✓] Style Sheet: ¶ Legal Text
- [✓] New tag for each paragraph
- [ ] Color: Black
- [ ] Type Style:

Edit Tagging Rules dialog box with a new rule for the `<parag_legal>` element type

9 Click **OK** to save the changes you've made to the tagging rule set named "White Paper Rules."

10 Click **Save** to close the **Tagging Rules** dialog box.

!!! Make sure you leave the active **XML Workspace** palette and the active QuarkXPress Passport document open before proceeding — you'll need them for the following sections.

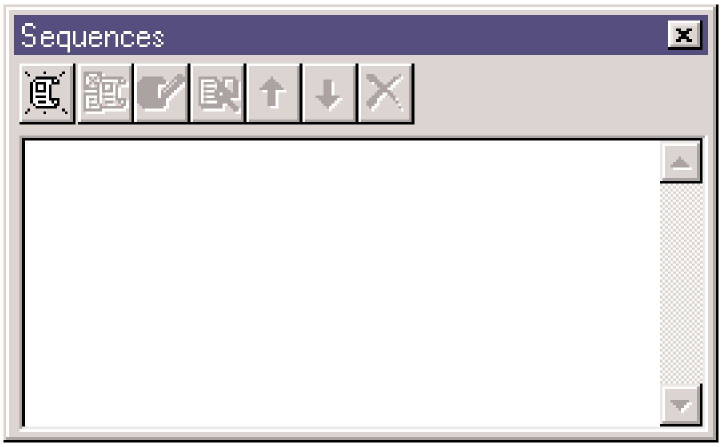
Combining Text Boxes Into a Sequence

You've already seen that avenue.quark can help you to quickly and easily tag and extract the contents of a single text box. Now we'll make the tagging process even easier by combining the series of text boxes that make up the body of the white paper into a *sequence,* and then tagging that sequence as a single unit.

!!! Sequences are a feature of Item Sequence QuarkXTensions software, which ships with avenue.quark. To use sequences, you must have Item Sequence QuarkXTensions software installed in the "XTension" folder inside your QuarkXPress Passport application folder.

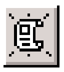
To combine a series of text boxes into a sequence:

1 Choose **View → Show Sequences** to display the **Sequences** palette.



The **Sequences** palette lets you create named sequences of text boxes.

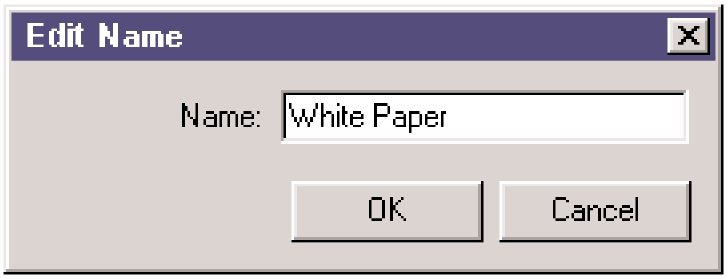
2 Click the **New Sequence** button to create a new sequence.



New Sequence button

3 Click the **Edit Name** button to display the **Edit Name** dialog box. Type `White Paper` in the **Name** field and then click **OK.**

Edit Name button



The **Edit Name** dialog box lets you rename the selected sequence.

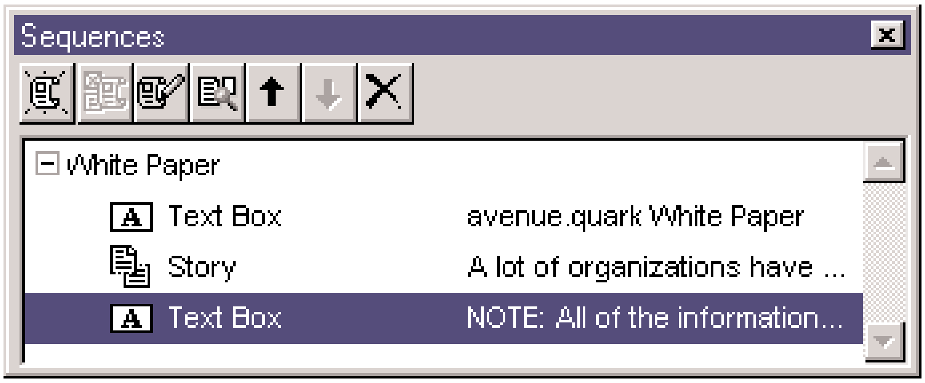
4 Click the text box containing the headline (“avenue.quark White Paper”) to select it.

5 Click the **Add Item** button to add the text box to the “White Paper” sequence.

Add Item button

6 Select the text box that begins with “A lot of organizations have content....” Then click the **Add Item** button to add it to the sequence, right after the headline box.

7 Go to page 2 and select the text box at the bottom right of the page. This text box contains the legal text for the document. Click the **Add Item** button to add this box to the sequence, right after the box that contains the bulk of the white paper's content.



White Paper

Text Box	avenue.quark White Paper
Story	A lot of organizations have ...
Text Box	NOTE: All of the information...

The **Sequences** palette displays a sequence as a list of named boxes.

The text boxes that make up the white paper have now been combined, in the proper order, into a single sequence named "White Paper." This makes it easier for you to deal with them as a unit. In the next section, we'll export the sequence in XML format.

!!! Make sure you leave the **Sequences** palette, the active **XML Workspace** palette, and the active QuarkXPress Passport document open before proceeding — you'll need both for the next section.

Tagging with Sequences and a New Rule

Now that you've added a new tagging rule for the legal text at the end of the white paper and combined all of the text boxes that make up the white paper into a single sequence, you'll see how easy it is to tag all of the important text in the document in one pass.

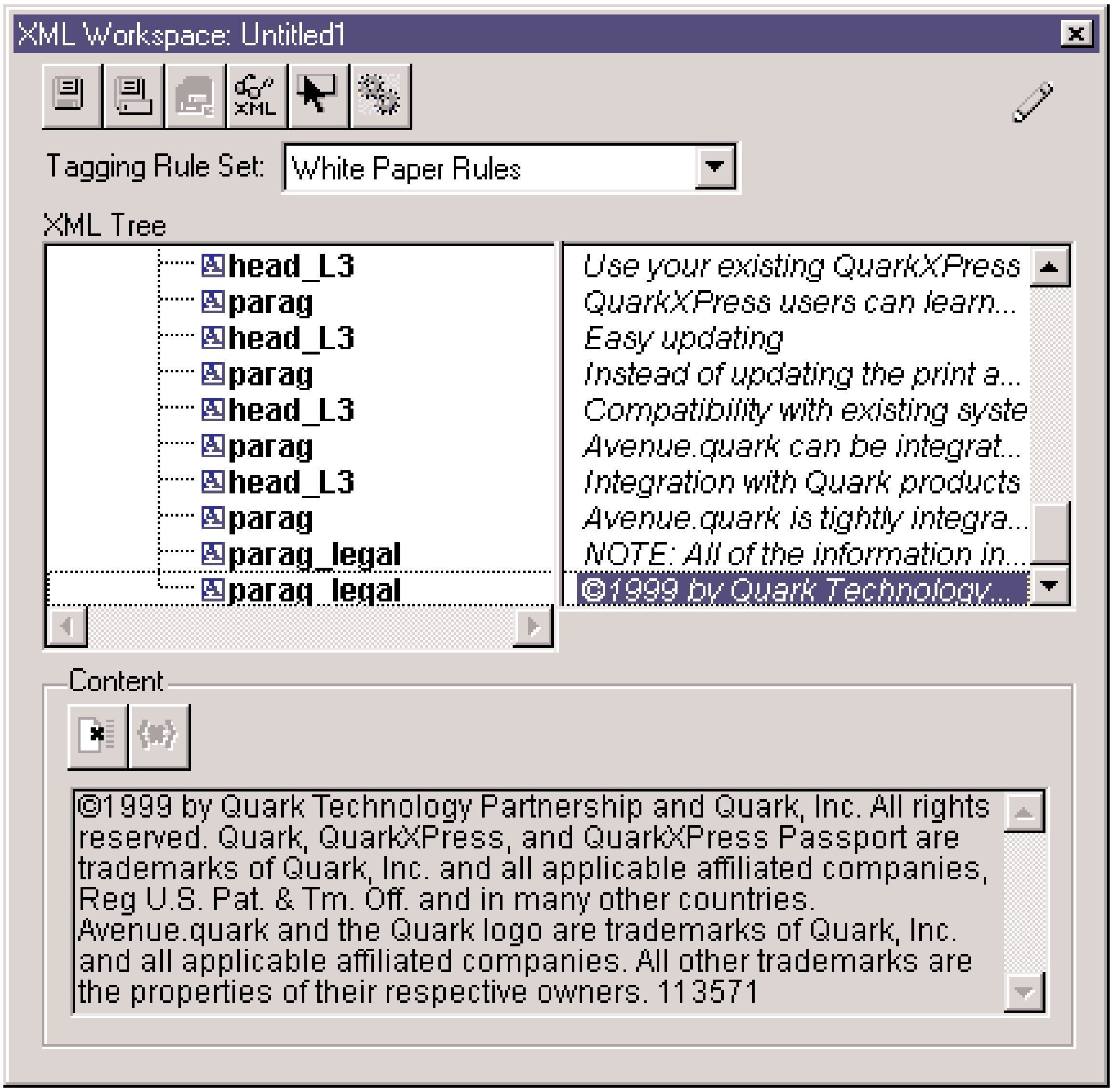
To tag the "White Paper" sequence:

1 Select the sequence named "White Paper."

2 Press ⌘ (Mac OS) or Ctrl (Windows) and drag the sequence onto the `<whitePaper>` element in the **XML Workspace** palette. Avenue.quark automatically tags the content in all three text boxes, creating new elements as necessary.

Wait, it didn't work! If something unexpected happened when you dragged the sequence onto the `<whitePaper>` element in the **XML Tree** list, don't worry about it. Just close the **XML Workspace** palette and go back to Step 1.

3 Scroll through the **XML Tree** list until you get to the bottom. Notice that this time, avenue.quark tagged all the paragraphs that use the "Legal Text" paragraph style sheet.


XML Workspace: Untitled1
Tagging Rule Set:
White Paper Rules
XML Tree
head_L3
parag
head_L3
parag
head_L3
parag
head_L3
parag
parag_legal
parag_legal
Use your existing QuarkXPress
QuarkXPress users can learn...
Easy updating
Instead of updating the print a...
Compatibility with existing syste
Avenue.quark can be integrat...
Integration with Quark products
Avenue.quark is tightly integra...
NOTE: All of the information in...
©1999 by Quark Technology...
Content

©1999 by Quark Technology Partnership and Quark, Inc. All rights reserved. Quark, QuarkXPress, and QuarkXPress Passport are trademarks of Quark, Inc. and all applicable affiliated companies, Reg U.S. Pat. & Tm. Off. and in many other countries. Avenue.quark and the Quark logo are trademarks of Quark, Inc. and all applicable affiliated companies. All other trademarks are the properties of their respective owners. 113571

Tagged document including legal text tagged as `<parag_legal>`

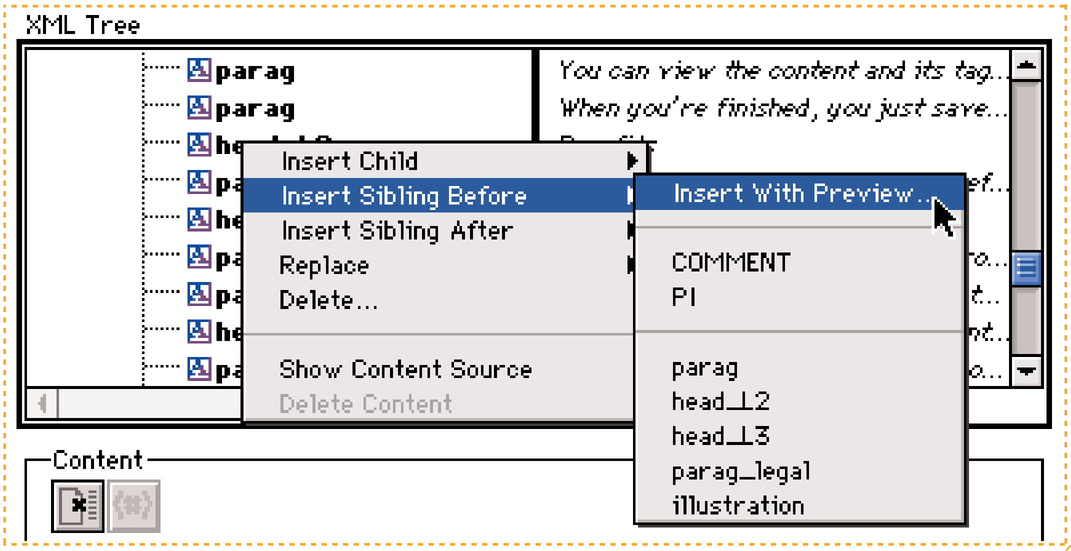
4 Leave both the QuarkXPress Passport white paper document and the new XML document open; you'll continue to work with them in the next section.

Tagging a Picture and Caption

The only important parts of the white paper that we haven't tagged yet are the picture and caption in the lower left corner of page 1. In this section, you'll add a new `<illustration>` element to the XML document, and then add the picture and its caption to that element's children.

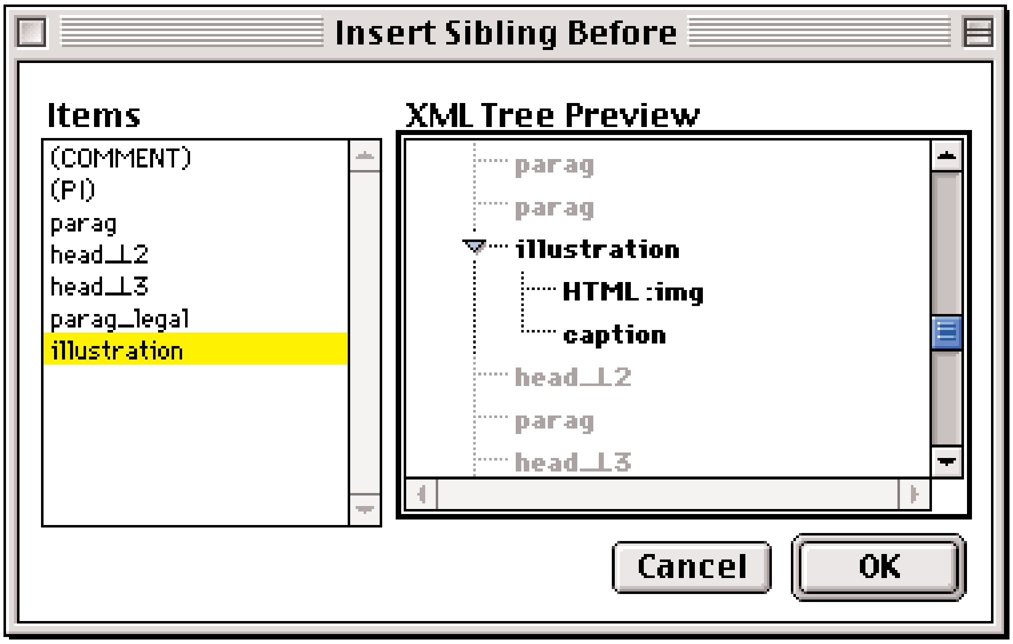
1 In the **XML Workspace** palette, scroll through the **XML Tree** list until you can see the `<head_L2>` element that contains the word, "Benefits."

2 Control+click (Mac OS) or right+click (Windows) on this element to display a pop-up menu. Choose **Insert Sibling Before** to display a sub-menu, and then choose **Insert With Preview.** The **Insert Sibling Before** dialog box is displayed.



The **XML Tree** pop-up menu lets you insert an item before the item selected in the **XML Tree** list.

3 In the **Items** list, select **illustration** to indicate that you want to insert an `<illustration>` element. The **XML Tree Preview** list displays a preview of what the current branch of the XML document will look like after the `<illustration>` element is inserted. Scroll through the **XML Tree Preview** list down until you can see the bold `<illustration>` element.

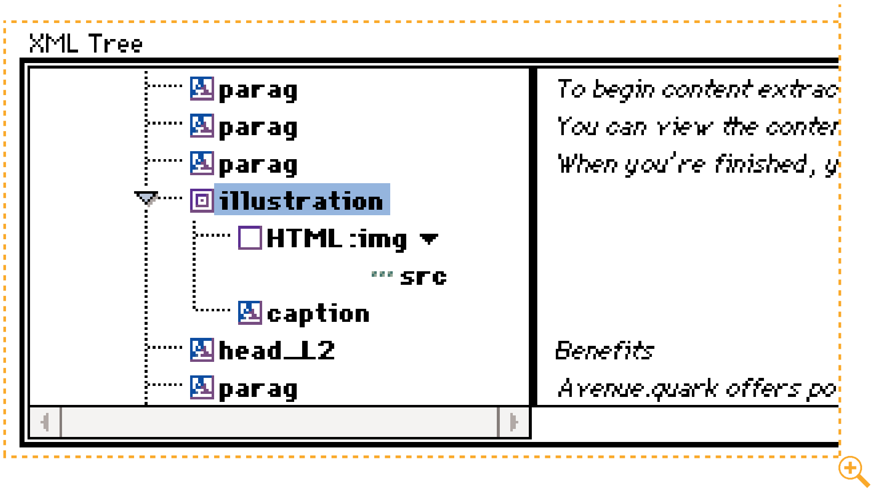


The **Insert Sibling Before** dialog box lets you view any mandatory children of the item you are inserting.

The **XML Tree Preview** list lets you view any mandatory children of the element you want to insert. The mandatory children of the `<illustration>` element are an `<HTML:img>` element and a `<caption>` element.

4 Click **OK.** The `<illustration>` element is added before the selected `<head_L2>` element.

5 In the **XML Tree** list, click the ▸ next to the `<HTML:img>` element to display the `src` attribute associated with that element.



The newly-inserted `<illustration>` element has two children: an `<HTML:img>` element and a `<caption>` element. The `<HTML:img>` element has an attribute named `src`.

6 Select the text box under the picture at the lower left on page 1. Then select the **Item** tool ✥, press ⌘ (Mac OS) or Ctrl (Windows), and drag the text box onto the element named `<caption>` in the **XML Tree** list. The text of the caption is copied into the `<caption>` element.

7 Select the picture box above the caption box. Then press ⌘ (Mac OS) or Ctrl (Windows) and drag the picture box to the attribute named “src” in the **XML Tree** list. The name of the picture file is copied into the `<HTML:img>` element’s `src` attribute.

What just happened? You just told avenue.quark to put the picture file's name in the `src` attribute for the `<HTML:img>` element. Only the file name was inserted, not the picture itself; this means the picture file must be in the same folder as the XML file when it is viewed, or the picture won't display in the browser.

8 Click the **Save** button on the **XML Workspace** palette, and save the document in the "avenue.quark Tutorial" folder on your hard disk as "MyDoc2.xml." (Again, leave the **Encoding** pop-up menu set to **UTF8** and check **Save XML as Standalone.**)

Save button

9 Use the procedure described in "Viewing your XML file with CSS style sheets" (in the section named "Viewing Extracted XML Content — Windows only") to view "MyDoc2.xml." Make sure the file named "diagram1.gif" is in the same folder as "MyDoc2.xml" if you want the picture to be visible.

Now you've seen a few of the things that avenue.quark can do. For more information, see "A Guide to avenue.quark" ("Guide to avenue.quark.pdf"), provided with the avenue.quark Tutorial.