QuarkXPress Server 8

Overview

Welcome to the QuarkXPress® Server Web Integration Guide (WIG). The WIG describes the QuarkXPress Server interface and includes sample applications that demonstrate how to build a solution that integrates with QuarkXPress Server or QuarkXPress Server Manager.

Supported interfaces

The WIG describes two separate interfaces:

- HTTP: Provides the ability to interact with the server using URLs that contain calls or point to XML files that contain calls. Client applications can be written in any language that supports HTTP requests.
- Web services: Provides the ability to interact with the server via Web services using the QuarkXPress Server Manager object model. Client applications can be written in Java™, .NET, or any other programming language that can consume SOAP-based Web services.

Note:

If you want to develop a custom load balancer or a custom application in Java using the WIG Object Model, JDK™ 1.5 is required.

Note:

If you want to use Web Objects in ASP.NET / Visual C#®, the .NET 1.1/2.0 framework with development environment (Visual Studio®) is required.

The Dynamic Publishing Process (DPP)

Dynamic Publishing Process (DPP) is the process in which QuarkXPress Server opens a project, loads content, modifies a layout, examines the project, converts data into a particular render type, and then closes the project.

The process has the following different stages:

-
- Pre-Processing Stage: During this stage an initial setup of the project is done, such as create style sheets, color, or H&J rules.
-
- Content Loading Stage: During this stage dynamic content is loaded into the boxes in the QuarkXPress project.
-
- Layout Modification Stage: QuarkXPress Server modifies the layout of the project during this stage. For example, QuarkXPress Server changes the angle of the boxes by 30 degrees.
-
- Post-Processing Stage: In this stage no modifications to the project are made. QuarkXPress Server examines the constructed project and performs bookkeeping tasks.

WIG vs. XTensions® Developer Kit (XDK)

The WIG allows Web developers to build client-side applications that use the features available in QuarkXPress Server. The XDK allows software developers to implement features that are not available in QuarkXPress Server, such as server-side processing and application-specific services. Note that the

QuarkXPress Server XDK requires knowledge of C or C++.

Where do I go from here?

For an introduction to writing applications with the HTTP interface, see the Getting started: HTTP page.

For an introduction to writing applications with the Web services interface, see the Getting started: Web services page.

New and enhanced features in QuarkXPress Server 8

QuarkXPress Server 8 includes a number of new and improved features. This section lists those new or enhanced features that affect the WIG.

Rubi text support

Rubi text clarifies the meaning or pronunciation of base text and is commonly used in Japanese typography. Base text can run vertically or horizontally, and rubi text usually follows the direction of the base text. Modifier SXT now supports the deconstruction and modification of rubi text constructed in QuarkXPress 8 and construction of new rubi text in document construction and modification. Note that rubi text present in a project can't be modified.

An element named RUBI has been added to the Modifier DTD, as a sibling to RICHTEXT, allowing the characteristics of rubi text to be specified. If Annotation is set to true for rubi text, original glyphs in rubi are replaced from the equivalent annotation glyph in the font (if font supports annotation or there is no change). For details about this element, see The Annotated Modifier DTD

.

Support for font sets

Font sets let you control how different types of characters — such as alphabetic (Roman) and Han characters — display when they occur together in text.

You can create a new font set in the QuarkXPress 8 Edit Font Set dialog box (
Edit > Font Sets > New). Each font set is composed of a set of font types, each of which has its own settings.

Modifier SXT now supports font sets applied to text in QuarkXPress on deconstructing a document using the "XML" render type, and when modifying (or constructing) a document using Modifier XML. Note that you can't modify the fontset created in a project, and you can't modify the fontset applied on saved text in a project.

FONTSET is added as a RICHTEXT attribute in the Modifier DTD.

Support for double strikethrough

QuarkXPress 8 adds the new text formatting option
double strikethrough. It is supported as an attribute of the RICHTEXT element in Modifier XML.

Support for emphasis marks

QuarkXPress 8 adds the capability to add emphasis marks as a text attribute to text. Modifier SXT now supports QuarkXPress 8 emphasis marks as an attribute to the RICHTEXT element.

Story direction

In QuarkXPress 8, you can position text so that it runs left-to-right and top-tobottom or top-to-bottom and right-to-left. In Modifier XML, the story direction can be specified through the TEXT@STORYDIRECTION attribute.

Sending support

Sending lets you fix the distance between the left edges of successive character bounding boxes in horizontal text, or the top edges of successive character bounding boxes in vertical text. You can apply sending in QuarkXPress by selecting text and entering an explicit measurement (such as 2mm or 8q) in the Track Amount field in the Classic or
Character Attributes tab of the Measurements palette. Sending is now also supported in Modifier SXT as an attribute of RICHTEXT (similar to the implementation of tracking support).

Grouped character support

The Grouped Character feature can be used to include a group of horizontal characters, such as Roman characters, within a vertical line of text. Grouped characters always display horizontally and do not break at the end of a line.

```
<!ELEMENT GROUPCHARACTERS ((RICHTEXT | HIDDEN)+)>
<!ATTLIST GROUPCHARACTERS
SCALEDIRECTION (HORIZONTAL | VERTICAL) #IMPLIED
SCALEAMOUNT CDATA #IMPLIED
SENDING CDATA #IMPLIED
TRACKAMOUNT CDATA #IMPLIED
>
```

The GROUPCHARACTERS element specifies one or more RICHTEXT (or HIDDEN text) elements to be grouped together in the text flow. You can specify the scale and sending of the grouped text characters as attributes to this element.

Support for new OpenType text styles

You can apply an OpenType(R) style to characters to display different, specially designed, or repositioned glyphs within the current font. For example, you can apply Fractions to access specific fraction glyphs instead of manually formatting fractions by resizing and repositioning existing characters. Likewise, applying Standard Ligatures represents characters according to ligatures available in the font. (See "Using ligatures" for more information.) You can apply many styles in combination, although some, such as Superscript and Subscript, are mutually exclusive.

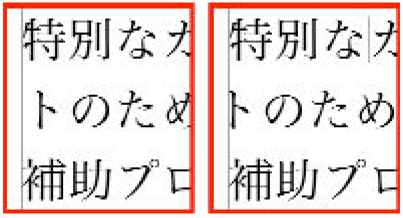
Since version 7.2, QuarkXPress Server has supported the application of OpenType styles, which are provided as attributes to the RICHTEXT element. These have been supplemented with the new OpenType styles that have been added to QuarkXPress 8.

OTGlyphs are special glyphs created by combining multiple characters - e.g 3/5 can be represented in a single glyph depending upon the fonts used. These glyphs are available in the Glyph palette in QuarkXPress.

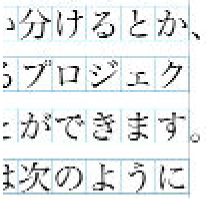
In the Modifier DTD, we support using new attributes OTFEATURE and OTVARIANT under the RICHTEXT node. Attribute OTFEATURE contains the value of the OpenType feature applied on text such as AlternateFractions (afrc), AlternateAnnotations, etc. The OTVARIANT attribute shows which variant to use, among the multiple matches found.

Support for hanging character sets

Hanging character sets handle both hanging punctuation and margin alignment. Margin alignment lets you hang characters partially outside the margin to create visually uniform text alignment along the margin. Hanging punctuation lets you hang punctuation characters fully outside the margin so that the text is flush against either a uniform margin at the beginning of a line of text (leading) or against a uniform margin at the end of a line of text (trailing). For example, the period in the second sample text below is hanging outside the trailing margin.



The second line in this sample text shows no hang on the left, but shows a leading hang on the right.



The punctuation characters in this sample text are trailing hanging characters.

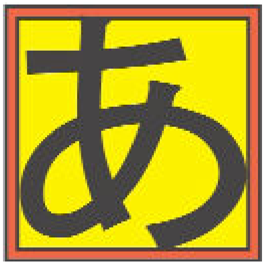
You can create custom hanging character classes and hanging character sets in QuarkXPress, or you can use the default classes and sets that come with the software. A hanging character class is a group of characters that should always hang outside the margin or indent inside the margin by the same percentage. A hanging character set is a group of hanging character classes. You can use a hanging character set to apply one or several hanging character classes to paragraphs.

In Modifier XML, the hanging characters set applied to text is specified by the FORMAT@HANGINGCHARACTERS attribute.

Support for Character Alignment

The Character Alignment feature gives you several options for aligning small characters in a line of text to the largest character in a line of text. You can align characters based on their baselines, their em boxes, or their ICF boxes.

Em boxes are the bounding boxes of characters. The ideographic character face (ICF) box is a boundary inside the em box beyond which a glyph cannot extend. ICF boxes are necessary to ensure that glyphs in an East Asian text flow do not touch each other. The red area in the diagram below represents the boundaries of the em box. The yellow area represents the ICF box.

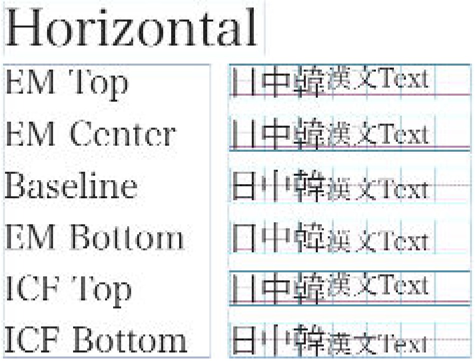


Red represents the em box. Yellow represents the ICF box.

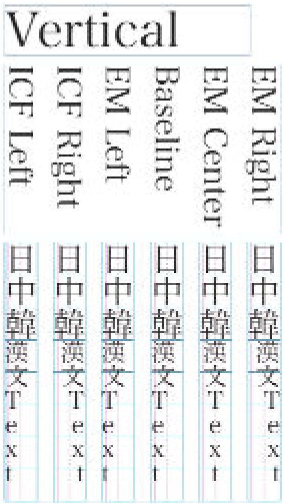
The alignment options available in QuarkXPress through the Style> Character Alignment submenu are available in Modifier XML through the FORMAT@CHARACTERALIGNMENT attribute, which can have the following values:

-
- ICFBOXTOP
- Aligns small characters with the top of the ICF box.
-
- ICFBOXBOTTOM
- Aligns small characters with the bottom of the ICF box.
-
- ICFBOXLEFT
- Aligns small characters with the left side of the ICF box.

-
- ICFBOXRIGHT
- Aligns small characters with the right side of the ICF box.
-
- EMBOXTOP
- Aligns small characters with the top edge of the em box of the largest character in a line of horizontal text.
-
- EMBOXBOTTOM
- Aligns small characters with the bottom edge of the em box of the largest character in a line of horizontal text.
-
- EMBOXCENTER
- Aligns small characters with the center of the em box of the largest character.
-
- EMBOXLEFT
- Aligns small characters with the left side of the em box of the largest character.
-
- EMBOXRIGHT
- Aligns small characters with the right side of the em box of the largest character.
-
- ROMANBASELINE
- Aligns small characters with the baseline of the largest character.

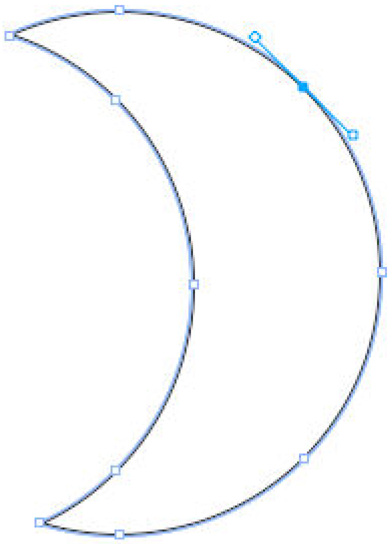


Examples of horizontal and vertical character alignment

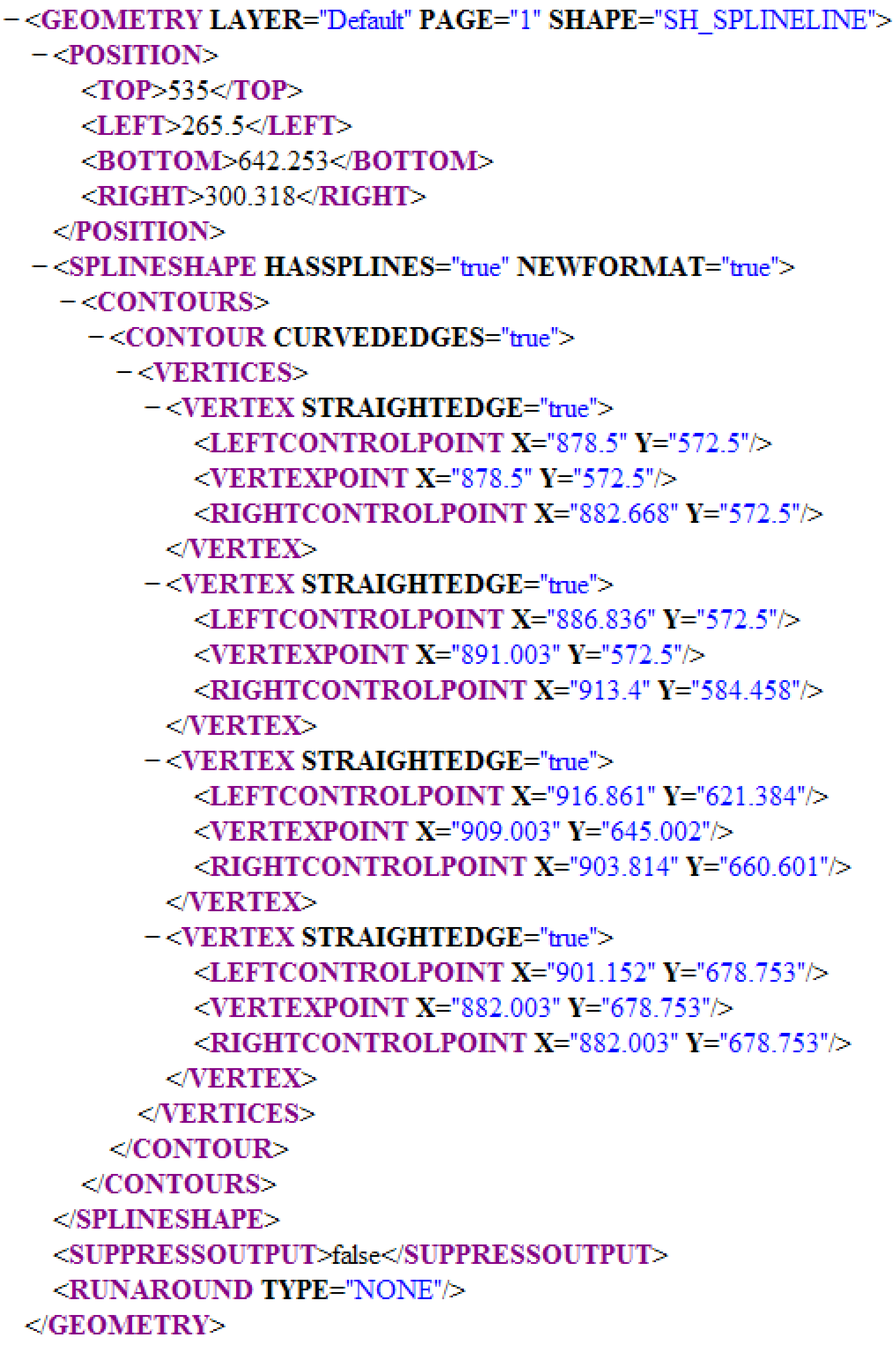


Support for Bézier curves

Previous versions of QuarkXPress Server only supported the modification of the content and of the geometry of the bounding rectangle of a Bézier shape or curve created in QuarkXPress. This also meant that it was not possible to construct a Bézier curve or shape using the Construct namespace, as there was no description in the Modifier XML format of the box shape, only its bounding rectangle. This meant that any shapes would be lost in a workflow which involved deconstructing and then reconstructing a document to Modifier XML. In QuarkXPress Server 8.0, the shape of a Bézier box is not lost with a construct request. The DTD provides all support to maintain the shape of a Bézier box in a construct request



For example, for a crescent shape like that pictured here (shown in QuarkXPress, with the drawing points visible), the Modifier XML to represent it shows both a regular position element, representing the box geometry, and then one or more contours, each with multiple vertices.



```
- <GEOMETRY LAYER="Default" PAGE="1" SHAPE="SH_SPLINELINE">
  - <POSITION>
      <TOP>535</TOP>
      <LEFT>265.5</LEFT>
      <BOTTOM>642.253</BOTTOM>
      <RIGHT>300.318</RIGHT>
    </POSITION>
  - <SPLINESHAPE HASSPLINES="true" NEWFORMAT="true">
    - <CONTOURS>
      - <CONTOUR CURVEDEDGES="true">
        - <VERTICES>
          - <VERTEX STRAIGHTEDGE="true">
              <LEFTCONTROLPOINT X="878.5" Y="572.5"/>
              <VERTEXPOINT X="878.5" Y="572.5"/>
              <RIGHTCONTROLPOINT X="882.668" Y="572.5"/>
            </VERTEX>
          - <VERTEX STRAIGHTEDGE="true">
              <LEFTCONTROLPOINT X="886.836" Y="572.5"/>
              <VERTEXPOINT X="891.003" Y="572.5"/>
              <RIGHTCONTROLPOINT X="913.4" Y="584.458"/>
            </VERTEX>
          - <VERTEX STRAIGHTEDGE="true">
              <LEFTCONTROLPOINT X="916.861" Y="621.384"/>
              <VERTEXPOINT X="909.003" Y="645.002"/>
              <RIGHTCONTROLPOINT X="903.814" Y="660.601"/>
            </VERTEX>
          - <VERTEX STRAIGHTEDGE="true">
              <LEFTCONTROLPOINT X="901.152" Y="678.753"/>
              <VERTEXPOINT X="882.003" Y="678.753"/>
              <RIGHTCONTROLPOINT X="882.003" Y="678.753"/>
            </VERTEX>
          </VERTICES>
        </CONTOUR>
      </CONTOURS>
    </SPLINESHAPE>
    <SUPPRESSOUTPUT>false</SUPPRESSOUTPUT>
    <RUNAROUND TYPE="NONE"/>
  </GEOMETRY>
```

Support for clipping paths

QuarkXPress Server now supports clipping paths. See the CLIPPING element in the DTD.

Support for blends

Modifier XML now supports the specification of blends within a box, allowing blend effects used in QuarkXPress documents (as specified in the color palette) to be preserved and created in Modifier XML based workflows.

Getting started: HTTP

QuarkXPress Server accepts HTTP requests from a browser. You can submit HTTP requests manually in the form of a URL from a browser or dynamically from a client-server solution.

Regardless of the method you use, the server processes requests and returns rendered layouts in the HTTP response. Depending on the original request, QuarkXPress Server preferences, and the type of data being returned (whether that be PDF, JPEG, QXP, XML, or one of the other formats QuarkXPress Server supports), rendered layouts display in the browser or are saved to a server location.

Responses returned to a client-server solution can be manipulated based on the functionality built into the client application. Such a solution may consist of QuarkXPress Server (running on a server computer connected to a network) plus a front-end application (usually Web-based) that provides a graphical user interface (GUI) for end users. The front-end application translates the end users' input into HTTP requests and sends the requests to QuarkXPress Server or QuarkXPress Server Manager. The server processes the requests and returns rendered layouts.

All you need to use QuarkXPress Server is the ability to generate HTTP GET/POST requests. That means you can write front-end applications in just about any language that allows you to make HTTP GET/POST requests.

This Web Integration Guide (WIG) explains the functions available in QuarkXPress Server and how HTTP requests need to be structured.

Related topics:

Dissecting a QuarkXPress Server URL

Interpreting the QXP Server response

Using HTTP GET and POST requests

Function overview

Getting started: Web services

The Web services interface is a collection of request classes. You can easily download and use the corresponding SDK WSDL class definitions from here:
http://<server>:<port>/quark/services/qxpsmsdk?wsdl
Note: Replace <server> above with the IP address of the QuarkXPress Server Manager computer, and <port> with the port number on which to contact QuarkXPress Server Manager. The default port is 8090 for QuarkXPress Server Manager.
These classes can be chained together to form compound QuarkXPress Server requests. The samples distributed as part of this documentation demonstrate how these classes can be used to invoke a QuarkXPress Server command and manipulate the response.
To determine which class provides access to a particular QuarkXPress Server functionality, see the
Function overview
. In addition to the classes listed there, the Web services interface includes the following:

-
- QManagerSDKSvc processes QuarkXPress Server requests. This object's generic processRequest() method takes a *QRequestContext* argument and returns a *QContentData* object containing the QuarkXPress Server response. See the samples distributed or the code snippets in the function documentation
- for details on how this service can be used.
-
- QRequestContext
- is the argument you pass to QManagerSDKSvc. This object contains settings which must be set once per request. All chained requests are set inside the request context.
-
- QRequest
- is the base class for all request objects (such as PDFRenderRequest). Consequently, all request objects share some common data members.
-
- RequestParameters
- is a generic class for executing any request and for adding dynamic properties to a request.
-
- NameValueParam is a generic class for adding dynamic properties to a request. This class is specifically for requests that take *box name/id*
- as the parameter name and its content as the value.
-
- QContentData
- is the response returned when a request is executed. QContentData is a hyperlink that follows the same pattern as the classes above.
-
- QException
- is the exception class for the Manager. It is returned by the getErrorObject method.
-
- QManagerScriptingSvc
- is the Web services scripting interface.

In addition to the core functionality, you can extend the WIG to include your own XTensions software applications by simply modifying an XML file and redeploying the WIG web service.

Note: To exclude empty tags in the request HTML using the WIG, set the value of the appropriate variable to *null*
.

Note:
For Javadocs and WSDL schemas, see the links that display on the QuarkXPress Server Manager Welcome page when you launch QuarkXPress Server Manager. JSP samples are also available from the Welcome page.

Related topics:

Sample applications
Function overview

QRequestContext

Description	Argument passed to QManagerSDKSvc. Contain settings that must be set once per request. All the chained requests are set inside the request context.		
Type	Web Service Data Object		
Members	*Name*	*Type*	*Description*
	documentName	String	File or object name on which the command will b
	serverName	String	Server name. Default is NULL. Load balancer se the host itself in this case.
	serverPort	int	Port at which the desired server is listening.
	userName	String	Server admin username.
	userPassword	String	Server admin password.
	maxRetries	int	Max number of times to try executing the comma returning failure.
	requestTimeout	int	Max time out in milliseconds.
	useCache	boolean	Indicates whether the cache should be checked fo existing result or if the command should be execu
	responseAsURL	boolean	This value indicates whether the server should sen response as-is (text or binary) or store the respon server and return its location as a URL. Because model works on SOAP, which can be slow when large binary files, you might choose to set this valu you suspect that the response is going to be sever megabytes or larger.
	bypassFileInfo	boolean	Indicates whether file info should be fetched befor the command.
	context	String	Context in which the command is being executed.
	request	QRequest	QuarkXPress Server request is instances of reque chained together.

Example, Object Model	sdk.QRequestContext rc = new sdk.QRequestContext();
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.responseAsURL = this.DocumentSettings1.responseAsURL.Checked;
	rc.useCache = this.DocumentSettings1.useCache.Checked;
	rc.bypassFileInfo = this.DocumentSettings1.bypassFileInfo.Checked;
	//Create the service and call it QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);		

QManagerSDKSvc

Description	Web service called to process the QuarkXPress Server request. It has a generic method processRequest() that takes *QRequestContext* as an argument and returns *QContentData* as the QuarkXPress Server response.			
Type	Web Service			
Methods	**processRequest**	Processes the request context and returns the result.		
		Parameter	Type	Description
		requestCmd	QRequestContext	Argument passed to QManagerSDKSvc . Contains settings that must be set once per request. All the chained requests are set inside the request context.
	createSession	Creates a new session and returns a session ID.		
		Parameter	Type	Description
		timeout	long	Timeout for the session in milliseconds. If no call is executed in that time, session is expired and all the open documents in that session are closed without saving. If 0 is passed as value of timeout, default timeout is used. If negative value is passed as timeout, session never expires.
	closeAlldocs	Closes all open documents in the session without saving them. If session does not exist, error is returned. If an error occurs while closing the document, it is logged in the logs. However, the document is marked closed in the internal cache, and no error is returned.		
		Parameter	Type	Description
		sessionId	String	Session whose documents are to be closed.
	closeDoc	Closes the specified document without saving it.		

<table>
<tr><td></td><td></td><td>If session does not exist, error is returned.
If the document is not open, error is returned.
If the document is opened in another session, error is returned.
If an error occurs while closing the document, it is logged in the logs. However, the document is marked closed in the internal cache, and no error is returned.
<table>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>docName</td><td>String</td><td>Document to be closed.</td></tr>
<tr><td>sessionId</td><td>String</td><td>Session in which document was opened.</td></tr>
</table></td></tr>
<tr><td></td><td>closeSession</td><td>Closes the specified session.
If the session does not exist, error is returned.
If any documents are still open in the session, error is returned.
<table>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>sessionId</td><td>String</td><td>Session to be closed.</td></tr>
</table></td></tr>
<tr><td></td><td>getErrorObject</td><td>Gets the internal error object.
If you receive an exception from Web services caused by QuarkXPress Server or Manager (and not a runtime exception such as a null pointer exception), you call this method and pass a stringified form of the exception. The method returns an error object which has easy-to-use methods for getting the error code, getting the error message, etc.</td></tr>
<tr><td></td><td>getOpenDocs</td><td>Gets all the open documents in the session.
If the session does not exist, error is returned.
<table>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>sessionId</td><td>String</td><td>Session whose open documents are sought.</td></tr>
</table></td></tr>
<tr><td></td><td>getOpenSessions</td><td>Gets all open sessions.</td></tr>
<tr><td></td><td>getPreferences</td><td>Gets QuarkXPress Server preferences.</td></tr>
<tr><td></td><td>setPreferences</td><td>Sets QuarkXPress Server preferences.</td></tr>
<tr><td></td><td>getXPressDOM</td><td>Creates a DOM for the specified document.</td></tr>
<tr><td></td><td>newDoc</td><td>Creates a new document for modification and keeps it open until further notice. The document is created with a single layout.
To create a more complex document, use the processRequestEx API.
If a document with the same name is already open, error is returned.
If the session does not exist, error is returned.
<table>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>docName</td><td>String</td><td>Document to be</td></tr>
</table></td></tr>
</table>

				opened for modification. Provide the name only. Relative path can be provided at the time of saving.
		jobJacketName	String	Name of the job jacket to be used. The job jacket is assumed to be already available on the QuarkXPressserver .
		jobTicketName	String	Name of the job ticket to be used.
		host	String	QuarkXPress Server that should be used for this document modification. If null, server is taken from load balancer. The server name provided should be a registered server, currently active, or an error is thrown.
		port	int	QuarkXPress Server port for the server specified in the previous parameter. This is meaningful only if server has been provided in the previous parameter.
		sessionId	String	Session in which document should be opened.
	openDoc	Opens the specified document and keeps it open until further notice. If the document is already open, error is returned. If the session does not exist, error is returned.		
		Parameter	Type	Description
		docName	String	Document (along with relative path if required) to be opened for

				modification.
		host	String	QuarkXPress Server which should be used for this document modification. If null, server is taken from load balancer. The server name provided should be a registered server, currently active, or an error is thrown.
		port	int	QuarkXPress Server port for the server specified in the previous parameter. This is meaningful only if server has been provided in the previous parameter.
		sessionId	String	Session in which document should be opened.
	processRequestEx	Executes the request context. If session id is specified, document is kept open after request is executed. If no session is specified, request is executed normally without keeping the document open. If the document is open in another session, error is returned. If the document is marked dirty, error is returned. A document is marked dirty when the server that opened the document has become inactive. In such a case, the document must be closed and opened again.		
		Parameter	Type	Description
		reqContextObj	QRequestContext	Request to be executed.
		sessionId	String	Session in which the request should be executed. It can be null. If session id is provided, the document is kept open. If no session id is provided, the request is executed normally, as if processRequest was called.

	saveAllDocs	Saves all open documents in the session. The documents are saved one by one. If error occurs while saving the document, error is returned immediately and rest of the documents remain unsaved. If a document is marked dirty, it cannot be saved and an error is returned. A document is marked dirty when the server that opened the document has become inactive. In such a case, the document must be closed and opened again.		
		Parameter	Type	Description
		relativePath	String	Relative path where open documents should be saved. If this is provided, the copies of the opened documents with changes made so far are saved in the new location and the opened documents are still unsaved but have all the changes made so far.
		sessionId	String	Session in which the document exists.
	saveDoc	Saves the open document. If a document is marked dirty, it cannot be saved and an error is returned. A document is marked dirty when the server that opened the document has become inactive. In such a case, the document must be closed and opened again.		
		Parameter	Type	Description
		docName	String	Document to be saved. Must be same as used while opening or creating the document.
		newName	String	New name of the document. Null if it is to be saved as old name.
		relativePath	String	Relative path where the document should be saved. The relative path can also contain the new name of the document. If this is provided, a copy of the open document

				with changes made so far is saved in the new location and the opened document is still unsaved but has all the changes made so far.
		sessionId	String	Session in which the document exists.
	getXPressDOMEx	Lets you create a DOM of a particular layout or portion of a layout.		
	getXMLFromXPressDOM	Creates an XML string out of the DOM.		
	getXPressDOMFromXML	Takes a raw XML representation of a project as a string and returns an object model representing that project, with Project as the root class.		

Example, Object Model	QRequestContext rc = new QRequestContext();
	rc.documentName = "test.qxp";
	rc.responseAsURL = false;
	JPEGRenderRequest jpegRequest = new JPEGRenderRequest();
	rc.request = jpegRequest;
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	QContextData response = svc.processRequest(rc);			

QRequest

Description	Base class for all the request objects, such as PDFRenderRequest. All the request ob some common data members, which are described below.		
Type	Web Service Data Object		
Members	*Name*	*Type*	*Description*
	request	QRequest	QuarkXPress Serve includes instances o objects chained tog

RequestParameters

Description	Generic class for executing any request and also for adding dynamic properties to the request.				
Type	Web Service Data Object				
Members	*Name*	*Type*	*Description*		
	namespace	String	Namespace of the request - e.g., jpeg.		
	params	NameValueParam[]	Parameter array for the specified request - e.g., jp		
Additional Comments	This class can be used to send any request for which a specific class does not exist. When this request exists in the chain, its namespace is concatenated with the namespaces of other requests. So the namespace provided here can be null. The parameters of this class can be used to parameterize the request being sent to the server.				

Example, Object Model	QRequestContext rc = new QRequestContext();
	RequestParameters request = new RequestParameters();
	request.setNamespace("jpeg");
	rc.setRequest = request;
	NameValueParam p1 = new NameValueParam();
	p1.setParamName = "jpegquality";
	p1.setTextValue = "4";
	request.setParams(new NameValueParam[]{p1});				

NameValueParam

Description	Generic class for adding dynamic properties to the request. This class is specifically for the requests that take the box name/id as the parameter name and the box content as the parameter value.		
Type	Web Service Data Object		
Members	*Name*	*Type*	*Description*
	paramName	String	Name of the parameter. In most cases this will be of the box.
	textValue	String	Text value of the box.
	streamValue	byte[]	Stream value of the box. Either text or stream val set.
	contentType	String	The MIME content type of the parameter.

QContentData

Description	Response to a Web Services call to QuarkXPress Server.				
Type	Web Service Data Object				
Members	*Name*	*Type*	*Description*		
	contentType	String	The type of the response. For example, "text/xml" "text/plain."		
	textData	String	If the response type is text, this contains the text. this value is null.		
	responseURL	String	If the "responseAsURL" parameter was set to "tru request, this contains the URL of the response. O this value is null.		
	streamValue	binary	If the response type is binary, this contains the by Otherwise, this value is null.		
	encodingType	String	If the response type is text, this value indicates the of the text (e.g., UTF-8 or ANSI).		
	actualServerPortUsed	String	Identifies the server port.		
	actualServerUsed	String	Identifies the server.		
	headers	String	If the response returned by the server is headers, this array contains the header response.		
	multipartResponse	String	If the response returned by the server is multipart contains the multipart response parts returned by		

Example, Object Model	QRequestContext context = new QRequestContext();
	context.setDocumentName("sample.qxp");
	context.setResponseAsURL(true);
	JPEGRenderRequest request = new JPEGRenderRequest();
	request.setJPEGQuality("4");
	context.setRequest(request);
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	QContentData response = service.processRequest(context);
	System.out.println(response.getResponseURL());				

QException

Description	Exception class for the Manager. This class is returned by the getErrorObject metho		
Type	Exception		
Members	*Name*	*Type*	*Description*
	httpResponseCode	String	HTTP response whi command .
	managerErrorCode	String	Manager error cod exception.
	managerErrorMessage	String	Manager localized message.
	serverErrorCode	String	QuarkXPress serve response from Serv
	serverErrorMessage	String	Response message QuarkXPress Serv
	serverExtendedMessage	String	QuarkXPress Serv extended message.

Example, Object Model	String docName = "notexisting.qxp";
	try {
	QRequestContext ctx = getRequestContext(docName);
	QRequest request = getJPEGRequest();
	ctx.setRequest(ctx);
	QContentData response = getService().processRequest(ctx);
	System.out.println(response.getResponseURL());
	}
	catch (Exception ex) {
	//PLEASE NOTE that the following would work only if manager threw an exce is not a runtime exception. In latter cases, an empty error object will be returned.
	QException error = getService().getErrorObject(ex.toString());
	System.out.println(error.getServerErrorCode());
	}		

QManagerScriptingSvc

Description	Scripting interface via web service.			
Type	Web Service Data Object			
Methods	checkScriptSyntax	Checks the syntax of the script.		
		Parameter	Type	Description
		id	String	Script id.
	deleteScript	Deletes a script.		
		Parameter	Type	Description
		id	String	Script id.
	executeScript	Executes a script.		
		Parameter	Type	Description
		id	String	Scipt id.
	executeScriptFunction	Executes a function of a script.		
		Parameter	Type	Description
		id	String	String id.
		function	String	Function to execute.
	executeScriptFunctionWithArguments	Executes a function of a script, passing arguments to it.		
		Parameter	Type	Description
		id	String	String id.
		function	String	Function to execute.
		arguments	String[]	Arguments to pass to functio
	executeScriptWithVars	Execute a script, declaring variables for the script to use.		
		Parameter	Type	Description
		id	String	Script id.
		variables	QScriptVar[]	Variables to be used by scrip
	getAllScripts	Gets all scripts saved with the system.		
	getErrorObject	Creates error object from error string.		
		Parameter	Type	Description
		errorString	String	Error string to use.
	getScript	Gets script with specified id.		
		Parameter	Type	Description
		id	String	Script id.
	getScriptExecutionDetails	Gets runtime details of a script.		
		Parameter	Type	Description
		scriptId	String	Script id.
	getSupportedLanguages	Gets supported scripting languages.		
	isLanguageSupported	Gets whether a specified scripting language is supported.		
		Parameter	Type	Description

		language	String	Language to check.
	updateScript	Updates a script. If the script does not exist, adds it.		
		Parameter	Type	Description
		script	QScript	Script to update or add.

Dissecting a QuarkXPress Server URL

The general URL format to access QuarkXPress Server and apply parameters to projects or to modify QuarkXPress Server behavior through a Web browser is as follows:
http://Server:Port/Namespace/Directory/Documentname?Parameter=Value
For QuarkXPress Server Manager, use the following URL:
http://Server:Port/quark/servlet/qxpsm/Namespace/Directory/Documentname?Parameter=Value
Note: This Guide provides numerous sample URLs in QuarkXPress Server format. To convert these examples for use with QuarkXPress Server Manager, simply insert /quark/servlet/qxpsm after Port/ .
Note: Earlier versions of QuarkXPress Server Manager work with absolute paths only (for example, Hard Drive:Users:UserName:FolderName:ImageName on Mac OS). With QuarkXPress Server Manager 7.22, you can use absolute paths or you can use relative paths. When you modify a project with SDK objects or SDK classes (such as “SaveAsRequest”) that uses absolute paths, you can use relative paths. The relative paths are relative to QuarkXPress Server, which means the path is relative to the document pool. If you use multiple QuarkXPress Servers, you should be sure to use a common document pool.

Server

The name or IP address of the computer for QuarkXPress Server or QuarkXPress Server Manager.

Port

The port number on which to contact QuarkXPress Server or QuarkXPress Server Manager. The default port is 8080 for QuarkXPress Server and 8090 for QuarkXPress Server Manager.

Namespace

Defines what the URL action will be and any parameters and conditions available to that namespace.

Directory

The path in the document pool where the project is stored.
Note: The directory path is the relative path from the QuarkXPress Server document pool. To access the root level, no directory path is necessary.

Document Name

The name of the QuarkXPress project that you can access from the document pool or the content provider.

Parameter

Further defines the URL action with attributes and values allowed for the namespace or general call.
Parameters are passed in the form attribute=value and are separated by the "&" character.

Related topics:

Getting started: HTTP
Interpreting the QXP Server response
Using HTTP GET and POST requests
Function overview

Interpreting the QuarkXPress Server response

Success scenario

When QuarkXPress Server Manager successfully processes a request through the HTTP interface, the response is the same as QuarkXPress Server's response unless the user has given additional parameters to Manager - e.g, response as url, response redirect, use cache, etc. See the User Guide for all the additional parameters available through the Manager HTTP interface.

Failure scenario

In case of error, QuarkXPress Server Manager retries the request on the same or different QuarkXPress Server depending on the error and global settings done in the admin client of QuarkXPress Server Manager. See the User Guide for the details. If Manager is unable to process the request, it sends back an XML error response in addition to all the header error codes returned by QuarkXPress Server. The XML contains all the details of error that occurred. Such an XML error response might look like this:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <error>
  <httpresponsecode>404</httpresponsecode>
  <xpressservererrorcode>-43</xpressservererrorcode>
  <xpressservererrormessage>File not found.</xpressservererrormessage>
  <xpressserverextendedmessage> <![CDATA[ Error #-43 - File not found. ]]>
</xpressserverextendedmessage>
  <xpressservermanagererrorcode>M8000001</xpressservermanagererrorcode>
  <xpressservermanagererrormessage>The server could not locate the specified
file.</xpressservermanagererrormessage>
  </error>
```

Related topics:

Getting started: HTTP
Dissecting a QuarkXPress Server URL
Using HTTP GET and POST requests
Function overview

Using HTTP GET and POST

requests

This section describes how you can use HTML to interact with QuarkXPress Server. QuarkXPress Server supports both the GET and the POST methods of HTML.

When you use the GET method, the browser encodes form data into a URL. When you use the POST method, form data is passed within a message body. Use the GET method when the form processing is idempotent, and in such cases only. As a simplification, we can say that GET is for getting (retrieving) data whereas POST

can involve storing or updating data, ordering a product, or sending an e-mail.

Working with QuarkXPress Server using an HTTP GET request

To view the HTML, click here

.

Use this HTML to specify a server and its port where you want to send the request. You can specify the name of a project, the output type, and scaling. You can also specify the name of a text box and a picture box, and the paths of the text files and picture files to flow into them. You must specify the path of the text and picture file on the server system. You can also use this HTML to specify the page and layout number of the project.

The form section of the HTML begins with the following line of code:

```
<form id = form1 method="get" enctype="application/x-www-form-urlencoded">
```

For both METHOD="GET" and

METHOD="POST", the processing of a user's submit request (such as, click Submit) in a browser begins with the construction of the form data set, which is then encoded in a manner that depends on the ENCTYPE attribute. That attribute has two possible values: multipart/form-data is for POST submissions only, while application/x-www-form-urlencoded (the default) can be used both for POST and for GET methods.

Next, you must input the server IP and port and the project name. The following lines of code create text fields for them:

```
<TABLE cellSpacing=1 cellPadding=1 border=1 id=TABLE1 >
  <TBODY>
  <TR>
    <TD>
    <INPUT id=ServerTxt name=ServerTxt value="Quark Server"
      readOnly size=13 style="WIDTH: 107px; HEIGHT: 22px">
    </TD>
    <TD>
    <INPUT id=Server maxLength=50 size=16 value=localhost name=Server
      style="WIDTH: 170px; HEIGHT: 22px">
    </TD>
  </TR>
  <TR>
    <TD>
    <INPUT id=PortTxt name=PortTxt value="Port Number"
      readOnly size=13 style="WIDTH: 107px; HEIGHT: 22px">
    </TD>
    <TD>
    <INPUT id=Port maxLength=50 size=17 value=8080 name=Port
      style="WIDTH: 170px; HEIGHT: 22px">
    </TD>
  </TR>
  <TR>
  <TBODY>
</TABLE>
<TR>
  <p></p>
  <TD>
  <INPUT id=DocTxt name=DocTxt value="Document Name"
    readOnly size=13 style="WIDTH: 107px; HEIGHT: 22px">
  </TD>
  <TD>
  <INPUT id=Doc maxLength=50 size=18 name=Doc style="
    WIDTH: 170px; HEIGHT: 22px">
  </TD>
</TR>
```

The
Output Type drop-down menu contains the render formats for the output. The following lines of code create a drop-down menu that contains all the render types supported by QuarkXPress Server.

```
Output Type:
<SELECT id="select1" name="returntype">
  <OPTION value="jpeg">JPEG</OPTION>
  <OPTION value="pdf">PDF</OPTION>
  <OPTION value="qxpdoc">QuarkXPress document</OPTION>
  <OPTION value="eps">EPS Document</OPTION>
  <OPTION value="postscript">POSTSCRIPT</OPTION>
  <OPTION value="png">PNG</OPTION>
</SELECT><td/>
```

Similarly, you can enter the scaling amount on the output using the scale fields.

Note: Scaling supports JPEG, PNG, and EPS render types.

```
Scale:
 <SELECT id="select2" name="scale">
  <OPTION value="1">100%</OPTION>
  <OPTION value="2">200%</OPTION>
  <OPTION value="3">300%</OPTION>
  <OPTION value="5">500%</OPTION>
  <OPTION value=".5">50%</OPTION>
</SELECT><p/>
```

The
input fields for entering the text box name and the file to be flowed into it will display on the HTML page when you use the following lines of code:

```
<TD>
<INPUT id=box1Txt value="Text Box Name"
  readOnly style="WIDTH: 181px; HEIGHT: 22px" size=16>
</TD>
```

```
 <TD>
<INPUT id=box1 maxLength=256 size=43 style="
  WIDTH: 293px; HEIGHT: 22px"></TD>
</TR>
 <TR>
<TR>
<TD>
<INPUT id=box1FileTxt value="File on Server"
  readOnly style="WIDTH: 181px; HEIGHT: 22px" >
</TD>
 <TD>
<INPUT id=box1File maxLength=256 size=43 style="
  WIDTH: 293px; HEIGHT: 22px">
</TD>
</TR>
```

Similarly, the
input fields for entering the picture box name and the file to be flowed into it will display on the HTML page using the following lines of code:

```
<TR>
<TD>
<INPUT id=box2Txt value="Picture Box Name"
  readOnly style="WIDTH: 181px; HEIGHT: 22px" size=16>
</TD>
<TD>
<INPUT id=box2 maxLength=256 size=43 style="
  WIDTH: 293px; HEIGHT: 22px">
</TD>
</TR>
<TR>
<TD>
<INPUT id=box2FileTxt value="File on Server"
  readOnly style="WIDTH: 181px; HEIGHT: 22px">
</TD>
<TD>
<INPUT id=box2File maxLength=256 size=43 style="
  WIDTH: 293px; HEIGHT: 22px">
</TD>
</TR>
```

For creating the
text fields to enter the page and layout number of the project to be rendered, use following lines of code:

```
<TABLE cellSpacing=1 cellPadding=1 width="188" border=1 style="WIDTH: 188px; HEIGHT:
61px">
  <TR>
    <TD>
    <INPUT id=PageTxt value = "Page"
      readOnly style="WIDTH: 50px; HEIGHT: 22px" size=3>
    </TD>
    <TD>
    <input id=Page size="16" maxlength="256"
      style="WIDTH: 147px; HEIGHT: 22px">
    </TD>
  </TR>
  <TR>
    <TD>
    <INPUT id=LayoutTxt value = "Layout"
      readOnly style  ="WIDTH: 50px; HEIGHT: 22px" size=4>
    </TD>
    <TD>
    <input id=Layout size="16" maxlength="256"
      style="WIDTH: 147px; HEIGHT: 22px">
    </TD>
  </TR>
</TABLE>
```

This completes the UI section of the form.

Next you need a button on the form that will process the request and send it to QuarkXPress Server. For this you must create a button on the form. When the button is clicked, it will call the function Submit_onclick().

```
<input type="submit" value="Render document"
    name="Submit" LANGUAGE="javascript"
    onclick="return Submit_onclick()"/>
```

This function is written in the HEAD section of the HTML as:

```
<head>
<TITLE>Quark Stream</TITLE>
<script ID="clientEventHandlersJS" LANGUAGE="javascript">
  function Submit_onclick() {
  var prefix;
  var renderer;
  var file;
  var url;
  var box1Name;
  var box2Name;
  var dataImportStamp = "@dataimport";
  prefix = "http://" + document.getElementById("Server").value + ":";
  port = document.getElementById("Port").value + "/";
  renderer = document.getElementById("select1").value + "/";
  file = document.getElementById("Doc").value;
  box1Name = document.getElementById("box1").value;
  if (box1Name != "")
    {
    document.getElementById("box1File").name = box1Name + dataImportStamp;
    }
  else
    {
    document.getElementById("box1File").name = "";
    }
  box2Name = document.getElementById("box2").value;
  if (box2Name != "")
    {
    document.getElementById("box2File").name = box2Name + dataImportStamp;
    }
  else
    {
```

```
    document.getElementById("box2File").name = "";
    }
  document.getElementById("Page").name = "Page";
  document.getElementById("Layout").name = "Layout";
  url = prefix + port + renderer + file;
  document.getElementById("form1").action = url;
}
</script>
</head>
```

This function reads the values you input in the various fields of the HTML form (server, port, project name, scale, output type, text box name, text file to be flowed, picture box name, picture file to be flowed, page number, and layout number).

Note: When you enter text in the "File on the Sever" text box, prepend "file:" to the path of the text file. For example, suppose the text file is named "data.txt" and it is on the C: drive on the server. Specify its path as: "file:C:\data.txt".

Since we are using the Quark Data Import XTensions software of QuarkXPress Server to flow text and picture data, the suffix @dataimport is appended to the name of the box.

The complete URL is a combination of a server, port, and render type of the file name.

The form data is transmitted as follows:

The action of the form is defined in the HTML by this line of code in the JavaScript function Submit_OnClick():

```
document.getElementById("form1").action = url;
```

The form's method is GET. The user agent gets the value (the URL) of the action, appends a ? to it, then appends the form data set, which it encodes using the application/x-www-form-urlencoded content type. The user agent then traverses the link to this URL. In this scenario, form data is restricted to ASCII codes.

Working with QuarkXPress Server Manager using an HTTP GET request

The methodology is identical to using HTTP GET with QuarkXPress Server (see above), except that with Manager we recommend that you not use GET if you are working with non-ASCII characters. The reason is that the behavior of GET requests in the case of non-ASCII characters is highly dependent on the browser and there is no standard that all browsers follow. In such a case, use POST requests. Of course, you would use Manager URL (/quark/servlet/qxpsm/...) if you are sending a request to Manager.

Working with QuarkXPress Server using an HTTP POST request

To view the HTML, click here

.

Use this HTML form to specify a server and its port where you want to send the request. You can specify the name of a project, the output type, and scaling. You can also specify the name of a text box and a picture box, and you can browse text and picture files on your local system to be flowed into them. The text and picture files that you browse are sent to the server as a part of the post request. In addition, you can specify the page and layout number of the project.

All the code described in the previous section will work with a few changes.

The form section begins with:

```
<form id = form1 method="post" enctype="multipart/form-data">
```

The common code for browsing the text files and the picture files from the client is:

```
<TD><INPUT id=box1FileTxt value="File on Client"
   readOnly style="WIDTH: 180px; HEIGHT: 22px" ></TD>
<TD><input id=box1File type="file"
  size="32" maxlength="256" style="WIDTH: 293px;
  HEIGHT: 22px"></TD></TR>
```

When you click the "Render Document" the following takes place.

The action of the form is defined in the HTML by this line of code in the JavaScript™ function Submit_onClick():

```
document.getElementById("form1").action = url;
```

The form's method is POST. The user agent conducts an HTTP post transaction using the value of the action attribute (the URL), and a message is created according to the content type specified by the enctype attribute.

Working with QuarkXPress Server Manager using an HTTP POST request

The methodology is identical to using an HTTP POST with QuarkXPress Server (see above), except that with Manager, you must use UTF-8 as character encoding in forms. Of course, you would use Manager URL (/quark/servlet/qxpsm/...) if you are sending a request to Manager.

Related topics:

Getting started: HTTP
Dissecting a QuarkXPress Server URL
Interpreting the QXP Server response
Function overview

Integrating and enhancing QuarkXPress Server Manager

This section is for those who want to enhance QuarkXPress Server Manager or integrate it with other software. Please refer to http://localhost:8090/qxpsmdocs/apidocs/index.html for manager API documentation. (Note that the port number used to retrieve the API documentation is 8090 by default, but you should use whatever port number you specified when installing QuarkXPress Server Manager.)

QuarkXPress Server Manager uses the spring framework to instantiate pluggable objects. This also means that the manager has been developed using interface-based programming. When the manager starts up, it reads the contents of ManagerContainerConfig.xml, which is a spring context definition file. All the beans in the file are instantiated. The manager then proceeds to read ManagerConfig.xml and initializes manager by reading various configuration options.

Integrating with Other Web Servers

By default, QuarkXPress Server Manager is integrated with Tomcat®. QuarkXPress Server Manager needs a cache virtual directory to work.

The context definition file contains a bean definition called ContainerAdapter. By default, it uses the Tomcat adapter, QTomcatContainerAdapterImpl. This adapter assumes the virtual directory to be cache and reads the location of the virtual directory from the file cache.xml located in the conf/Catalina/localhost folder of Tomcat.

If QuarkXPress Server Manager needs to be hosted in another web server, the options are to write your own adapter or use QDefaultContainerAdapterImpl provided with QuarkXPress Server Manager. This adapter assumes that the cache folder is located under the web application context folder. The name of the cache folder is also configurable and can be set using the spring configuration file or the setCacheFolderRelativePath method.

Embedding QuarkXPress Server Manager

You can embed QuarkXPress Server Manager in standalone (and other) applications. To do so, you must first initialize QuarkXPress Server Manager, as shown below:

```
QConfigurationData initializationData = new QConfigurationData();
initializationData.setBeanDefinitionConfigFile("ManagerContainerConfig.xml");
QClassFactory.getInstance().init(initializationData);
```

You can configure other QuarkXPress Manager options using QConfigurationManager. Next, you must register one or more QuarkXPress Server hosts, as show below:

```
QConfigManager configManager =
QClassFactory.getInstance().getExecutionEngine().getConfigManager();
 String currentDirectory = System.getProperty("user.dir");
configManager.setCacheFolder(new File(new File(currentDirectory), "cache").getAbsolutePath());
```

```
 configManager.setLogLevel(STANDALONE_CLIENT_LOG_LEVEL);
configManager.setPingType(QPingTypeEnum.PING_SIMPLE);
 QConnectionInfo connInfo = new QConnectionInfo();
connInfo.setServerName(<XPRESS_SERVER_NAME>);
connInfo.setServerPort(<XPRESS_SERVER_PORT>);
connInfo.setUserName(<XPRESS_SERVER_ADMIN_USER>);
connInfo.setPassword(<XPRESS_SERVER_ADMIN_PASSWORD>);
QHostSummary host = new QHostSummary();
host.setConnectionInfo(connInfo);
configManager.registerHost(host);
```

Once you have done so, you can use the embedded QuarkXPress Server Manager as shown below:

```
XMLRequest xmlRequest = new XMLRequest();
QRequestContext context = new QRequestContext();
context.setDocumentName(<SAMPLE_DOCUMENT>);
context.setResponseAsURL(false);
context.setRequest(xmlRequest);
QContentData response = QRequestProcessor.getInstance().processRequest(context);
System.out.println(response.getTextData());
```

Writing special request handlers

If you need to perform custom actions on specific flags, you need to define special flags and write handlers for those. These flags can then be passed as GET parameters to the Servlet, as additional QParam parameters in QCommand (executed using QManagerSvc.executeCommand), or as additional NameValueParam parameters in a derived class of QRequest using QManagerSDKSvc.processRequest. The servlet will automatically create parameters out of these flags and set these in the command before sending it for execution.

To handle these special flags, you can write your request handler derived from the class QRequestHandler. This new handler class can then be inserted anywhere in the chain of responsibility pattern starting with QDocProviderImpl and ending with QHostRequestHandler. Note: try not to change end points. In your handler implementation, handle your special flags, and either return a response after handling or pass the control to the successor for further handling.

Implementing a custom load balancer for QuarkXPress Server Manager

1.
1. Implement the com.quark.manager.lb.QLoadBalancer interface.

1. To use this interface, add a reference to managerengine.jar in your project.

1. This interface method contains the following methods.

getLoadBalancerAlgorithm	
Signature	public String getLoadBalancerAlgorithm();
Description	Returns the name of the algorithm that is mapped to the current load balancer while loading the server.

Returns	The algorithm name used to load balance the list of hosts.
getLoadBalancerDescription	
Signature	public String getLoadBalancerDescription();
Description	Gets the description of the load balancing algorithm, which will be displayed in the admin client.
Returns	Description of the load balancer.
useFileInfo	
Signature	public boolean useFileInfo();
Description	Gets flag telling whether the load balancer uses file information to decide on appropriate host.
Returns	True if fileinfo command should be fired before rendering. False otherwise.
getAvailableHost	
Signature	public QHostProxy getAvailableHost(QHostProxy[] hosts, QCommand command);
Description	Gets available host out of the provided list of hosts to execute the specified command.
Parameters	hosts List of hosts that should be scanned for the most eligible host. command Command for which host is being searched.
Returns	Available host. Can be used for next request.

2.
2. Make a jar for the load balancer.
3.
3. Deploy the jar to the following folder:
3. {Apache-Tomcat Home}\webapps\axis\WEB-INF\lib
4.
4. Configure
4. ManagerContainerConfig.xml for bean mapping.
    1.
    1. Go to the folder
    1. {Apache-Tomcat Home}\webapps\axis\WEB-INF\classes.
    2.
    2. Open the ManagerContainerConfig.xml file and look for the XML tag bean whose id has the value *ConfigurationManager*
    2. .
    3.
    3. Within that tag find the property name *availableLoadBalancers*
    3. .
    4.
    4. In the <list> tag, add the following:
    4. <ref bean={your newbeanID}/>

5.
5. Now define the bean id with *your new bean ID* above this ConfigureManager tag:
5. <bean id={your newbeanID} class={yourLoadBalancerClass}/>
6.
6. Restart the Tomcat server.
7.
7. Log on with the admin client and select the menu option:
7. Global Setting > Load Balancer Method > Choose Load Balancer.
8.
8. Locate your new Load Balancer method and select Save.

Generating custom client SDK class

To generate a custom client SDK class, add new classes and generate new stubs as described below.

Adding new classes

To add new classes:

1.
1. Modify ManagerSDK.xml to reflect changes in Modifier.dtd. ManagerSDK.xml is stored in
1. Server/utilities/ManagerSDK.xml.
2.
2. Using a different folder, create backups of managersdkro.jar, managerdomgenerator.jar, and managerrequestserializer.jar. All of these files are stored in
2. <Installation Folder>/Server/apache-tomcat-5.5.16/webapps/quark/WEB-INF/lib.
3.
3. Execute "Server/utilities/ClientSDKRequestObjectGenerator.sh" (Mac OS) or "Server/utilities/ClientSDKRequestObjectGenerator.bat" (Windows) and look for errors. If you encounter an error, address the problem and execute "ClientSDKRequestObjectGenerator" again. When the process completes successfully, "managersdkro.jar", "managerdomgenerator.jar", and "managerrequestserializer.jar" are regenerated in the
3. webapps/quark/WEB-INF/lib folder. Check the timestamps to verify that the files are new.
4.
4. Launch QuarkXPress Server Manager.
5.
5. Modify "Server/utilities/deploy_sdk.wsdd" to add the bean mapping for the newly generated class. You can add the mapping to position it within the classes corresponding to changes in the DTD or XML, which will make it easier to track changes.
6.
6. Edit "Server/utilities/deploy.sh" (Mac OS) or "Server/utilities/deploy.bat" (Windows) to modify the port number where QuarkXPress Server Manager is running, if it is different from 8090.
7.
7. Execute "Server/utilities/deploy.sh" (Mac OS) or "Server/utilities/deploy.bat" (Windows) and check for errors.
8.
8. Open a Web browser and type the following URL:
8. http://localhost:8090/quark/services/qxpsmsdk?wsdl. Verify that the class you just added is visible in WSDL.

To generate new stubs:

1.
1. Execute "Server/utilities/stub.sh" (Mac OS) or "Server/utilities/stub.bat"(Windows) and look for

errors. If you encounter an error, address the problem and execute "Server/utilities/stub.sh" or "Server/utilities/stub.bat" again. If the process succeeds, "managerwebservicestubs.jar" is generated in the

1. Server/utilities directory. Use these Java stubs in your Java applications that communicate with QuarkXPress Server Manager.
2.
2. If the application you are developing is in Visual Studio .NET, then you need to generate stubs again. Simply open your solution in Visual Studio, and either refresh the Web service reference or remove and add the Web service reference again.

Understanding ManagerSDK.xml

"ManagerSDK.xml" is used to generate client SDK classes for QuarkXPress Server requests. Each element of "ManagerSDK.xml" corresponds to a request handler, a render type, or an element in the DTD. A client SDK class is generated for each element in XML. Each property in the DTD and each parameter of the request handler or render type also corresponds to a unique element in the XML. A class variable is generated for each property. See below for details.

Class: One element for each SDK class generated. The class generated would be derived from QRequest.

Name: Name of the generated class.

namespace: The namespace recognized by QuarkXPress Server when this request class is translated into a QuarkXPress Server request.

Description: Description of the class. Unless it has a null value, description forms the header of the generated class and is included in the generated API docs.

Alias: The alias to be used as an element name if this request class is serialized to XML. For example,when the Project class is serialized to XML, the element used is
Project.

SerializeAs: Option that decides how the class should be serialized.

NameValue means that all members of the class should be handled as name value pairs in the request to QuarkXPress Server. This is the default option in, for example, in JPEGRenderRequest and ModifierStreamRequest.

XML means that the class should be serialized as XML with the class name or alias as the element. All the fields of the class are serialized as child elements. If the field is a subclass of QRequest, it is processed recursively. If the field is an array, it is mandatory that it should be an array of QRequest-derived classes.

Mixed means that the class should be serialized as XML with the class name or alias as the element. All the primitive fields of the class would be serialized as attributes. If the field is a subclass of QRequest, it would be serialized as a child element and it is processed recursively. If the field is an array, it is mandatory that it should be an array of
QRequest-derived classes.

Attribute is valid only if the parent class has provided its serialize option as "XML" or "Mixed." It means that the class should be serialized as XML with the class name or alias as the element. The class fields can only be primitive in this case. All such fields should be serialized as attributes of the element. Further, "value" fields are serialized as values of the element.

Attribute: One element for each class field.

Name: Name of the generated class variable.

Accessor: Unless it has a null value, this is the name of the accessor to get the property. Otherwise, the name generated would be "get" + CamelCase(name). For example, if the name of the property were "quality," the default accessor method would be "getQuality." It can be overridden by using this attribute (e.g., "getJPEGQuality").

mutator: Unless it has a null value, this is the name of the mutator to set the property. Otherwise, the name generated would be "set" + CamelCase(name). For example, if the name of the property were "quality," the default mutator method would be "setQuality." It can be overridden by using this attribute (e.g., "setJPEGQuality").

Description: Description of the property. Unless it has a null value, the description would be included in variable headers and accessor and mutator headers. This information is also included in generated API docs.

Type: Unless it has a null value, this would be the type of the class variable. By default, the variable is of type string. If the type is anything other than primitive data type, that type should be defined as a separate Class element. If the type has a value of "reference," it means the class defined by "name" is a reference that will be used by a "reference" attribute in the same
Class element. Before serialization, the referring values are set in this instance.

Reference: Unless it has a null value, this means that during serialization, the value of the field should be set in the reference class provided.

Note: the reference class should have been declared using "type=reference" as explained above.

Readonly: If the value is "true," it means that this field is for read-only purposes and should be ignore during serialization.

Hidden: If the value is "true," it means that this field should be generated as a private variable, which means it would not be included in WSDL.

Deprecated: If the value is "true", it means that this fields has been deprecated, should not be used, is not supported, and will be removed in a future version of QuarkXPress Server.

cdata: If the value is "true," it means that the value of this field is to be wrapped in the cdata section before sending it to QuarkXPress Server. This is valid only if the field is "value", that is value of the element in modifier XML.

<others>: If any other attributes are defined, a class field with the name as <name>_<others> would be created, and you can write your own implementation for it.

Scripting support

You can write server-side scripts for QuarkXPress Server Manager. These scripts are actually clients that run in the same context as QuarkXPress Server Manager in Tomcat but do not have the overhead of SOAP.

QuarkXPress Server Manager ships with QuarkXPress Server Manager Scripting Environment for editing scripts, but you can use almost any script-editing application. You can run scripts manually by choosing a menu option or toolbar option in QuarkXPress Server Manager Scripting Environment. Also, you can schedule scripts to start and end according to specific time intervals or until conditions are met.

QuarkXPress Server Manager includes a number of sample scripts and libraries for reference. The libraries include ready-to-use functions that perform various tasks in scripts. The samples show you how to use those libraries, and also how to write scripts without using those libraries.

When writing scripts, you can directly access the following functions: print, readUrl, runCommand, spawn, sync, load, debug, info, warn, error, and
exception. Launch QuarkXPress Server Manager Scripting Environment and open the sample scripts to see how these functions can be accessed.

By default, the scripting environment of QuarkXPress Server Manager uses the file system to store the scripts. However, if the need arises, you can write a custom implementation of the storage provider by implementing the QScriptStorage interface and configuring the Spring configuration file, ManagerContainerConfig.xml.

The scripts thus saved can also be executed remotely using Web services. Please see QManagerScriptingSvc for details.

Keep document open (sessions)

In earlier versions of QuarkXPress Server Manager, the software opened a QuarkXPress project, performed a function, and then closed the project. To avoid the delays involved in repeatedly opening and closing a QuarkXPress project, QuarkXPress Server Manager can now keep QuarkXPress projects open until they need to be closed.

To keep projects open for a set period of time, you can now create a session and then open one or more projects in that session. You can specify a timeout interval while creating the session. If the session is not used during the interval, all open projects in that session are closed.

An open project can be modified and saved at any time during the process. An open project can even be saved at another location relative to the QuarkXPress Server document pool. You can also create a new project and keep it open.

For an example of session management, see the dynamicfit scripting sample included with QuarkXPress Server Manager. The dynamicfit script opens a session, opens a QuarkXPress project, and modifies a text box until the text in it fits. To see the dynamicfit script, launch QuarkXPress Server Manager Scripting Environment and open the dynamicfit scripting sample.

Function overview

This section provides an overview of available functions in QuarkXPress Server. For detailed information about a specific function, click on the function name.
Note: QuarkXPress Server uses
case-sensitive XML code.

For QuarkXPress Server Manager users

This section describes the object model by including the collection of request classes that can be chained together to form compound QuarkXPress Server requests. This WIG object model is exposed as a Web service, and the WSDL class definitions can be easily downloaded and used by the client. The samples distributed as part of this WIG demonstrate how these classes can be used to invoke a QuarkXPress Server command and manipulate the response.
The "predefined" classes in this WIG are composed of render types, render modifiers, content modifiers, xml modifiers, xml deconstructors and constructors, and request handlers. The render modifiers are included as properties of the renderer request classes.

Render types

Render types are namespaces you can use to return a QuarkXPress project in a specified file format. Developers can implement additional rendering formats through server XTensions software.

Function	Description	QuarkXPress Server Manager object model classes
eps	Returns an EPS file.	EPSRenderRequest
jpeg	Returns a JPEG image.	JPEGRenderRequest
pdf	Returns a PDF file.	PDFRenderRequest
png	Returns a PNG image.	PNGRenderRequest
postscript	Returns a PostScript file.	PostScriptRenderRequest
ppml	Returns PPML output.	PPMLRenderRequest
qcddoc	Returns a QuarkCopyDesk article.	CopyDeskDocRequest
qxpdoc	Returns a QuarkXPress project file.	QuarkXPressRenderRequest
qxpr	Returns an RLE Raw Custom format image.	RLERawCustomRenderRequest
raw	Returns a project in a QuarkXPress internal format.	RawCustomRenderRequest
screenpdf	Returns a low-resolution PDF file.	ScreenPDFRenderRequest

Render modifiers

Render modifiers let you control which parts of a project are returned and set the scale of returned renderings. For QuarkXPress Server Manager users, render modifiers are included as properties of the render request classes, so they do not have corresponding classes of their own.

Property	Description

box	Renders a single box identified by the supplied name or item ID.
boxes	Renders one or more boxes identified by the supplied names or item IDs.
layer	Renders only the layers identified by the supplied names.
layout	Renders the layout identified by the supplied name, regardless of which layout was active when the project was last saved.
page	Renders only the identified page.
pages	Renders only the identified pages.
scale	Indicates the scale at which the project should be rendered.
spread	Renders only the identified spread.
spreads	Renders only the identified spreads.

Compatibility with different type of objects

Render modifiers work with the following types of objects:

Format	Box	Page	Spread	Doc/Page Range	XSL
JPEG	yes	yes	yes	no	no
PNG	yes	yes	yes	no	no
PostScript(R)	no	yes	yes	yes	no
QuarkXPress	no	no	no	yes for project, no for page	no
Raw	yes	yes	yes	yes	no
EPS	no	yes	yes	yes	no
PDF	no	yes	yes	yes	no
XML	yes	no	no	no	yes

Note: Additional render-type-specific parameters are listed on each render type's page.

Content modifiers

Content modifiers let you alter the content and formatting of boxes in layouts without using the XML modify parameter.

Function	Description	QuarkXPress Server Manager object model classes
fontname	Lets you apply a font to imported text.	RequestParameters
Insert picture	Lets you import a picture into a picture box.	RequestParameters
Insert text	Lets you import text into a text box.	RequestParameters
Picture effects	Lets you apply and delete picture effects with QuarkVista presets.	VistaRequest
XML Import	Lets you import XML content into a QuarkXPress project using placeholders.	XMLImportRequest

XML modify

The modify parameter
lets you modify QuarkXPress projects using XML.

Function	Description	QuarkXPress Server Manager object model classes
Modifying box properties and content	Lets you modify box properties and content.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Box Geometry Layout Runaround
Creating boxes	Lets you create boxes.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Layout
Deleting boxes	Lets you delete boxes.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Layout
Creating tables	Lets you create tables.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Table Layout
Modifying picture properties	Lets you modify picture properties.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Box Layout Picture
Modify text attributes	Let you modify text attributes.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Box Layout RichText Text
Importing data	Lets you import content into text boxes and picture boxes.	ModifierFileRequest ModifierRequest ModifierStreamRequest Project Box

		Content Layout

XML deconstruct and construct

The following namespaces let you construct, deconstruct, and reconstruct entire QuarkXPress projects using XML.

Function	Description	QuarkXPress Server Manager object model classes
xml	Returns an XML representation of a QuarkXPress project.	XMLRequest
construct	Turns an XML representation into a QuarkXPress project.	ConstructRequest ConstructFileRequest ConstructStreamRequest

Administrative request handlers

Request handlers allow you to change the behavior of QuarkXPress server.

Function	Description	QuarkXPress Server Manager object model classes
addfile	Adds the attached QuarkXPress project to the document pool.	AddFileRequest
clang	Specifies the language of the client operating system.	RequestSetting
cplatform	Specifies the client operating system.	RequestSetting
delete	Removes the specified project or folder from the document pool.	DeleteRequest
fileinfo	Retrieves the creation date, modification date, and file size of the specified project in XML format.	FileInfoRequest
flush	Purges a particular project from the open project cache and the memory project cache.	FlushRequest
flushall	Purges all projects from the open project cache and the memory project cache.	FlushAllRequest
getdocinfo	Retrieves information about a specific project in the document pool.	GetDocInfoRequest
getprefs	Retrieves the preference settings of the server in XML format.	GetPreferencesRequest
getprojinfo	Retrieves information about a specific QuarkXPress project in the document pool.	GetProjectInfoRequest
getserverinfo	Retrieves information about QuarkXPress Server.	GetServerInfoRequest
literal	Returns a file with no processing, exactly as it exists on the server.	LiteralRequest

setprefs	Sets one or more preference settings.	ServerPreferences SetPreferencesRequest
saveas	Lets you save a copy of a file in any location available to QuarkXPress Server.	SaveAsRequest
shutdown	Shuts down the server.	ShutdownRequest

Related topics:

Getting started: HTTP
Getting started: Web services
Dissecting a QuarkXPress Server URL
Interpreting the QXP Server response
Using HTTP GET and POST requests

Render types

The process in which a QuarkXPress project is opened in QuarkXPress Server and transformed into another file format (or another QuarkXPress project) is called **Rendering**

. The render type is the format of the project that was rendered and returned to the user or saved to disk. By default, the project render type used by QuarkXPress Server is JPEG.

Note: Bitmap-based render types display in the browser when rendered, such as JPEG and PNG. Non-bitmap-based rendering types do not display in the browser and are downloaded to the user, such as EPS, PostScript, and QuarkXPress project.

Alerts	Cannot open this document type. Please select a QuarkXPress document or template.	HTTP Error #500 This alert is displayed when you try to render a file that is not a QuarkXPress project that exists in the document pool. *What to do*: You can only render QuarkXPress projects.
	The file system document pool is not enabled.	HTTP Error #404 This alert is displayed when the file system document pool is disabled. *What to do*: Choose the menu option **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box and check **Enable File System Document Pool**. Click **OK** and resubmit the render request.
	File not found	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you try to render a project that does not exist. *What to do*: Check the name of the project.
	I/O error trying to read or write to disk.	HTTP Error #500 QuarkXPress Server Error #-36 This alert is displayed when QuarkXPress Server is running on Windows and a shared network folder is selected as the document pool, but the folder is no longer shared. *What to do*: Choose the menu option **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box and set the correct document pool.
	Cannot find required volume or folder.	HTTP Error #404 QuarkXPress Server Error #-35 This alert is displayed when QuarkXPress Server is running on Mac OS and a shared network volume is selected as the document pool, but the volume is no longer shared. *What to do*: Choose the menu option **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box and set the correct document

		pool.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/4/2005 13:49:36 - sample.qxp - Type: image/jpeg - Size: 64002 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the QuarkXPress Server Error Log file contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/4/2005 13:51:32 - Error - Error Code: 10119 - Cannot open this document type. Please select a QuarkXPress project or template.	
Example GET URL	http://localhost:8080/sample.qxp	
Note	There are two ways to specify a render format: • • Enter the render type directly in the browser address field: • http://localhost:8080/pdf/project.qxp • • Choose the menu option **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box, and then choose the default render type from the Default Render Type drop-down menu in the • Server tab.	

eps

Requests EPS rendering of a page or spread in a QuarkXPress project.

Namespace	EPS		
Parameters	outputstyle	stylename	Specifies an output style for EPS output. *stylename* is the name of an output style in the Output Styles dialog box. For example: http://localhost:8080/eps/sample.qxp ?outputstyle=mystylename document is the name of an output style saved in the project's Captured Settings. For example: http://localhost:8080/eps/sample.qxp ?outputstyle=document
	epsformat	color | dcs2	Specifies EPS output format. Default is *color*
	epspreview	tiff | none	Defines a preview of EPS output. Default is *tiff*
	epsdata	ascii | binary | clean8bit	Specifies the data type of the EPS output. Default is *clean8bit*
	epstransparent	1 | 0 | true | false | yes | no	Specifies whether the EPS output is transparent.
Render Modifier Parameters	page	integer	Specifies the single page to be rendered.
	scale	Float .1 to 6.92 for Windows® .1 to 8 on Mac OS®	Specifies a percentage of the size of the page to be returned. Minimum value is .1, meaning 10% of the size. Maximum value is 8 (800% of size) on Mac OS and 6.92 (692%) on Windows.
	spread	integer	Specifies which spread to render. Spread numbers start with 1; spread number 1 references the first page (which is the first spread) in a project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; Layout=1 references the first layout in project. You can also specify the layout name with this parameter.

Response	A QuarkXPress project previewed in EPS format.	
Alerts	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you submit a render request with the *pages* or *box* parameter. *What to do*: Do not use the *pages* or *box* parameter with an EPS render type. It does not support these parameters.
	This Output Style does not exist.	This alert is displayed when you specify a non-existent output style.
	This Output Style cannot be used with this render type.	This alert is displayed when you specify an output style that does not conform to the render type.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 10:03:30 - eps/sample.qxp - Type: application/postscript - Size: 2654464 - Client: 127.0.0.1 If any alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in an error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 11:27:24 - Error - Error Code: 10008 - The renderer for this image type has no way of rendering the desired objects.	
Example GET URL	http://localhost:8080/eps/sample.qxp?epsformat=color&epsdata=clean8bit&epspreview=tiff&epsbleed=0&epstransparent=0	

QuarkXPress Server Manager Object Model	Request Object Name : EPSRenderRequest
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the necessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the EPS renderer request and embed it in the request context.
	EPSRenderRequest epsreq = new EPSRenderRequest();
	epsreq.setEPSData(request.getParameter("EPSData"));
	epsreq.setEPSFormat(request.getParameter("EPSFormat"));
	epsreq.setEPSPreview(request.getParameter("EPSPreview"));
	requestCtx.setRequest(epsreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.	

Notes	• • *To generate an EPS image without using the EPS namespace*	

	• Click the **Server** tab in the **Server Configuration** dialog box. Choose **EPS** from the **Type** drop-down menu in the **Default Render** area. Click **OK**. Now submit the EPS request without using the EPS namespace. Sample URL for this type of request is: • http://localhost:8080/sample.qxp • • You can specify an output style and set additional local parameters of that output style. For example: • http://localhost:8080/eps/sample.qxp?outputstyle=mystylename • where *symmetric* is not specified in the output style. • • You can specify an output style and override any setting in that output style with an additional parameter. For example: • http://localhost:8080/eps/sample.qxp?outputstyle=mystylename • where *asymmetric* is specified in the output style but is overridden with *symmetric* • . • • If you do not specify an output style for EPS output, the *Default EPS Output Style* will be used. In this case, the URL is: • http://localhost:8080/eps/sample.qxp

jpeg

Returns a JPEG file of a QuarkXPress project.

Namespace	JPEG		
Parameters	jpegquality	1 | 2 | 3 | 4	Sets the image quality of a rendered JPEG image. The values are: 1 (highest quality), 2 (high quality), 3 (medium quality), and 4 (lowest quality). The default value is 1.
	upadateimage	true | false	Specifies whether to return modified pictures in the response or not. If set to false, modified pictures are not returned; if set to true or if not included, modified pictures are returned.
Render Modifier Parameters	boxes	string	Returns multiple boxes on a JPEG output.
	page	integer	Specifies the single page to be rendered.
	scale	Float .1 to 6.92 for Windows .1 to 8 on Mac OS)	Determines a percentage of the size of the page to be returned. Minimum value is .1 (meaning 10% of size). Maximum value on Mac OS is 8 (800% of size). Maximum value on Windows is 6.92 (692% of size).
	box	string	Returns a single box.
	spread	integer	Specifies which spread to render. Spread numbers start with 1; spread number 1 refers to the first page (which is the first spread) in a project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	JPEG		
Alerts			
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server		

	error log file. The transaction entry in the error log contain the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 11:27:24 - Error - Error Code: 10008 - The renderer for this image type has no way of rendering the desired objects.
Example GET URL	http://localhost:8080/jpeg/sample.qxp?jpegquality=1

Example, Object Model	**Request Object Name : JPEGRenderRequest**
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the necessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	//STEP2: Create the JPEG renderer request and attach it to the request context.
	JPEGRenderRequest jpreq = new JPEGRenderRequest();
	jpreq.setJPEGQuality(request.getParameter("jpegQuality"));
	jpreq.setLayout(request.getParameter("Layout"));
	requestCtx.setRequest(jpreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.

Notes	*To generate a JPEG image without using the JPEG namespace* Click the **Server** tab in the **Server Configuration** dialog box. Choose **JPEG** from the **Type** drop-down menu in the **Default Render** area. Click **OK** . Enter the JPEG request as http://localhost:8080/sample.qxp

literal

Returns the contents of a file without any attempt to process it as a template. The literal namespace returns any type of project requested. Depending on the file's MIME type, the requested project can be returned within the browser (for example, JPEG) or saved to disk (for example, a Microsoft® Word document).

Namespace	literal	
Parameters		
Response	The requested file returned in the HTTP response.	
Alerts	Incorrect administration realm username and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. *What to do*: Find out the correct username and password that were set in the server configuration and then resubmit *literal* with the correct user name and password.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/10/2005 10:04:52 - literal/Test1.doc - Type: application/vnd.Quark.QuarkXPress - Size: 800768 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm username and password.	
Example GET URL	http://localhost:8080/literal/Story.doc	

Example, Object Model	**Request Object Name : LiteralRequest**
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals("")) rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new LiteralRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	The *literal* request requires an administrator user name and password if those were set in the **Server Configuration** dialog box. When the *literal* request is submitted to the browser, it asks for a user name and password. Enter the user name and password that were set in the **Server Configuration** dialog box and click **OK**.	

png

Returns a PNG file of a QuarkXPress project.

Namespace	PNG		
Parameters	pngcompression	1 | 2 | 3 | 4	Sets the PNG compression for the PNG output. The values are: 1 (lowest compression), 2 (medium compression), 3 (high compression), and 4 (highest compression). The default value is 1.
	upadateimage	true | false	Specifies whether to return modified pictures in the response or not. If set to false, modified pictures are not returned; if set to true or if not included, modified pictures are returned.
Render Modifier Parameters	boxes	string	Returns multiple boxes.
	page	integer	Specifies the single page to be rendered.
	scale	Float .1 to 6.92 for Windows .1 to 8 on Mac OS	Determines a percentage of the size of the page to return. Minimum value is .1 (meaning 10% of size). Maximum value on Mac OS is 8 (800% of size). Maximum value on Windows is 6.92 (692% of size).
	box	string	Returns a single box.
	spread	integer	Specifies which spread to render. Spread numbers start with 1; spread number 1 refers to the first page (which is the first spread) in a project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	PNG		
Alerts			
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:52:59 - png/sample.qxp - Type: image/png - Size: 5454 - Client: 127.0.0.1		

	If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 11:27:24 - Error - Error Code: 10008 - The renderer for this image type has no way of rendering the desired objects.
Example GET URL	http://localhost:8080/png/sample.qxp?pngcompression=1

Example, Object Model	**Request Object Name : PNGRenderRequest**
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the nescessary properties
	sdk.QRequestContext rc = new sdk.QRequestContext();
	boolean responseAsURL = false;
	rc.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the PNG renderer request and embed it in the request context.
	PNGRenderRequest pngreq = new PNGRenderRequest();
	pngreq.setPNGCompression(request.getParameter("PNGCompression"));
	pngreq.setLayout(request.getParameter("Layout"));
	pngreq.setSpread(request.getParameter("Spread"));
	pngreq.setPage(request.getParameter("mPage"));
	rc.setRequest(pngreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(rc);
	Please refer to the samples for further details on the use of the WIG object model.

Notes	*To generate a PNG image without using a PNG namespace* Click the **Server** tab in the **Server Configuration** dialog box. Choose **PNG** from the **Type** drop-down menu in the **Default Render** area. Click **OK** . Enter the PNG request as: http://localhost:8080/sample.qxp

postscript

Generates a PostScript file of a QuarkXPress project.

Namespace	PostScript		
Parameters	prntbleed	Page | asym, clip<boolean>, top<float>, bottom<float>, left<float>, right<float> | sym, clip<boolean>, amount<float>	*prntbleed=asym, clip, top, bottom, left, right* : Specifies asymmetric bleed values for a page. The clip value is of Boolean type (yes/no); the top, bottom, left, and right values are of float type. For example, http://localhost:8080/postscript/Sample.qxp?prntbleed=asym,true,1,2,2,1 results in an asymmetric bleed of 1 on the top, 2 on the bottom, 2 on the left, and 1 on the right. prntbleed=sym,clip,amount: Specifies the amount for a symmetric bleed. The clip value is of Boolean type (yes/no) and the amount value is of float type. For example, http://localhost:8080/postscript/Sample.qxp?prntbleed=sym,true,1 results in a symmetric bleed of 1 on all sides. default: prntbleed=sym,yes,0
	outputstyle	stylename, document	Specifies an output style for PostScript output. *stylename* is the name of an output style in the Output Styles dialog box. For example: http://localhost:8080/postscript/sample.qxp?outputstyle=stylename document is the name of an output style saved in the project's Captured Settings (defined in the QuarkXPress Print dialog box). For example: http://localhost:8080/postscript/sample.qxp?outputstyle=document
Render Modifier Parameters	page	integer	Specifies the single page to be rendered.
	pages	String (page range)	Specifies the multiple pages to be rendered.
	spread	integer	Specifies which spread to render. Spread numbers start with 1; spread

			number 1 refers to the first page (which is the first spread) in a project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	The QuarkXPress project is printed as PostScript.		
Alerts	This page range is invalid.	HTTP Error #500 QuarkXPress Server Error #147 This alert is displayed when you try to render a page range that exceeds the number of pages in the QuarkXPress project. *What to do*: Check the number of pages in the project and enter a correct page range to render.	
	No file produced. The document requested contains only blank pages.	HTTP Error #500 This alert is displayed when you try to render a blank project. *What to do*: You cannot generate a blank PostScript file.	
	PostScript printer mapped to file not found	HTTP Error #500 This alert is displayed when the postscript printer or driver is not set to *Print to File*. *What to do*: Install a postscript printer and set the postscript printer to *Print to File*.	
	This Output Style does not exist	This alert is displayed when you specify a non-existent output style.	
	This Output Style cannot be used with this render type	This alert is displayed when you specify an output style that does not conform to the render type.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/2/2004 20:04:08 - postscript/Sample.qxp - Type: application/postscript - Size: 1143346 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/2/2005 19:58:27 - Error - Error Code: 10121 - No file produced. The document requested contains only blank pages.		
Example GET URL	http://localhost:8080/postscript/Sample.qxp		
Example, Object Model	**Request Object Name : PostScriptRenderRequest** Code Snippet : //STEP1: Create the QuarkXPress Server Request Context and set the nescessary		

	properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	 //STEP 2(SPECIFIC TO REQUESTS):Create the Post Script renderer request and embed it in the request context.
	PostScriptRenderRequest pscreq = new PostScriptRenderRequest();
	pscreq.setPrintBleed(request.getParameter("PrintBleed"));
	pscreq.setPrintPPD(request.getParameter("PrintPPD"));
	pscreq.setPages(request.getParameter("Pages"));
	requestCtx.setRequest(pscreq);
	 //STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.

Notes	• • *To generate postscript without using a postscript namespace* • Click the **Server** tab in the **Server Configuration** dialog box. Choose **PostScript** from the **Type** drop-down menu in the **Default Render** area. Click **OK**. Now submit a postscript request without using a postscript namespace. • Sample URL for this type of request is: • http://localhost:8080/sample.qxp • • Install a PostScript printer on your computer to generate a PostScript or PDF document correctly. • • You can specify an output style and set additional local parameters of that output style. For example: • http://localhost:8080/postscript/sample.qxp?outputstyle=mystylenam&prntbleed=sym,yes,100 • where *symmetric* • is not specified in the output style. • • You can specify an output style and override any setting in that output style with an additional parameter. For example: • http://localhost:8080/postscript/sample.qxp?outputstyle=mystylenam&prntbleed=sym,yes,100 • where *asymmetric* is specified in the output style but is overridden with *symmetric* • . • • If you do not specify an output style for PostScript output, the *Default Print Output Style* will be used. In this case, the URL is: • http://localhost:8080/postscript/sample.qxp • • You can specify that the project's embedded PostScript settings are to be

	used with the parameter *outputstyle=document*. Note: this will map to the *Captured Settings* in the • Print dialog box.

pdf

Returns a PDF file of a QuarkXPress project.

Namespace	PDF		
Parameters	outputstyle	stylename, document	Specifies an output style for PDF output. *stylename* is the name of an output style in the Output Styles dialog box. For example: http://localhost:8080/pdf/sample.qxp?outputstyle=stylename document is the name of an output style saved in the document's Captured Settings as defined in the Export as PDF dialog box. For example: http://localhost:8080/pdf/sample.qxp?outputstyle=document
	title	string	Sets the title of the PDF document.
	subject	string	Sets the subject field of the PDF document.
	author	string	Sets the author of the PDF document.
	keywords	string	Sets the keywords field of the PDF document.
	includehyperlinks	1 | 0 | true | false | yes | no	Specifies whether hyperlinks should be included in the PDF document.
	exportlistsashyperlinks	1 | 0 | true | false | yes | no	Specifies whether lists should be exported as hyperlinks. To use this parameter, set the *includehyperlinks* parameter to true.
	exportindexesashyperlinks	1 | 0 | true | false | yes | no	Specifies whether the index should be exported as hyperlinks. To use this parameter, set the *includehyperlinks* parameter to true.
	exportlistsasbookmarks	1 | 0 | true | false | yes | no	Specifies whether lists should be exported as bookmarks. To use this parameter, set the *includehyperlinks* parameter to *true*.
	mode	composite or separations	Specifies whether the PDF output is a composite or a separation.
	printcolors	cmyk, rgb, grayscale, cmykandspot, asis	Defines the print color of the rendered PDF output. For example, to print the PDF in RGB format, use the value rgb. This option is only used with the composite mode.

	plates	converttoprocess, processandspot, inripseps	Specifies the type of separation to be used in the PDF document. For example, if you choose the value converttoprocess, it breaks the process color CMYK and prints the PDF as a separation. This option is only used with the separation mode.
	produceblankpages	1 | 0 | true | false | yes | no	Specifies whether PDF output should include blank pages. This option is only used with the composite mode.
	useopi	1 | 0 | true | false | yes | no	Specifies whether or not to use OPI for the PDF output.
	images	includeimages, omittiff, omittiffandeps	Specifies whether or not to include tiff or eps images from the OPI server.
	registration	off, centered, offcenter	Specifies the registration for the PDF document.
	offset	0-30 (in points)	Specifies the offset of registration to use on the PDF document.
	bleed	pageitemsonly, symmetric	Specifies the type of bleed to use on the PDF document.
	offsetbleed	0-6 (in inches)	Specifies the offset of bleed to use on the PDF document. This parameter is used when the bleed is symmetric.
	spreads	1 | 0 | true | false | yes | no	Specifies the PDF output display spread.
	lowresolution	1 | 0 | true | false | yes | no	Generates a low resolution PDF document of 36 dpi.
	colorimagedownsample	9-2400	Generates a PDF document with color images that are downsampled to a resolution specified with this parameter.
	grayscaleimagedownsample	9-2400	Generates a PDF document with grayscale images that are downsampled to a resolution specified with this parameter.
	monochromeimagedownsample	9-2400	Generates a PDF document with monochrome images that are downsampled to a resolution specified with this parameter.
	colorcompression	true | false	Specifies whether Manual JPEG Medium compression should be applied to color images.
	grayscalecompression	true | false	Specifies whether Manual JPEG Medium compression should be applied to grayscale images.
	monochromecompression	true | false	Specifies whether ZIP compression should be applied to monochrome images.
	pdffile	string	Saves the PDF file with the name given with the parameter. You can use this

			parameter only when "PDF to Folder" is set in PDF preferences.
	psfile	string	Saves the postscript file with the name given with the parameter. You can use this parameter only when "PostScript for later Distilling" is set in PDF preferences.
	thumbnail	bw | color	Embeds a thumbnail.
	mode	composite | separations	Specifies color mode.
	fontdownload	yes | no	Turns font download on or off. You cannot use this parameter to specify which fonts are downloaded.
	layers	string	Comma-separated list of the layers you want printed.
	transparencyres	Integer value from 36 to 3600	Specifies the transparency flattening resolution.
	verification	pdfx1a | pdfx3	Sets PDF/X 1a or PDF/X 3 verification.
	separate	yes | no	Specifies whether to output the project pages as separate PDF files.
	produceblankplates	yes | no	Specifies whether to output blank QuarkXPress plates in the PDF file.
	download	Boolean 1 | 0 | true | false	Optional parameter. Note: Using this parameter ensures that the correct file suffix is applied to the downloaded document's file name. Case 1: TRUE <http://server:port/pdf/doc.qxp?download=true> or <http://server:port/pdf/doc.qxp?download=1> In this case, a Save As dialog is displayed to the user with filename shown as doc.pdf. The user has the option to save the file or open the file. Even if the browser has the capability (PDF plugin installed) to open the PDF file, still the save as dialog is displayed. This parameter value is not case sensitive. Case 2: FALSE <http://server:port/pdf/doc.qxp?download=false> or <http://server:port/pdf/doc.qxp?download=0> In this case, if the browser has the plugin to display the PDF file, then the file is opened in the browser. No Save As dialog appears. If the browser has no plugin for opening the PDF file, then the Save As dialog appears, with the filename

			shown as doc.qxp. Ideally, this should not be used by the customer, as giving this parameter as FALSE is the same thing as not giving this parameter at all.
Render Modifier Parameters	page	integer	Specifies the single page to be rendered.
	pages	String (page range)	Specifies the multiple pages to be rendered.
	spread	integer	Specifies which spread to render. Spread numbers start with 1; spread number 1 refers to the first page (which is the first spread) in a project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; Layout=1 references the first layout in a project. You can also specify the layout name with this parameter.
	spreads	Boolean 1 | 0 | true | false | yes | no	Generates the preview in spreads.
Response	A QuarkXPress project is returned as a PDF.		
Alerts	This page range is invalid	HTTP Error #500 QuarkXPress Server Error #147 This alert is displayed when try to render a page range that exceeds the number of pages in the QuarkXPress project. *What to do*: Check the number of pages in the project and enter a correct page range to render.	
	No file produced. The project requested contains only blank pages.	HTTP Error #500 This alert is displayed when you try to render a blank project. *What to do*: To generate the PDF of a blank project, select the menu option **QuarkXPress Server > Document Controls > Output Styles**. In the **Output Styles** dialog box, select the **PDF Output Style** and click **Edit**. In the **Edit PDF Style** dialog box, check **Include Blank Pages** and click **OK**. When you submit the PDF request, a blank page is displayed in the PDF document.	
	This Output Style does not exist.	This alert is displayed when you specify a non-existent output style.	
	This Output Style cannot be used with this render type.	This alert is displayed when you specify an output style that does not conform to the render type.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time		

	of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/2/2005 17:17:17 - pdf/sample.qxp - Type: application/pdf - Size: 1927016 - Client: 127.0.0.1. If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/2/2005 18:17:44 - Error - Error Code: 10364 - Invalid Parameter Value.
Example GET URL	This URL renders the "sample.qxp" file in PDF with symmetric bleed applied on output: http://localhost:8080/pdf/sample.qxp?bleed=symmetric&offsetbleed=2 This URL renders a PDF in which the color images are downsampled to a resolution of 300 dpi and Manual JPEG Medium compression is applied to the output: http://localhost:8080/pdf/sample.qxp?colorimagedownsample=300&colorcompression=true

Example, Object Model	**Request Object Name : PDFRenderRequest**
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the nescessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the PDF renderer request and embed it in the request context. the request context.
	PDFRenderRequest pdfreq = new PDFRenderRequest();
	pdfreq.setAuthor(request.getParameter("Author"));
	pdfreq.setTitle(request.getParameter("Title"));
	pdfreq.setLayout(request.getParameter("Layout"));
	pdfreq.setSpread(request.getParameter("Spread"));
	pdfreq.setPage(request.getParameter("mPage"));
	pdfreq.setPages(request.getParameter("Pages"));
	if(strLowResolution !=null && strLowResolution.equals("True"))
	   pdfreq.setLowResolution("true");
	requestCtx.setRequest(pdfreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.

Notes	• • *To generate PDF directly* • Choose the menu option **QuarkXPress Server > Preferences** to display the **Preferences** dialog box. Select the **PDF** option and choose **PDF Direct** from the **Destination** drop-down menu in the **Workflow** area. This generates a PDF directly when you enter a command in the browser: • http://localhost:8080/pdf/Sample.qxp • • *To generate PDF to folder*

- Choose the menu option **QuarkXPress Server > Preferences** to display the **Preferences** dialog box. Select the **PDF** option and choose **PDF to folder** from the **Destination** drop-down menu in the **Workflow** area. Click **Browse**
- and choose the destination folder. It generates a PDF file in the destination folder.
-
- *To generate Postscript for later distilling*
- Choose the menu option **QuarkXPress Server > Preferences** to display the **Preferences** dialog box. Select the **PDF** option and click **PostScript File for Later Distilling** in the **PDF Workflow** area. Click **Browse**
- and select the destination folder. It generates a ".ps" file in the destination folder.
-
- *The role of the PDF Filter XTensions software*
- The PDF Filter XTensions software is used by QuarkXPress Server to produce PDF output. If you disable this XTensions software
- , QuarkXPress Server will not be able to produce PDF documents.
-
- *To generate PDF in any resolution*
- You can use the *colorimagedownsample*, *grayscaleimagedownsample*, *monochromeimagedownsample*, *colorcompression*, *grayscalecompression*, and *monochromecompression*
- parameters to set the PDF resolution.
-
- *To generate PDF without providing a PDF namespace*
- Click the **Server** tab in the **Server Configuration** dialog box. Choose **PDF** from the **Type** drop-down menu in the **Default Render** area. Click **OK**. Submit the PDF request without including a PDF namespace.
- The following is a sample URL for this type of request:
- http://localhost:8080/sample.qxp
-
- You need the Adobe® Acrobat® plug-in to view PDF documents in a browser on Windows.
-
- You can specify an output style and set additional local parameters of that output style. For example:
- http://localhost:8080/pdf/sample.qxp?outputstyle=mystylename&bleed=symmetric
- where *symmetric*
- is not specified in the output style.
-
- You can specify an output style and override any setting in that output style with an additional parameter. For example:
- http://localhost:8080/pdf/sample.qxp?outputstyle=mystylename&bleed=symmetric
- where *symmetric*
- is specified in the output style but is overridden with asymmetric
-
- If you do not specify an output style for PDF output, the *Default Print Output Style* will be used. In this case, the URL is:
- http://localhost:8080/pdf/sample.qxp
-

	• You can specify that the project's embedded PDF settings are to be used with the parameter *outputstyle=document*. • Note: this will map to the *Captured Settings* in the **Export as PDF** • dialog box.

ppml

Requests PPML rendering of a page or spread in a QuarkXPress project.

Namespace	PPML		
Parameters	outputstyle	stylename	Lets you specify an output style for PPML output. *stylename* is the name of an output style in the Output Styles dialog box. For example: http://localhost:8080/ppml/sample.qxp?outputstyle=mystylename _document is the name of an output style saved in the project's Captured Settings. For example: http://localhost:8080/ppml/sample.qxp?outputstyle=document
	path	String	Takes system path as value. This parameter specifies where the output PPML file and the images are to be saved on the machine. For example: path=C:\output
Render Modifier Parameters	thexmldoc	XML	Accepts well-formed XML as input and applies those XML values to the rendered project. The name of the XML elements sent must match the name of the XML Placeholders in the QuarkXPress project.
	paginate	XML	Accepts well-formed XML as input and applies those XML values to the rendered project. The name of the XML elements sent must match the name of the XML Placeholders in the QuarkXPress project. This parameter creates output in a new layout and creates pages as per records in the XML in the new layout.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1; Layout=1 references the first layout in a project. You can also specify the layout name with this parameter.
Response	A QuarkXPress project is output in PPML format.		

Alerts	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you submit a render request with the *pages* or *box* parameter. *What to do*: Do not use the *pages* or *box* parameter with a PPML render type. It does not support these parameters.
	This Output Style does not exist.	This alert is displayed when you specify a non-existent output style.
	This Output Style cannot be used with this render type.	This alert is displayed when you specify an output style that does not conform to the render type.
	The file path is invalid.	HTTP Error #500 This alert is displayed when you specify an invalid path with the *path* parameter.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 10:03:30 - ppml/sample.qxp - Type: application/postscript - Size: 2654464 - Client: 127.0.0.1 If any alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in an error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 11:27:24 - Error - Error Code: 10008 - The renderer for this image type has no way of rendering the desired objects.	
Example GET URL	http://localhost:8080/ppml/sample.qxp?paginate=file:MacintoshHD:file.xml &path=C:\abc&includefont=true	

Example, Object Model	**Request Object Name : PPMLRenderRequest**
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the nescessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the PPML renderer request and embed it in the request context.
	PPMLRenderRequest ppmlreq = new PPMLRenderRequest();
	ppmlreq.setExportPath(request.getParameter("path"));
	ppmlreq.setLayout(request.getParameter("layout"));
	ppmlreq.setOutputStyle(request.getParameter("outputstyle"));
	requestCtx.setRequest(ppmlreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.	

Notes	• • *To generate an PPML image without using the PPML namespace* • Click the **Server** tab in the **Server Configuration** dialog box. Choose **PPML** from the **Type** drop-down menu in the **Default Render** area. Click **OK**. Now submit the PPML request without using the PPML namespace. • Sample URL for this type of request is: • http://localhost:8080/sample.qxp • • You can specify an output style and set additional local parameters of that output style. For example: • http://localhost:8080/ppml/sample.qxp?outputstyle=mystylename&path=C:\abc • where *symmetric* • is not specified in the output style. • • You can specify an output style and override any setting in that output style with an additional parameter. For example: • http://localhost:8080/ppml/sample.qxp?outputstyle=mystylename&path=C:\abc • where *asymmetric* is specified in the output style but is overridden with *symmetric* • . • • If you do not specify an output style for PPML output, the *Default PPML Output Style* will be used. In this case, the URL is • http://localhost:8080/ppml/sample.qxp?path=C:\abc

qxpdoc

Returns a QuarkXPress project.

Namespace	qxpdoc		
Parameters	qxpdocver	7 | 8 | korean6 | japanese6	Specifies the version in which the project is to be rendered. By default, a rendered project is returned in the same version as the installed instance of QuarkXPress Server. This parameter saves a QuarkXPress project to the same or lower version. Rendering of QXP4J/K and QXP6J/K documents is also supported.
	upadateimage	true | false	Specifies whether to return modified pictures in the response or not. If set to false, modified pictures are not returned; if set to true or if not included, modified pictures are returned.
Render Modifier Parameters	layout	String	Specifies the layout name or number to render. Layout numbers start with 1. Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	A QuarkXPress project.		
Alerts	QuarkXPress document return is disabled.	HTTP Error #500 This alert is displayed when you check **Disable QuarkXPress Document Return** in the **Server Configuration** dialog box. *What to do*: Click the **Server** tab in the **Server Configuration** dialog box. Uncheck **Disable QuarkXPress Document Return**. Click **OK** and resubmit the qxpdoc request to the server.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you submit a *qxpdoc* render request with the *page*, *pages*, *box*, or *spread* parameter. *What to do*: Do not use the *page*, *pages*, *box*, or *spread* parameter with the *qxpdoc* render type. The *qxpdoc* render type does not support these parameters.	
	Cannot save a QuarkXPress Project down to an earlier version.	HTTP Error #500 This alert is displayed when you attempt to save a QuarkXPress 6.x project to an earlier version of QuarkXPress with the *qxpdocver* parameter. *What to do*: You cannot save a QuarkXPress 6.x project to an earlier version of QuarkXPress. Render the project in the	

		same or higher version in which it was created.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 12:15:21 - qxpdoc/sample.qxp - Type: application/vnd.Quark.QuarkXPress - Size: 1519616 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 12:05:00 - Error - Error Code: 10123 - QuarkXPress document return is disabled.	
Example GET URL	http://localhost:8080/qxpdoc/sample.qxp	

Example, Object Model	**Request Object Name :**
	QuarkXPressRenderRequest
	Code Snippet :
	//STEP1: Create the QuarkXPress Server Request Context and set the nescessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	boolean responseAsURL = false;
	requestCtx.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the QuarkXpress renderer request and embed it in the request context.
	QuarkXPressRenderRequest qxpreq = new QuarkXPressRenderRequest();
	qxpreq.setDocumentVersion(request.getParameter("XpressDocVersion"));
	qxpreq.setLayout(request.getParameter("Layout"));
	requestCtx.setRequest(qxpreq);
	//STEP3: Create the WIG service and call the processRequest() API
	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.	

Notes	• • A QuarkXPress project cannot be saved to an earlier version. This includes QuarkXPress 7 and QuarkXPress 8 projects. However, a QuarkXPress 8 project can be downsaved to a QuarkXPress 7 project. • • *To generate a QuarkXPress document without using the qxpdoc namespace* • Click the **Server** tab in the **Server Configuration** dialog box. Choose **QuarkXPress Document** from the **Type** drop-down menu in the **Default Render** area. Click **OK**. Now submit the request to generate the QuarkXPress project without using the *qxpdoc* namespace. • The following is a sample URL for this type of request:	

	• http://localhost:8080/sample.qxp

qcddoc

Returns a QuarkCopyDesk article, which is a contiguous text flow of text that can occupy one or more linked boxes in QuarkXPress. A QuarkCopyDesk article can contain multiple components, which are independent text flows. QuarkXPress Server handles QuarkCopyDesk articles as a render type and a document provider. The qcddoc namespace is the render type, which allows a QuarkCopyDesk article to be rendered. To read an article, QuarkXPress Server uses the copydesk namespace as a document provider to render QuarkCopyDesk articles (for example, http://localhost:8080/pdf/copydesk/abc.qcd).

Name space	qcddoc		
Parameters	article		Signifies which article to export from a QuarkXPress project. For example: http://localhost:8080/qcddoc/abc.qxp?article=article1
	component		Can be used to preview/render a particular component within an QuarkCopyDesk article. For example: http://localhost:8080/copydesk/abc.qcd?component=comp1
	format	lightweight| fullfeatured	Exports the QuarkCopyDesk article from a QuarkXPress project in either of the following formats: lightweight or fullfeatured. For example: http://localhost:8080/qcddoc/a

			bc.qxp?article=article1&format=fullfeatured
	saveastemplate	true | false	Outputs the article file (.qcd) as an article template (.qct).
Render Modifier Parameters	modify	XML	Accepts well-formed XML as an input and modifies the project accordingly.
Response	A QuarkCopyDesk article		
Alerts	There is no box with the specified identifier.	The specified box for a component in the article does not exist. What to do: Use a valid box name or ID.	
	The number of characters in the article name can't be greater than max limit.	This alert is displayed when an article name is greater than 32 characters. What to do: Reduce the number of characters in the article name.	
	The article/component name is not unique.	This alert is displayed when there is a conflict with the article names or names of components within the article. What to do: Make sure every article name is unique and every component name within an article is unique.	
Logs	If the article is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, article name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 12:15:21 - qcddoc/sample.qcd - Type: application/vnd.Quark.QuarkCopyDesk - Size: 1519616 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message.		
Example GET URL	http://localhost:8080/qcddoc/copydesk/sample.qcd		
Example, Object Model	**Request Object Name :** CopyDeskDocRequest Please refer to the samples for further details on the use of the WIG object model.		

screenpdf

Overrides the setting in the PDF Workflow area of the **PDF** pane in the Preferences dialog box (*PDF Direct*, *PDF to Folder*, or PostScript File for Later Distilling
) and always returns a low-resolution PDF to the browser.

Namespace	Screenpdf		
Parameters	outputstyle	stylename	Specifies an output style for PDF output. *stylename* is the name of an output style in the Output Styles dialog box. For example: http://localhost:8080/screenpdf/sample.qxp?outputstyle=mystylename _document is the name of an output style saved in the project's Captured Settings. For example: http://localhost:8080/screenpdf/sample.qxp?outputstyle=document
	title	string	Sets the title of the PDF document.
	subject	string	Sets the subject field of the PDF document.
	author	string	Sets the author of the PDF document.
	keywords	string	Sets the keywords field of the PDF document.
	includehyperlinks	1 | 0 | true | false | yes | no	Specifies that hyperlinks should be included in the PDF document.
	exportlistsashyperlinks	1 | 0 | true | false | yes | no	Specifies whether lists should be exported as hyperlinks. To use this parameter, set the *includehyperlinks* parameter to *true*.
	exportindexesashyperlinks	1 | 0 | true | false | yes | no	Specifies whether the index should be exported as hyperlinks. To use this parameter, set the *includehyperlinks* parameter to *true* .
	exportlistsasbookmarks	1 | 0 | true | false | yes | no	Specifies whether lists should be exported as bookmarks. To use this parameter, set the *includehyperlinks* parameter to *true*.
	mode	composite or separations	Specifies whether the PDF output is a composite or a separation.
	printcolors	cmyk, rgb, grayscale, cmykandspot, asis	Specifies the print color of rendered PDF output. For example, to print PDF in RGB format, use *rgb* for the value. This option is only used with

			the composite mode.
	plates	converttoprocess, processandspot, inripseps	Specifies the type of separation to be used in the PDF document. For example, if you set the value to *converttoprocess*, then it breaks the process color CMYK and prints the PDF as a separation. This option is only used with the separation mode.
	produceblankpages	1 | 0 | true | false | yes | no	Specifies whether PDF output can generate blank pages. This option is only used with the composite mode.
	useopi	1 | 0 | true | false | yes | no	Specifies whether to use an OPI format for PDF output.
	images	includeimages, omittiff, omittiffandeps	Specifies whether to include TIFF or EPS images from the OPI server. To use this parameter, set the *useopi* parameter to *true*.
	registration	off, centered, offcenter	Specifies the registration for the PDF document.
	offset	0-30 (in points)	Specifies the offset of registration to use on the PDF document.
	bleed	pageitemsonly, symmetric	Specifies the type of bleed to use on the PDF document.
	offsetbleed	0-6 (in inches)	Specifies the offset of bleed to use on the PDF document. This parameter is used when the bleed is symmetric.
	spreads	1 | 0 | true | false | yes | no	Specifies the PDF output display spread.
	lowresolution	1 | 0 | true | false | yes | no	Generates a low-resolution PDF of 36 dpi.
	colorimagedownsample	9-2400	Generates a PDF with color images that are downsampled to a resolution specified with this parameter.
	grayscaleimagedownsample	9-2400	Generates a PDF with gray scale images that are downsampled to a resolution specified with this parameter.
	monochromeimagedownsample	9-2400	Generates a PDF with monochrome images that are downsampled to a resolution specified with this parameter.
	colorcompression	true | false	Specifies whether to apply Manual JPEG Medium compression to color images.
	grayscalecompression	true | false	Specifies whether to apply Manual JPEG Medium compression to grayscale images.

	monochromecompression	true | false	Specifies whether to apply ZIP compression to monochrome images.
	pdffile	string	Saves the PDF file under the name given with the parameter. You can use this parameter only when *PDF to Folder* is selected in the PDF Workflow area of the **PDF** pane in the Preferences dialog box.
	psfile	string	Saves the postscript file under the name given with the parameter. You can use this parameter only when *PostScript File for Later Distilling* is selected in the PDF Workflow area of the **PDF** pane in the Preferences dialog box.
	thumbnail	bw | color	Embeds a thumbnail.
	mode	composite | separations	Specifies color mode.
	fontdownload	yes | no	Turns font download on or off. You cannot use this parameter to specify which fonts are downloaded.
	layers	string	Comma-separated list of the layers you want printed.
	transparencyres	Integer value from 36 to 3600	Specifies the transparency flattening resolution.
	verification	pdfx1a | pdfx3	Sets PDF/X 1a or PDF/X 3 verification.
	separate	yes | no	Specifies whether to output the project pages as separate PDF files.
	produceblankplates	yes | no	Specifies whether to output blank QuarkXPress plates in the PDF file.
Render Modifier Parameters	page	integer	Specifies the single page to render.
	pages	String (page range)	Specifies the multiple pages to render.
	spread	integer	Specifies which spread to render. Spread numbers start with 1. Spread number 1 refers to the first page (which is the first spread) in the project.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1. Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
	spreads	Boolean (1 | 0 | true | false | yes | no)	Generates the preview in spreads.
Response	A QuarkXPress project is returned as a PDF document.		
Alerts	This page range is	HTTP Error #500	

	invalid.	QuarkXPress Server Error #147 This alert is displayed when you try to render a page range that exceeds the number of pages in the QuarkXPress project. *What to do*: Check the number of pages in the project and enter a valid page range to render.
	No file produced. The document requested contains only blank pages.	HTTP Error #500 This alert is displayed when you try to render a blank project. *What to do*: To generate the PDF of a blank project, select the menu option **QuarkXPress Server > Document Controls > Output Styles**. In the **Output Styles** dialog box, select the **PDF Output Style** and click **Edit**. In the **Edit PDF Style** dialog box, check **Include Blank Pages** and click **OK**. When you submit the PDF request, a blank page is displayed in the PDF document.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/16/2005 15:20:28 - screenpdf/sample.qxp - Type: application/pdf - Size: 2209561 - Client: 127.0.0.1. If an alert is displayed, a transaction message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log file contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/2/2005 18:17:44 - Error - Error Code: 10364 - Invalid Parameter Value.	
Example GET URL	http://localhost:8080/screenpdf/sample.qxp?colorimagedownsample=72&colorcompression=0 This URL generates a low-resolution PDF on the screen. The *colorimagedownsample* and *colorcompression* parameters override any settings in the **PDF** options in the **Preferences** dialog box.	

Example, Object Model	**Request Object Name :**
	ScreenPDFRenderRequest
	Code Snippet:
	//STEP1: Create the QuarkXPress Server Request Context and set the nescessary properties
	sdk.QRequestContext requestCtx = new sdk.QRequestContext();
	String docName = request.getParameter("documentName") ;
	requestCtx.setDocumentName(docName);
	//STEP 2(SPECIFIC TO REQUESTS):Create the QuarkXpress renderer request and embed it in the request context.
	ScreenPDFRenderRequest screenpdfRequest = new ScreenPDFRenderRequest();
	screenpdfRequest.setColorImageDownSample(request.getParameter("ColorImageDownSample"));
	screenpdfRequest.setCompression(request.getParameter("Compression"));
	requestCtx.setRequest(screenpdfRequest);
	//STEP3: Create the WIG service and call the processRequest() API	

	QManagerSDKSvcServiceLocator serviceLocator = new QManagerSDKSvcServiceLocator();
	QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
	sdk.QContentData data = service.processRequest(requestCtx);
	Please refer to the samples for further details on the use of the WIG object model.

Notes	• • *To view the PDF image in a browser on Windows* • The Adobe Acrobat plug-in is required to view PDF documents in a browser on Windows. • • *To generate a PDF in any resolution* • You can use the *colorimagedownsample*, *grayscaleimagedownsample*, *monochromeimagedownsample*, *colorcompression*, *grayscalecompression*, and *monochromecompression* • parameters to set the PDF resolution.

fontname

Specifies the font to be applied to new text flow. When this parameter is used, QuarkXPress Server ignores the original font of the text box and inserts the new text with the font specified by the parameter.

Parameters	[boxid/boxname]:fontname	string	Works with the combination of *boxid* and *boxname*. Replaces the font. This parameter can also be used with *dataimport*: [boxname]fontname@dataimport=<font name>
Response	Preview of project with font applied on new text in text box.		
Alerts	The specified font is not available.	This alert is displayed if you specify a font that is unavailable.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 12/2/2005 16:24:13 - project2.qxp - Type: image/jpeg - Size: 11380 - Client: 127.0.0.1 If an error occurs, the error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 12/2/2005 16:16:26 - Error - Error Code: -43 - File not found.		
Example GET URL	This URL applies Comic Sans MS font to text flown into a box called *story:* http://localhost:8080/png/sample.qxp?Story=This is top story&Story:fontname=Comic Sans MS		

Example, Object Model	Request Object Name: RequestParameters
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the fontname renderer request and embed it in
	RequestParameters request = new RequestParameters();
	NameValueParam nameValue1 = new NameValueParam();
	nameValue1.paramName = this.boxname.Text;
	if(!this.boxvalue1.Text.Equals(""))
	nameValue1.textValue = this.fontname.Text;
	request.params = new NameValueParam[]{nameValue1};
	rc.request = request;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);		

Inserting a picture

Places a new picture in an empty picture box or replaces an existing picture with a new one.

Parameters	<box-name>	string	This parameter replaces the content of the box named <box-name> with a new picture file. The new picture file is specified by the *file:* indicator. The picture file must be present in the document pool.
Response	The QuarkXPress project with a new picture in the picture box.		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you attempt to place a picture into a picture box but the picture does not exist in the document pool. *What to do* : Check whether the picture you attempted to get exists in the document pool. If not, then place the picture in the document pool and then resubmit the request to QuarkXPress Server. If the picture exists in the document pool, check the picture name in the request. QuarkXPress Server also supports relative paths, which can reduce the number of these alerts if paths are specified correctly. This alert is also displayed if the picture is placed in a subfolder of the document pool. In this case, use the *file:* indicator before the picture file path. For example: http://localhost:8080/sample.qxp?Publisher=file:ABC/Publisher.pdf In this example the Publisher.pdf file located in the ABC folder in the document pool is placed into the picture box named *Publisher*.	
	The specified file failed to load in the picture box.	HTTP Error #500 This alert is displayed when you attempt to place an invalid picture file in the picture box. *What to do*: Enter the correct file name and file type in the request.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: *8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1* If an alert is displayed, an error message is written to the QuarkXPress Server		

	error log. The following is a sample of the error log entry: *8/10/2005 10:39:07 - Error - Error Code: 10339 - The specified file failed to load in the picture box.*
Example GET URL	http://localhost:8080/sample.qxp?PictureBox=file:FrenchOpen.pdf

Example, Object Model	Request Object Name: RequestParameters
	sdk.QRequestContext rc = new sdk.QRequestContext();
	        if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the Box Param renderer request and embed it in
	RequestParameters request = new RequestParameters();
	NameValueParam nameValue1 = new NameValueParam();
	nameValue1.paramName = this.boxname1.Text;
	if(!this.boxvalue1.Text.Equals(""))
	  nameValue1.textValue = this.boxvalue1.Text;
	          request.params = new NameValueParam[]{nameValue1};
	rc.request = request;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • Use "&" to change the contents of multiple boxes in one request. The general URL for a multiple boxes request is: • http://localhost:8080/sample.qxp?Logo=file:logo.jpeg&TopPicture=file:TopPicture.pdf • where *Logo* and *TopPicture* • are the names of the two different picture boxes. • • Box names are case sensitive. If you used *logo* instead of *Logo* in the URL above, you would not be able to place the picture into the picture box named *Logo* • . • • You must include the *file:* • indicator before specifying the picture file in the request to place the picture in the picture box. You must ensure that the picture file exists in the document pool. • • *To place a picture in a picture box that exists in a subfolder of the document pool*: • **On Windows**: The general URL format is: • http://localhost:8080/sample.qxp?Pic1=file:Content\Tennis.pdf • where *Pic1* is the picture box name and the picture file "Tennis.pdf" exists in the subfolder "Content" in the document pool. • **On Mac OS**: The general URL format is: • http://localhost:8080/sample?Pic1=file:Content:Tennis.pdf • where *Pic1*

	• is the picture box name and the picture file "Tennis.pdf" exists in the subfolder "Content" in the document pool.

Inserting text

Adds new text to the text box or replaces existing content.

Parameters	<box-name>	string	The *string* parameter replaces the content of the text box named <box-name> with a new string value. Use the *file:* indicator to replace the contents of the text box with text saved in a file.
Response	The QuarkXPress project with the new contents in the text boxes.		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you try to put content from a text file into a text box but the text file does not exist in the document pool. *What to do*: Check whether the text file exists in the document pool. If not, create the text file in the document pool and resubmit the request to QuarkXPress Server. If the text file does exist in the document pool, check the file name in the request. This alert is also displayed if the text file is saved in a subfolder of the document pool. In this case, use the *file:* indicator before the text file path. For example: http://localhost:8080/sample.qxp?Story1=file:Story1\News.doc In this example, the News.doc file located in the Story1 folder in the document pool is placed into the text box named *Story1*.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 8/10/2005 10:32:57 - Error - Error Code: -43 - File not found.		
Example GET URL	http://localhost:8080/sample.qxp?Author=NewText http://localhost:8080/sample.qxp?TopStory=file:TopStory.doc		

Example, Object Model	Request Object Name : RequestParameters
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the Box Param renderer request and embed it in		

	RequestParameters request = new RequestParameters();
	NameValueParam nameValue1 = new NameValueParam();
	nameValue1.paramName = this.boxname1.Text;
	if(!this.boxvalue1.Text.Equals(""))
	 nameValue1.textValue = this.boxvalue1.Text;
	    request.params = new NameValueParam[]{nameValue1};
	rc.request = request;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • Use "&" to change the contents of multiple boxes in one request. The general URL for the multiple-box request is: • http://localhost:8080/sample.qxp?text1=NewText1&text2=NewText2 • where *text1* and *text2* • are the names of the two different boxes. • • Box names are case sensitive. If you had used *Text1* instead of *text1* in the URL mentioned above, you would not have been able to place the content into the text box named *text1* • . • • Include the *file:* • indicator before specifying the text file in the request to place the content in the text box. Ensure that the text file exists in the document pool. • • *To import text from a text file that exists in the subfolder of the document pool*: • **On Windows**: The general URL format is: • http://localhost:8080/sample.qxp?text1=file:Content\Story.txt • where *text1* is the text box name and the text file "Story.txt" exists in the subfolder "Content" in the document pool. • **On Mac OS®**: The general URL format is: • http://localhost:8080/sample?text1=file:Content:Story.txt • where *text1* • is the text box name and the text file "Story.txt" exists in the subfolder "Content" in the document pool. • • You can import an XTags file generated by QuarkXPress.

Picture effects

Applies and removes existing picture effects on images via a Web browser. Using various parameters, picture effects created in QuarkXPress (Vista effects) can be applied to or removed from the rendered images of QuarkXPress projects. QuarkVista effects work only with projects created in QuarkXPress 6.0 and later.

Parameters	applyvistaeffect	string	Lets you apply preset effects to pictures. The picture boxes on which Vista effects are to be applied are specified by the *vistabox* parameter. You can apply Vista effects on multiple picture boxes in one request.
	vistabox	string	Used in conjunction with the *applyvistaeffect* parameter. This parameter takes as its value a comma-separated list of the picture boxes to which the Vista effect is to be applied.
	deletevistaeffect	string	Removes picture effects from pictures. This parameter takes a comma-separated list of box ids or box names as its value.
Response	Preview of QuarkXPress project		
Alerts	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed when an invalid box name or no box name is specified with the *vistabox* parameter. *What to do*: Use a valid box name with the *vistabox* parameter.	
	The picture is not compatible with the Preset.	HTTP Error #500 This alert is displayed when you try to apply Vista effects on a non-compatible image present in a picture box. *What to do*: Either you cannot apply any type of Vista effect on the specified picture box or this preset cannot be applied to the picture box specified in the request. Apply another picture preset.	
	Cannot load picture effects to this box.	HTTP Error #500 This alert is displayed when you try to apply picture effects to a non-picture box or to an empty picture box with the *vistabox* parameter. *What to do*: Provide a valid picture box name with the *vistabox* parameter.	
	This Layout uses Picture Effects that are not supported in	This alert is recorded in the QuarkXPress Server error log file and is not displayed on the status monitor. This error occurs when you try to apply Vista effects on a QuarkXPress 5.0	

	Quarkxpress v5 format. Downsaving the document without picture effects.	document. *What to do*: You cannot apply Vista effects on QuarkXPress 5.0 documents.
	Invalid Preset File!	HTTP Error #500 This alert is displayed if the preset file is present at the Preset location but is an invalid file. *What to do*: Give a valid preset file name in the request.
	There is no Picture Effect with that name.	HTTP Error #500 This alert is displayed when you give a non-existent preset file name in the request with the *applyvistaeffect* parameter. *What to do*: Give a correct preset name with the applyvistaeffect parameter.
	The Vista effects were not applied to one or more images.	This alert is recorded in the QuarkXPress Server error log file and is not displayed on the status monitor. This error occurs when QuarkXPress Server is unable to apply picture effects on some picture boxes specified in the request. *What to do*: Specify only those picture boxes in the request that are compatible with the preset given with the *applyvistaeffect* parameter.
	This document may not display or print correctly because the Vista XTensions software is not present.	This alert is recorded in the QuarkXPress Server error log file and is not displayed on the status monitor. This error occurs when the Vista XTensions software is not present or has been disabled in QuarkXPress Server and a Vista request is submitted to QuarkXPress Server. *What to do*: Either put Vista XTensions software in the QuarkXPress Server XTensions folder or enable Vista XTensions in the XTensions Manager dialog box.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP. The following is a sample of a transaction entry: 11/21/2005 17:09:32 - qxpdoc/project3.qxp - Type: application/vnd.Quark.QuarkXPress - Size: 10273792 - Client: 127.0.0.1 If an error occurs, an error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 11/21/2005 16:40:13 - Error - Error Code: 10400 - The picture is not compatible with the Preset.	
Example GET URL	Apply Vista effect: http://localhost:8080/abc.qxp?applyvistaeffect=color.vpf&vistabox=pb2,box2 where pb2 and box2 are the name of boxes and colors.vpf is preset file saved. Delete Vista effect: http://localhost:8080/abc.qxp?deletevistaeffect=pb2 where pb2 is the name of the picture box from which picture effect has to be	

	removed.

Example, Object Model	Request Object Names : VistaRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	  if(!this.DocumentSettings1.documentName.Text.Equals(""))
	 rc.documentName = this.DocumentSettings1.documentName.Text;
	 //STEP 2(SPECIFIC TO REQUESTS):Create the QuarkVista request and embed it in request context
	VistaRequest vistareq = new VistaRequest();
	vistareq.deleteVistaEffect = this.deletevistaeffect.Text;
	vistareq.vistaBox = this.vistabox.Text;
	vistareq.applyVistaEffect = this.applyvistaeffect.Text;
	rc.request = vistareq;
	 //STEP 3(SPECIFIC TO REQUESTS):Create the JPEG renderer request and embed it in
	JPEGRenderRequest jpreq = new JPEGRenderRequest();
	vistareq.request = jpreq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • The Vista functionality described here only works for master-subrender mode. • • *Which render types support Vista XTensions software?* • All render types of QuarkXPress Server support Vista XTensions software. • • *I have installed the new QuarkXPress Server. Do I need to change something in QuarkXPress Server to work on Vista XTensions software?* • Vista comes as an XTensions software module with QuarkXPress Server. When you install QuarkXPress Server, Vista XTensions software is disabled by default. To use Vista, you must first enable it. Launch QuarkXPress Server. Open the XTensions Manager dialog box (**QuarkXPress Server > Server XTensions Manager**). Check **Vista** from the list to enable it. Click the **OK** button to close the • XTensions Manager dialog box. Now restart QuarkXPress Server. Vista is now enabled in QuarkXPress Server. • • *Can I apply Vista effects to multiple images in a single URL?* • Yes, you can apply the same Vista effects to multiple images in a single URL. • • *What happens if I try to apply and delete a Vista effect in the same URL?* • If you delete and apply vista effects in the same URL parameter, the *deletevistaeffect* • parameter will take precedence. •

	• *What happens if I open a project containing Vista effects but the XTensions software is not loaded?* • An error message is written to the QuarkXPress Server error logs (if error reporting is on): "Error. Vista XTensions module is not present." • • *What is the default value for Picture Effects Preset Location?* • The default selection in the Picture Effects Preset Location area in the Preferences dialog box is • QuarkXPress Server Preferences Folder. • • *How is the Picture Effects Presets folder created in the QuarkXPress Server root folder?* • This folder is created by the QuarkXPress Server installer. • • *Where can I specify or change the location for the folder containing Picture Effects Preset files?* • You can specify the location of the Picture Effects Presets folder in the Preferences dialog box (**QuarkXPress Server > Preferences > Picture Effects** •). Please note that the Picture Effects preference will only be shown if Vista XTensions software is enabled in QuarkXPress Server. • • *Where does QuarkXPress Server look for the preset file specified in the applyvistaeffect parameter?* • The Vista XTensions software adds a new tab called *Picture Effects* to the Preferences dialog box that can be used to specify the location of the Picture Effects Presets folder. The picture effects preset file specified in the *applyvistaeffect* • parameter is searched for in the Picture Effects Preset folder specified in this preference. • • *Which request handlers supply information about picture effects present in a document?* • Deconstructor shows the Vista effects present in a project. The picture effects are specified inside the *PICTUREEFFECTS* tag. If the value inside the *PICTUREEFFECTS* • tag is TRUE, then picture effects have been applied to the picture. Otherwise, the value inside the tag is FALSE.

saveas

Lets you save modified QuarkXPress projects in any supported format to any location on the network and also in the QuarkXPress Server document pool.
When this request is sent to Server Manager, either through HTTP or Web service, the document is saved in all registered QuarkXPress servers if the common doc pool has been switched off in the admin client. If the common doc pool is turned on, the document is saved in any one of the registered QuarkXPress servers.

Parameters	newname	string	Specifies the name under which the project should be saved.
	path	string	Specifies the location for saving the QuarkXPress project other than the document pool.
	savetopool	true | false	Specifies whether the project should be saved to the document pool. Default is *true*.
	replace	true | false	Specifies whether the saved project should replace any existing project with the same name in the specified location. Default is *true*.
Response	QuarkXPress Server responds with message "Document successfully saved."		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you enter the wrong XML file name as a parameter or when you request a project that does not exist in the document pool. *What to do*: Enter the correct name and a path of the XML file and ensure that the project referenced in the request exists in the document pool.	
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you try to render a project that does not exist. *What to do*: Check the name of the document.	
	The file path is invalid.	HTTP Error #500 This alert is displayed when an invalid path is given with the *path* parameter. *What to do*: Specify the correct file path with the *path* parameter.	
	The specified folder is Read-Only.	HTTP Error #500 This alert is displayed when you try to save a project in a read-only folder. *What to do*: Either change the permission of the folder where you want to save the project or change the path value to some	

		other valid path.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP address. The following is a sample of a transaction entry: 11/16/2005 15:41:42 - saveas/5mb.qxp - Type: - Size: 28 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log entry: 11/16/2005 15:42:12 - Error - Error Code: 10371 - The file path is invalid.	
Example GET URL	http://localhost:8080/saveas/pdf/sample.qxp?newname=Customer1&path=HDD:temp&savetopool=true This produces a PDF called Customer1 in the folder HDD:temp and also in the document pool of QuarkXPress Server. It replaces any existing project named Customer1 in the same location.	

Example, Object Model	Request Object Name : SaveAsRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the Save as request and chain it to the document context
	SaveAsRequest saveasreq = new SaveAsRequest();
	saveasreq.newName = this.newname.Text;
	if((this.path.Text != null) && (!this.path.Text.Equals("")))
	saveasreq.newFilePath = this.path.Text;
	saveasreq.replaceFile = this.replace.Checked.ToString();
	saveasreq.saveToPool = this.savetopool.Checked.ToString();
	rc.request = saveasreq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	• • The default value of *savetopool* is *true*, meaning "Save the project in the document pool." However, if you specify a value for the *path* parameter, this default value is overridden and changed to *false*. In this case, if you want the project saved to the document pool, you must explicitly set *savetopool* to *true* • . • • If you have set a watched folder for PDF/PS, then the file is saved in the watched folder instead of the path specified in the URL.	

XML import

Imports data into text boxes and picture boxes through XML and QuarkXPress Placeholders.

Parameters	thexmldoc	XML	Accepts well-formed XML as input and applies those XML values to the rendered project .
	layout	string	Specifies the layout name or number to render. Layout numbers start with 1; layout=1 refers to the first layout in the project . You can also specify the layout name with this parameter.
	paginate	XML	Accepts well-formed XML as input and applies those XML values to the rendered project. Creates pages based on the number of records in the XML.
Response	Preview of the QuarkXPress project with a value in the data import XML tag applied on boxes.		
Alerts	Invalid XML String	HTTP Error #500 This alert is displayed when an invalid XML string is passed as a value to the *thexmldoc* parameter. *What to do*: Enter a correct XML string with the *thexmldoc* parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP address. The following is a sample of a transaction entry: 8/5/2005 18:11:54 - sample.qxp - Type: image/jpeg - Size: 65982 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 8/9/2005 12:38:42 - Error - Error Code: 10396 - Invalid XML String.		
Example GET URL	http://localhost:8080/Sample.qxp?thexmldoc=<?xml version="1.0"?><BookReview><Book><Title>C:\Autumn.jpg</Title><Author> Brian Kernighan and Dennis Ritchie</Author></Book></BookReview> **On MAC OS** : The URL format is: http://localhost:8080/Sample.qxp?thexmldoc=<?xml version="1.0"?><BookReview><Book><Title>/Volumes/MacHD/Pictures/abc.tiff</Title><Author> Brian Kernighan and Dennis Ritchie</Author></Book></BookReview>		

Example, Object Model	Request Object Names : XMLImportRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	   if(!this.DocumentSettings1.documentName.Text.Equals(""))
	 rc.documentName = this.DocumentSettings1.documentName.Text;		

	//STEP 2(SPECIFIC TO REQUESTS):Create the XML Import request
	XMLImportRequest xmlimportreq = new XMLImportRequest();
	xmlimportreq.XMLDocument = this.thexmldoc.Text;
	rc.request = xmlimportreq;
	//STEP 3(SPECIFIC TO REQUESTS):Create the JPEG renderer request
	JPEGRenderRequest jpreq = new JPEGRenderRequest();
	xmlimportreq.request = jpreq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • You can use the absolute path or a relative path to import an image using the *thexmldoc* • parameter. • • Note: The relative path for XML import is relative to the location where the XML file is saved, not relative to the document pool. This support for relative paths in thexmldoc and • paginate parameters is different from the implementation in all other QuarkXPress Server operations.

Related topics:

paginate

paginate

Merges XML content into a template that you have set up in QuarkXPress using XML Import XTensions software.

Namespace	paginate		
Parameters	paginate	XML	Accepts well-formed XML as input and applies those XML values to the rendered project. Creates pages based on the number of records in the XML.
Response	Preview of the merged QuarkXPress project.		
Alerts	Invalid XML String	HTTP Error #500 This alert is displayed when an invalid XML string is passed as a value to the *paginate* parameter. *What to do*: Enter a correct XML string with the *paginate* parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP address. The following is a sample of a transaction entry: 8/5/2005 18:11:54 - sample.qxp - Type: image/jpeg - Size: 65982 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 8/9/2005 12:38:42 - Error - Error Code: 10396 - Invalid XML String.		
Example GET URL	http://localhost:8080/Sample.qxp?paginate=<?xml version="1.0"?><BookReview><Book><Title>C:\Autumn.jpg</Title><Author> Brian Kernighan and Dennis Ritchie</Author></Book></BookReview> **On MAC OS** : The URL format is: http://localhost:8080/Sample.qxp?paginate=<?xml version="1.0"?><BookReview><Book><Title>/Volumes/MacHD/Pictures/abc.tiff</Title><Author> Brian Kernighan and Dennis Ritchie</Author></Book></BookReview> Alternatively, you can specify a path to a file containing the XML: http://localhost:8080/Sample.qxp?paginate=file:MacHD:Sample.xml		
Notes	• • *paginate* • works with the following render types: pdf, postscript, qxp, and ppml. • • If you use *paginate* • with any other render type, the server returns only the first page of the paginated document. • • If you don't provide any XML, the default XML document associated with		

	the layout (applied in QuarkXPress) is used, but you must include the paginate parameter. For example: • http://localhost:8080/pdf/Sample.qxp?paginate • • You can specify an alternate XML file in the URL. This overrides the original XML file associated with the document.

Related topics:

About XML Import

Render modifiers

Use render modifiers to control how a rendered project is previewed.

Parameters	page	integer	Specifies the single page to be rendered.
	pages	String (page range)	Specifies the multiple pages to be rendered.
	box	string	Returns a single box.
	boxes	string	Returns multiple boxes.
	scale	Float .1 to 6.92 for Windows .1 to 8 for Mac	Determines a percentage of the size of the page to be returned. The minimum value for the scale parameter is .1 (meaning 10% of size). The maximum value for the *scale* parameter on Mac OS is 8 (meaning 800% of size). On Windows, the maximum scale value is 6.92 (692%).
	spread	integer	Specifies which spread to render. Spread numbers start with 1. Spread number 1 refers to the first page (which is the first spread) in a project.
	spreads	Boolean 1 | 0 | true | false | yes | no	Generates the preview in spreads.
	layout	String	Specifies the layout name or number to render. Layout numbers start with 1. Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	Preview of the QuarkXPress project with render modifiers applied.		
Alerts			
Logs	If the request is successfully processed, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, project name, paths to any external Composition layouts, type of response produced by server, size of response returned in bytes and client IP address. The following is a sample of a transaction entry: 1/16/2006 14:53:22 - color.qxp - Type: image/jpeg - Size: 8683 - Client: 127.0.0.1		
Example GET URL	http://localhost:8080/pdf/sample.qxp		
Notes	*Are the render modifiers case-sensitive?* No. None of the render modifiers is case-sensitive.		

box

Returns a single box.

Parameters	box	string	Returns a single box.
	overlap	string	Specifies whether to show the area overlapped by the specified box.
Response	Single box		
Alerts	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed when you request a box that does not exist. *What to do*: Check the box name or box ID in the document and enter a correct box request.	
	Cannot render box. The box must be within the page boundaries.	HTTP Error #500 This alert is displayed when you request a box that is placed outside the page boundary. *What to do*: You cannot render a box that is placed outside the page boundary.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you give the *box* parameter with the EPS, PDF, and qxpdoc render type. *What to do*: Do not use the box parameter with the EPS, PDF, and qxpdoc render types. They do not support the box parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 15:04:35 - sample.qxp - Type: image/jpeg - Size: 4366 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2004 15:00:33 - Error - Error Code: 10006 - There is no box with the specified identifier.		
Example GET URL	http://localhost:8080/png/sample.qxp?box=pictbox		
Notes	• • *To render a box placed in other layouts* • The following is a URL to render a box placed in a layout other than layout 1: • http://localhost:8080/png/sample.qxp?layout=2&page=3&box=textbo		

	x • This request renders box name *textbox* • present in page 3 of layout 2. • • When you render using the *box* parameter, box ID gets a higher priority than box name. When you submit a request to render a project with the *box* • parameter using numeric box names, it first checks the internal IDs of boxes in the project and then checks for the names of the boxes. If there is a conflict between box ID and numeric box name, the box with the internal ID is rendered. • • The jpeg, png, and xml namespaces support the *box* • parameter.

boxes

Returns more than one box.

Parameters	boxes	string	Returns more than one box.
	overlap	string	Specifies whether to show the area overlapped by the specified boxes.
Response	The boxes requested		
Alerts	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed when you request a box that does not exist. *What to do*: Check the box name or box ID in the document and enter a correct box request.	
	Cannot render box. The box must be within the page boundaries.	HTTP Error #500 This alert is displayed when you request a box that is placed outside the page boundary. *What to do*: You cannot render a box that is placed outside the page boundary.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you give the *box* parameter with the EPS, PDF, and qxpdoc render type. *What to do*: Do not use the boxes parameter with the EPS, PDF, and qxpdoc render types. They do not support the boxes parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 15:04:35 - sample.qxp - Type: image/jpeg - Size: 4366 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2004 15:00:33 - Error - Error Code: 10006 - There is no box with the specified identifier.		
Example GET URL	http://server:port/jpeg/doc.qxp?boxes=box1,box2		
Notes	• • *To render a box placed in other layouts* • The following is a URL to render a box placed in a layout other than layout 1: • http://localhost:8080/png/sample.qxp?layout=2&page=3&box=textbo		

	x • This request renders box name *textbox* • present in page 3 of layout 2. • • When you render using the *box* parameter, box ID gets a higher priority than box name. When you submit a request to render a document with the *box* • parameter using numeric box names, it first checks the internal IDs of boxes in the document and then checks for the names of the boxes. If there is a conflict between box ID and numeric box name, the box with the internal ID is rendered. • • The jpeg, png, and xml namespaces support the *box* • parameter.

layer

Layers.qrc is a required component with QuarkXPress Server. The layers component enables you to display the contents of a layer (including hidden layers) with various parameters. By using this component, you can add, delete, and modify layers in a QuarkXPress project.

Parameters	layer	string	Specifies the name of a layer to be printed or viewed. You can specify multiple layers in one request.
	addlayer	string	Adds a new layer. You can add only one layer in one request.
	deletelayer	string	Deletes an existing layer from a QuarkXPress project. When you use this parameter, items lying on the layer are also deleted. You can delete only one layer in one request.
	alllayers	1 | 0 | true | false | yes | no	Renders all the layers (including hidden layers) present in a QuarkXPress project. This parameter overrides both the *Visible* and *Suppressoutput* attributes. If true, all layers are shown. If false, document is rendered as such. Any value other than true/false is treated as an invalid parameter value.
	layerattribute	string	Modifies the attributes of an existing layer. You can modify only one layer in one request.
	name	string	Specifies a new name for an existing layer. You must use this parameter in conjunction with the *layerattribute* parameter.
	visible	1 | 0 | true | false | yes | no	Sets or changes the *visible* property of a layer. You can use this parameter in conjunction with the *addlayer* and *layerattribute* parameters. This parameter overrides the *visible* settings done in layer preferences.
	suppressoutput	1 | 0 | true | false | yes | no	Sets or changes the *suppressoutput* property of a layer. You can use this parameter in conjunction with the *addlayer* and *layerattribute* parameters. This parameter overrides the *suppressoutput* settings done in layer preferences.
	locked	1 | 0 | true | false | yes | no	Sets or changes the *locked* property of a layer. You can use this parameter in

			conjunction with the *addlayer* and *layerattribute* parameters. This parameter overrides the *locked* settings done in layer preferences.
	keeprunaround	1 | 0 | true | false | yes | no	Sets or changes the *keeprunaround* property of a layer. You can use this parameter in conjunction with the *addl* ayer and *layerattribute* parameters. This parameter overrides the *keeprunaround* settings done in layer preferences.
Response	Preview of QuarkXPress project		
Alerts	This layer does not exist. Please verify the layer name.	HTTP Error #500 This alert is displayed when you specify an invalid layer name with the *layer*, *layerattribute*, or *deletelayer* parameter. *What to do*: Check the value given with the parameter and enter the correct layer name.	
	Specify a layer name.	HTTP Error #500 This alert is displayed when you do not specify a layer name with the *layer*, *layerattribute*, *addlayer*, or *deletelayer* parameter. *What to do*: Specify a layer name.	
	A layer with the same name already exists.	HTTP Error #500 This alert is displayed when you attempt to add a layer that already exists in a QuarkXPress project. It is also displayed when you attempt to change the name of a layer to a name that already exists in the QuarkXPress project. *What to do*: Specify a layer name that does not exist in the QuarkXPress project.	
	Cannot change the name of the default layer.	HTTP Error #500 This alert is displayed when you attempt to change the name of the default layer. *What to do*: You cannot change the name of default layer. Specify another layer name to modify.	
	Cannot delete the default layer.	HTTP Error #500 This alert is displayed when you attempt to delete the default layer. *What to do*: You cannot delete the default layer. Specify another layer to delete.	
	Invalid parameter value.	HTTP Error #500 This alert is displayed when you do not specify additional attributes in the *addlayer* or *layerattribute* command, or the values of the additional attributes that you specify are invalid. *What to do*: Provide valid values for the attributes.	
	This layer has been locked and cannot be modified.	HTTP Error #500 This alert is displayed when you attempt to add or modify an item lying on a locked layer. *What to do*: Unlock the layer in the project and resubmit the render request.	

Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP. The following is a sample of a transaction entry: 11/17/2005 17:19:48 - qxpdoc/layerlayout.qxp - Type: application/vnd.Quark.QuarkXPress - Size: 84992 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 11/16/2005 19:42:48 - Error - Error Code: 10358 - A layer with the same name already exists.
Example GET URL	This URL renders the layer1 of QuarkXPress project (doc.qxp): http://localhost:8080/doc.qxp?layer=layer1 Addlayer example: http://localhost:8080/qxpdoc/doc.qxp?addlayer=NewLayer&visible=yes&suppresoutput=yes&locked=no Deletelayer example: http://localhost:8080/qxpdoc/doc.qxp?deletelayer=Layer1 Alllayers example: http://localhost:8080/qxpdoc/doc.qxp?alllayers=true Layerattribute example: http://localhost:8080/qxpdoc/doc.qxp?layerattribute=Layer1&name=Layer2&visible=true&keeprunaround=true
Example, Object Model	Creating a new Layer: You can add a new layer to an existing layout of a QuarkXPress project using the following code snippet with the Layer object. Layer layer = new Layer(); layer.name = "New Layer"; layer.operation = "CREATE"; RGBColor rgbcolor = new RGBColor(); layer.RGBColor = rgbcolor; layout.layer = new Layer[]{layer}; • • The rgbcolor parameter can have attributes: red, blue, and green • • Layer is linked to Layout, which is further linked with ModifierRequestContents • • An existing layer can be deleted by providing the ID name for the layer and including the operation attribute as “DELETE”
Notes	• • *Can I add, modify, or delete multiple layers in one request?* • No. You cannot add, modify, or delete multiple layers in a single request. • • *Can layers with the visible and suppressoutput properties set to false be printed?* • No. You cannot print layers whose visible and suppressoutput properties

are set to false.

-
- *How can a hidden layer be printed?*
- A hidden layer will not be displayed in any render type unless that layer is explicitly called with the layer parameter.
-
- *How can a layer with the suppressoutput property set to false be printed?*
- A layer that has been marked in a QuarkXPress project with
- Suppress Output can be printed when it is specifically called with the layer parameter.
-
- *Which render type will print a layer that has been set to suppressoutput?*
- If the layer has been set to
- Suppress Output, it will be shown for the JPEG, PNG, and QXPDOC render types, but will not be output for PDF, PostScript, or EPS.
-
- *Which request handlers give information about the layers present in a project?*
- Use the *deconstruct* and *getdocinfo*
- request handlers to view information about the layers present in a project.
-
- *How can I find and set default preferences for layers?*
- Use the menu option **QuarkXPress Server > Preferences > Default Print Layout > Layers**
- .
-
- *What happens when some parameters are not supplied with the addlayer parameter?*
- By default, the layer will adopt the attributes set in Layer Preferences for those that are not mentioned with the *addlayer*
- command.
-
- *Is there a relationship between the visible and suppressoutput properties?*
- Yes. If the *visible* property is set to false, then the *suppressoutput*
- property is automatically set to true.
-
- *Do boxes lying on a layer get deleted when the deletelayer parameter is used to delete that layer?*
- Yes. When any layer name is used with the *deletelayer*
- parameter, the boxes lying on that layer are also deleted.

layout

Renders the specified layout in the project.

Parameters	layout	string	Specifies the layout name or number to render. Layout numbers start with 1. Layout=1 refers to the first layout in the project. You can also specify the layout name with this parameter.
Response	Preview of project.		
Alerts	The requested layout does not exist.	HTTP Error #500 This alert is displayed when an invalid layout value is given. What to do: Give a correct layout value in the request.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes and client IP address. The following is a sample of a transaction entry: 12/1/2005 10:41:14 - jpeg/sample.qxp - Type: image/jpeg - Size: 63940 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of a transaction entry: 12/1/2005 10:39:45 - Error - Error Code: 10125 - The requested layout does not exist.		
Example GET URL	http://localhost:8080/png/sample.qxp?layout=2 http://localhost:8080/png/sample?layout=Layout 2 where Layout 2 is the name given to layout in the MAC document sample.		
Notes	*Which render types support the layout parameter?* The *layout* parameter is supported by the jpeg, png, pdf, qxpdoc, postscript, eps, and ppml render types.		

page

Returns a single page.

Parameters	page	integer	Specifies the single page to be rendered.
Response	Single page in rendered format		
Alerts	The requested page does not exist.	HTTP Error #500 This alert is displayed when you attempt to render a page that does not exist. *What to do*: Submit the request with a valid page number.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you give a *page* parameter with the *qxpdoc* render type. *What to do*: Do not use a *page* parameter with the *qxpdoc* render type. The *qxpdoc* render type does not support the *page* parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 12:24:13 - png/sample.qxp - Type: image/png - Size: 2645 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in an error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 12:48:15 - Error - Error Code: 10000 - The requested page does not exist.		
Example GET URL	http://localhost:8080/png/sample.qxp?page=2		

Example, Object Model	Creating a new Page: You can add a new page to an existing spread in a QuarkXPress project by using the following code snippet with the Page object:
	Spread spread = new Spread();
	Page page = new Page();
	page.UID = "5";
	page.operation = "CREATE";
	spread.page = new Page[]{page};
	•
	• Page is linked to spread, spread is linked to Layout, which is further linked with ModifierRequestContents
	•
	• An existing page can be deleted by providing the ID for the page and the operation attribute as “DELETE”		

Notes	• • *To render a page lying in another layout*		

	• The following is an URL to render a page lying in a layout other than layout 1: • http://localhost:8080/png/sample.qxp?layout=2&page=3 • This request renders page 3, which is in layout 2 • • The *page* • parameter supports the JPEG, PNG, PDF, PostScript, EPS, and PPML render types.

pages

Returns multiple pages.

Parameters	pages	String (page range)	Specifies the multiple pages to render.
Response	Multiple pages		
Alerts	This page range is invalid	HTTP Error #500 QuarkXPress Server Error #147 This alert is displayed when you try to render a page range that exceeds the number of pages in the QuarkXPress project. *What to do*: Check the number of pages in the project and enter a correct page range to render.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert is displayed when you use the *pages* parameter with the JPEG, EPS, PNG, or qxpdoc render type. *What to do*: Do not use the *pages* parameter with the JPEG, EPS, PNG, or qxpdoc render type. These render types do not support the *pages* parameter.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 14:04:44 - pdf/2000.qxp - Type: application/pdf - Size: 13271 - Client: 127.0.0.1 If an alert is dispayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2005 14:01:44 - Error - Error Code: 147 - This page range is invalid.		
Example GET URL	http://localhost:8080/pdf/sample.qxp?pages=2-4		
Notes	• • *To render pages placed in other layouts* • The following is a URL to render pages placed in a layout other than layout 1: • http://localhost:8080/pdf/sample.qxp?layout=2&pages=2,3 • This request renders page 2 and page 3, which exist in layout 2. • • The pdf and PostScript render types support the *Pages* • parameter.		

scale

Specifies the scale factor for enlarging or reducing rendered images. Used for all built-in formats that generate raster formats.

Parameters	scale	Float .1 to 6.92 for Windows .1 to 8 for Mac	Determines a percentage of the page size to be returned. The minimum value for the *scale* parameter is .1 (meaning 10% of size.). The maximum value for the *scale* parameter on Mac OS is 8 (meaning 800% of size). On Windows, the maximum scale value is 6.92 (692%).
Response	Preview of the project with scale applied.		
Alerts	Invalid scale parameter.	HTTP Error #500 This alert is displayed when an invalid scale value is given. *What to do*: Enter a valid scale value.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The size of the returned document is greater than the size of the original document because the scale applied on the document is returned by the server. The following is a sample of a transaction entry: 8/3/2004 15:19:04 - jpeg/sample.qxp - Type: image/jpeg - Size: 1647112 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/3/2004 15:47:50 - Error - Error Code: 10060 - Invalid scale parameter.		
Example GET URL	http://localhost:8080/png/sample.qxp?scale=2		
Notes	The *scale* parameter is supported by the JPEG, PNG, and EPS render types.		

spread

Specifies which spread to render.

Parameters	spread	integer	Specifies the spread number to render. Spread numbers start with 1. Spread number 1 refers to the first spread in the project.
Response	Preview of a project in a spread format.		
Alerts	The requested spread does not exist	HTTP Error #500 This alert is displayed when you use an invalid spread value. *What to do*: Enter a valid spread value.	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 15:19:04 - sample.qxp - Type: image/jpeg - Size: 1647112 - Client: 127.0.0.1 If an alert is displayed, a transaction error message is written to the QuarkXPress Server Error Log file. The transaction entry in the error log file contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 8/5/2005 9:43:02 - Error - Error Code: 10072 - The requested spread does not exist.		
Example GET URL	http://localhost:8080/png/sample.qxp?spread=2		
Example, Object Model	Creating a new Spread: You can add a new spread to an existing layout of a QuarkXPress project using the following code snippet with the Page object Spread spread = new Spread(); spread.UID = "5"; spread.operation = "CREATE"; layout.spread = new Spread[]{spread}; • • Spread is linked to Layout, which is further linked with ModifierRequestContents • • An existing spread can be deleted by providing the ID for the spread and the operation attribute as “DELETE”		
Notes	The JPEG, PNG, PDF, PostScript, and EPS render types support the *spread* parameter.		

spreads

Specifies which spreads to render.

Parameters	spreads	Boolean 1 | 0 | true | false | yes | no	Generates the preview in spreads.
Response	Preview of the project in spread format.		
Alerts			
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 1/16/2006 11:14:03 - pdf/project23.qxp - Type: application/pdf - Size: 1084 - Client: 127.0.0.1		
Example GET URL	http://localhost:8080/pdf/sample.qxp?spreads=true		
Notes	*Which render types support the spreads parameter?* The *spreads* parameter is only supported by the PDF render type.		

About XML modify

Modifies a QuarkXPress project using XML. The modify parameter has an associated DTD. The call structure required by the Modifier DTD
is very similar to the call structure required in previous versions of QuarkXPress Server. Only minor modifications are necessary to upgrade solutions written for previous versions to work with this version. Note that one major change to this DTD is that it's important not to specify units when providing measurement values.

Note: This topic covers the modify parameter when it is used without the construct namespace. You can also use the modify parameter to specify an XML file to use when constructing a project; for more information, see About XML deconstruct and construct

.

DTD	Modifier DTD		
Parameters	modify	string	Specifies the XML file or string that describes how to create the project. The XML file is specified by the *file:* indicator and supports an absolute path or a relative path to the document pool. Note: The XML file must adhere to the Modifier DTD and be present in the specified location.
Example GET URL	http://QXPServer8:8080/project1.qxp?modify=file:sample.xml		

Example XML	This xml deletes page 2 of a QuarkXPress layout:
	<PROJECT>
	 <LAYOUT>
	  <ID NAME="Layout 1" />
	  <SPREAD>
	   <ID UID="1" />
	   <PAGE OPERATION="DELETE">
	    <ID UID="2" />
	   </PAGE>
	  </SPREAD>
	 </LAYOUT>
	</PROJECT>		

Response	The updated QuarkXPress project		
Alerts			
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: *8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1*		

	If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of the error log entry: *8/10/2005 10:39:07 - Error - Error Code: 10339 - The specified file failed to load in the picture box.*
Notes	The xml namespace takes two arguments: The name of the project to be modified and a modify parameter that points to the XML file or string that describes how to create the project: http://QXPServer8:8080/project1.qxp?modify=file:path to XML file on server http://QXPServer8:8080/project1.qxp?modify=XML string You can also modify QuarkCopyDesk articles. To modify a QuarkCopyDesk article: http://localhost:8080/copydesk/abc.qcd?modify=file:XMLfile.xml

Related topics:

Modifying box properties and content
Creating boxes
Deleting boxes
Modifying picture properties
Modifying text attributes

Modifying box properties and content

To modify box properties and content, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID
-
- TEXT
-
- PICTURE
-
- GEOMETRY
-
- CONTENT
-
- SHADOW
-
- FRAME
-
- PLACEHOLDER
-
- METADATA

Example

The following XML shows how some of these parameters work.

```
<?xml version="1.0" encoding="UTF-8"?>
<PROJECT>
  <LAYOUT>
    <ID NAME="Layout 1"/>
    <SPREAD>
      <ID UID="1"/>
 <BOX BOXTYPE="CT_TEXT">
        <ID NAME="SERVICES"/>
        <GEOMETRY>
          <MOVEUP>50</MOVEUP>
          <MOVELEFT>30</MOVELEFT>
          <ALLOWBOXONTOPASTEBOARD>true</ALLOWBOXONTOPASTEBOARD>
        </GEOMETRY>
        <CONTENT
CONVERTQUOTES="true">HD:QuarkXPress:DocPool:Services.txt</CONTENT>
```

```
   </BOX>
   <BOX BOXTYPE="CT_TEXT">
    <ID NAME="FAMILY"/>
    <GEOMETRY>
     <MOVERIGHT>20</MOVERIGHT>
     <MOVEDOWN>30</MOVEDOWN>
     <ALLOWBOXONTOPASTEBOARD>true</ALLOWBOXONTOPASTEBOARD>
     <ALLOWBOXOFFPAGE>true</ALLOWBOXOFFPAGE>
    </GEOMETRY>
   </BOX>
   <BOX BOXTYPE="CT_TEXT">
    <ID NAME="PRODUCTS"/>
    <GEOMETRY>
     <GROWACROSS>44</GROWACROSS>
     <GROWDOWN>30</GROWDOWN>
     <ALLOWBOXONTOPASTEBOARD>false</ALLOWBOXONTOPASTEBOARD>
    </GEOMETRY>
   </BOX>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="MAP"/>
    <GEOMETRY>
     <SHRINKACROSS>30</SHRINKACROSS>
     <SHRINKDOWN>30</SHRINKDOWN>
    </GEOMETRY>
   </BOX>
   <BOX COLOR="Blue" BOXTYPE="CT_PICT">
    <ID NAME="CONTACT"/>
    <GEOMETRY>
     <STACKINGORDER>BRINGTOFRONT</STACKINGORDER>
     <RUNAROUND TYPE="ITEM" TOP="4" RIGHT="4" LEFT="4" BOTTOM="4"/>
     <ALLOWBOXOFFPAGE>false</ALLOWBOXOFFPAGE>
    </GEOMETRY>
   </BOX>
  </SPREAD>
 </LAYOUT>
</PROJECT>
```

Response	Preview of the QuarkXPress project with a new box created in the specified position.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect XML file as a parameter or when you request a project that does not exist in the document pool. *What to do*: Enter the correct name and path of the XML file and ensure that the project given in the request exists in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request.

		What to do: Enter the correct XML file name and resubmit the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	The XML document contains an invalid tag value.	HTTP Error #500 An alert similar to this one is displayed when you enter an incorrect value for a tag. For example, you enter a string value for a tag that accepts a numeric value. *What to do*: Enter a valid value in all tags of XML and resubmit the request.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry:	
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\updateBox.xml where the updateBox.xml file exists in the *C:* drive of the server. **On Mac OS**: The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:updateBox.xml where the updateBox.xml file exists in the xml folder of MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where the *xml-string* tag consists of valid XML commands of image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/> <BOX BOXTYPE="CT_PICT" COLOR="Blue" SHADE="50" OPACITY="50"> <ID NAME="MOUNTAINS"/><CONTENT> file:Services.eps</CONTENT> </BOX></SPREAD></LAYOUT></PROJECT>	
Example 1, Object Model	Request Object Names : • • ModifierRequest • • ModifierRequestContents • • Layout • • ID •	

	• Box
	•
	• Geometry
	•
	• Runaround
	•
	• ModifierFileRequest
	• : member contents is used to set the file path or send the XML itself.
	sdk.QRequestContext rc = new sdk.QRequestContext();
	    if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	 //STEP 2(SPECIFIC TO REQUESTS):Create the BOX modifier renderer request and embed it in request context
	ModifierRequest request = new ModifierRequest();
	Project contents = new Project();
	Geometry geo = new Geometry();
	geo.moveUp = this.moveup.Text;
	geo.color = this.color.Text;
	geo.growDown = this.growdown.Text;
	geo.shrinkAcross = this.shrinkacross.Text;
	Box box = new Box();
	box.UID = this.Boxid.Text;
	box.geometry = geo;
	Layout layout1 = new Layout();
	layout1.name = this.layout.Text;
	layout1.boxes = new Box[]{box};
	if(this.runaround.Checked == true)
	{
	  Runaround runaround = new Runaround();
	  runaround.type = this.runaroundtype.Text;
	  runaround.top = this.top.Text;
	  runaround.left = this.left.Text;
	  runaround.right = this.right.Text;
	  geo.runaround = runaround;
	}
	contents.layouts = new Layout[]{layout1};
	request.contents = contents;
	rc.request = request;
	 //Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Example 2, Object Model	New box modifier attributes: To edit the geometrical properties of an existing box in a QuarkXPress project, the new object linking is shown below. ModifierRequest < Project < Layout < Spread < Box < Geometry The Geometry object contains the properties: allowBoxOffPage, allowBoxOnToPasteBoard, angle , growAcross, growDown, layer, linestyle (of type 'Linestyle'), moveDown, moveLeft, moveRight, moveUp, page, position (of type 'Position'), runaround (of type 'Runaround'), shape, shrinkAcross, shrinkDown, stackingOrder, and suppressOutput. Runaround object contains the properties: bottom, edited, invert, left, noise, outset,

	outsideOnly, pathName, restrictToBox, right, smoothness, threshold, top, type

Related topics:

About XML modify
Creating boxes
Deleting boxes
Modifying picture properties
Modifying text attributes

Creating boxes

To create a new box, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID
-
- TEXT
-
- PICTURE
-
- GEOMETRY
-
- CONTENT
-
- SHADOW
-
- FRAME

Example

The following XML shows how some of these parameters work.

```
<PROJECT>
 <LAYOUT>
  <ID UID="layout 1"/>
  <SPREAD>
  <ID UID="1"/>
    <ID/>
    <BOX OPERATION="CREATE" BOXTYPE="CT_PICT">
     <ID NAME="PRODUCTS"/>
     <GEOMETRY PAGE="2" SHAPE="SH_RECT">
       <POSITION>
        <TOP>5</TOP>
        <LEFT>5</LEFT>
        <BOTTOM>10</BOTTOM>
        <RIGHT>10</RIGHT>
       </POSITION>
     </GEOMETRY>
    </BOX>
  </SPREAD>
 </LAYOUT>
</PROJECT>
```

Response	Preview of the QuarkXPress project with new box crated in specified position.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect XML file as

		a parameter or when you request a project that does not exist in the document pool. *What to do*: Enter the correct name and path of the XML file and ensure that the project given in the request exists in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request. *What to do*: Enter the correct XML file name and resubmit the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert is displayed when you enter an incorrect value for a tag. For example, you enter a string value for a tag that accepts a numeric value. *What to do*: Enter a valid value in all tags of XML and resubmit the request.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 4/12/2007 14:51:50 - Error - Error Code: 10207 - The XML document is not valid or well formed. Project: /table.qxp	
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\createBox.xml where the createBox.xml file exists in the *C:* drive of the server. **On Mac OS** : The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:createBox.xml where the createBox.xml file exists in the xml folder of MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where the *xml-string* tag consists of valid XML commands of image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/> <BOX OPERATION="CREATE"><ID NAME="MISSION"/> </BOX></SPREAD></LAYOUT></PROJECT>	

Example, Object Model	Creating a new Box: You can add a new box of a specified type to an existing spread in a QuarkXPress project at the specified position using the following code snippet with the Box object.
	Spread spread = new Spread();
	Box box = new Box();
	box.name = "textbox1";
	Geometry geometry = new Geometry();
	Position position = new Position();
	position.top = "110";
	position.left = "89";
	position.bottom = "220";
	position.right = "300";
	geometry.position = position;
	geometry.shape = "SH_RECT";
	geometry.page = "1";
	geometry.layer = "Default";
	box.geometry = geometry;
	box.boxType = "CT_TEXT";
	box.operation = "CREATE";
	spread.box = new Box[]{box};
	•
	• While creating a box, position attributes need to be specified.
	•
	• Box is linked to Spread, Spread to Layout, which is further linked with ModifierRequestContents
	•
	• An existing box can be deleted by providing the ID name for a box and the operation attribute as “DELETE”

Related topics:

About XML modify
Modifying box properties and content
Deleting boxes
Modifying picture properties
Modifying text attributes

Deleting boxes

To delete a box, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID

Example

The following XML shows how these parameters work.

```
<PROJECT>
 <LAYOUT>
  <ID UID="Layout 1"/>
  <SPREAD>
   <ID UID="1"/>
   <BOX OPERATION="DELETE">
    <ID NAME="SERVICES"/>
   </BOX>
  </SPREAD>
 </LAYOUT>
</PROJECT>
```

Response	Preview of the QuarkXPress project with the box deleted.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect XML file as a parameter or when you request a project that does not exist in the document pool. *What to do*: Enter the correct name and path of the XML file and ensure that the project given in the request exists in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request. *What to do*: Enter the correct XML file name and resubmit the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert is displayed when you enter a wrong value for a tag. For example, you enter a string value for a tag that accepts a numeric value.

		What to do: Enter a valid value in all tags of XML and resubmit the request.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry:	
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\deleteBox.xml where the deleteBox.xml file exists in the *C:* drive of the server. **On MAC** : The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:deleteBox.xml where the deleteBox.xml file exists in the xml folder of MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where the *xml-string* tag consists of valid XML commands of image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/> <BOX OPERATION="DELETE"><ID NAME="HISTORY"/> </BOX></SPREAD></LAYOUT></PROJECT>	
Notes	You can use the xml namespace or Telegraph XTensions software to determine the ID or name of the box you want to delete.	

Related topics:

About XML modify
Modifying box properties and content
Creating boxes
Modifying picture properties
Modifying text attributes

Creating tables

To create a new table, use the following parameters in the Modifier DTD:

-
- SPREAD
-
- TABLE
-
- COLSPEC
-
- COLUMN
-
- ROW
-
- CELL

Example

The following XML shows how some of these parameters work.

```
<PROJECT>
  <LAYOUT>
    <ID UID="Layout 1"/>
    <SPREAD>
      <ID UID="1"/>
      <TABLE OPERATION="CREATE" ROWS="5" COLUMNS="3">
        <ID NAME="
STATS"/>
        <GEOMETRY PAGE="1"/>
          <POSITION>
            <TOP>5</TOP>
            <LEFT>5</LEFT>
            <BOTTOM>30</BOTTOM>
            <RIGHT>30</RIGHT>
          </POSITION>
        </GEOMETRY>
        <FRAME WIDTH="1" COLOR="Gray"/>
      </TABLE>
    </SPREAD>
  </LAYOUT>
</PROJECT>
```

Response	Preview of the QuarkXPress project with new table crated in specified position.
Alerts	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address.

	The following is a sample of a transaction entry: 4/10/2007 17:54:37 - tab.qxp - Type: image/jpeg - Size: 9049 - Client: 127.0.0.1
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\createTable.xml where the createBox.xml file exists in the *C:* drive of the server. **On Mac OS** : The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:createTable.xml where the createTable.xml file exists in the xml folder of MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where the *xml-string* tag consists of valid XML commands of image properties. For example: http://localhost:8080/sample.qxp?modify= <LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/> <TABLE OPERATION="CREATE" ROWS="5" COLUMNS="3"><ID NAME="STATS"/> <GEOMETRY PAGE="1"/><POSITION><TOP>5</TOP><LEFT>5</LEFT> <BOTTOM>30</BOTTOM><RIGHT>30</RIGHT></POSITION></GEOMETRY> </TABLE>SPREAD></LAYOUT></PROJECT>

Example, Object Model	Creating new Table: User can add new table specifying the number of rows and columns, to an existing spread of a QuarkXPress project at the specified position using the following code snippet for Table object.
	Spread spread = new Spread();
	Table table = new Table();
	table.name = "textbox1";
	Geometry geometry = new Geometry();
	Position position = new Position();
	position.top = "110";
	position.left = "89";
	position.bottom = "220";
	position.right = "300";
	geometry.position = position;
	geometry.shape = "SH_RECT";
	geometry.page = "1";
	geometry.layer = "Default";
	table.geometry = geometry;
	table.rows = "2";
	table.columns = "4";
	table.maintainGeometry = "true";
	table.operation = "CREATE";
	spread.tables = new Table []{table};
	•
	• While creating a table, position attributes need to be specified.
	•
	• Table is linked to Spread, Spread to Layout, which is further linked with ModifierRequestContents
	•
	• An existing table can be deleted by providing the ID name for a table and

	the operation attribute as “DELETE”

Related topics:

About XML modify
Modifying box properties and content
Deleting boxes
Modifying picture properties
Modifying text attributes

Modifying text attributes

You can use the modify parameter to change the attributes of text in a QuarkXPress project. All modifications are done on a text box basis. To modify text properties, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID
-
- TEXT
-
- STORY
-
- PARAGRAPH
-
- FORMAT
-
- DROPCAP
-
- TABSPEC
-
- TAB
-
- RULE
-
- RICHTEXT

Example

The following XML shows how some of these parameters work.

```
<PROJECT>
  <LAYOUT>
    <ID UID="Layout 1"/>
    <SPREAD>
      <ID UID="1"/>
      <BOX BOXTYPE="CT_TEXT">
        <ID NAME="ABOUT"/>
        <TEXT>
          <STORY CLEAROLDTEXT="true" FITTEXTTOBOX="true" CONVERTQUOTES="true">
            <RICHTEXT FONT="Castellar" PLAIN="true"/>
          </STORY>
        </TEXT>
      </BOX>
      <BOX BOXTYPE="CT_TEXT">
    <ID NAME="HISTORY"/>
    <TEXT>
     <STORY>
      <PARAGRAPH>
       <FORMAT ALIGNMENT="RIGHT"/>
       <RICHTEXT SIZE="12">This text is 12pt and right justified.</RICHTEXT>
      </PARAGRAPH>
     </STORY>
    </TEXT>
   </BOX>
   <BOX BOXTYPE="CT_TEXT">
    <ID NAME="PRODUCTS"/>
    <TEXT>
     <STORY>
      <RICHTEXT BOLD="true">This is bold text.</RICHTEXT>
      <RICHTEXT BOLD="true" COLOR="Red" ITALIC="true" SIZE="20">This text is bold, red,
italic, and 20pt.</RICHTEXT>
     </STORY>
    </TEXT>
   </BOX>
  </SPREAD>
 </LAYOUT>
</PROJECT>
```

Response	Preview of a QuarkXPress project with the values in the ModifierXT tags applied on text boxes.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you enter the wrong XML file as a parameter or when you request a project that does not exists in the document pool. *What to do*: Enter the correct name and path of the XML file and check whether the project given in the request exists in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request. *What to do*: Enter the correct XML file name and resubmit the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed if the text box specified by the child text node of the *ID* tag in the XML file does not exist in the QuarkXPress project. *What to do*: Enter the correct box name or box ID in the XML and resubmit the request.

	The text size value is outside the valid range.	HTTP Error #500 This alert is displayed if the value given in the *SIZE* tag is not valid. *What to do*: Enter a correct text size value. The value of the *SIZE* parameter can vary between 2-720 pts.
	The specified color is not available to the document	HTTP Error #500 This alert is displayed when an invalid color value is given in a *COLOR* tag. *What to do*: Enter a valid color value.
	The specified font is not available	HTTP Error #500 This alert is displayed when an invalid font name is used or a font is specified that does on exist on the server. *What to do*: Enter a correct font value in the *FONTS* tag.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert is displayed when you give a wrong value for a tag. For example, you give a string value for a tag which accepts an integer value. *What to do*: Enter valid values in all tags of the XML file and resubmit the request.
	The specified box cannot be modified.	HTTP Error #500 This alert is displayed when you try to implement text modifier properties on boxes other than text boxes. For example, you use the box ID of a picture box in the XML. *What to do*: Check the box ID or name of the text box in the project and use the same box ID or name in the XML.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, the project name, the type of response produced by the server, the size of the response returned in bytes, and the client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 8/5/2005 13:32:10 - Error - Error Code: 10006 - There is no box with the specified identifier.	
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\modifier.xml where the modifier.xml file exists in the C: drive of the server. **On Mac OS** : The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:modifier.xml where the modifier.xml file exists in the xml folder in MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where *xml-string* consists of valid XML commands of the image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="1"/><SPREAD><ID UID="1"/>	

	<BOX BOXTYPE="CT_TEXT"><ID NAME="BACKGROUND"/> <TEXT><STORY><RICHTEXT FONT="Castella" PLAIN="true"> This is text.</RICHTEXT></STORY></TEXT></BOX></SPREAD> </LAYOUT></PROJECT>

Example 1, Object Model	Request Object Names :
	•
	• ModifierRequest
	•
	• ModifierStreamRequest
	•
	• Project
	•
	• RichText
	•
	• Text
	•
	• ID
	•
	• Box
	•
	• Layout
	•
	• ModifierFileRequest:
	• member contents is used to set the file path
	sdk.QRequestContext rc = new sdk.QRequestContext();
	   if(!this.DocumentSettings1.documentName.Text.Equals(""))
	 rc.documentName = this.DocumentSettings1.documentName.Text;
	 //STEP 2(SPECIFIC TO REQUESTS):Create the Text Modifier renderer request and embed it in request context
	ModifierRequest textReq = new ModifierRequest();
	Project contents = new Project();
	RichText richText1 = new RichText();
	richText1.value = this.text1.Text;
	richText1.color = this.color1.Text;
	Text boxText1 = new Text();
	Story story = new Story();
	story.richText = new RichText[]{richText1};
	boxText1.story = story;
	if(this.fittextbox1.Checked)
	 boxText1.fitTextToBox = "true";
	if(this.clearoldtext1.Checked)
	 boxText1.clearOldText = "true";
	Box box1 = new Box();
	box1.UID = txtBox1;
	box1.text = boxText1;
	Layout layout1 = new Layout();
	layout1.name = layoutText;
	layout1.boxes = new Box[]{box1};
	contents.layouts = new Layout[]{layout1};

	textReq.contents = contents;
	rc.request = textReq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Example 2, Object Model	New text modifier attributes: To edit the properties of an existing text box in a QuarkXPress project, the new object linking is shown below. ModifierRequest < Project < Layout < Spread < Box < Text < Story < Paragraph < RichText The RichText object contains the properties: baselineShift, bold, charstyle, charStyle, color, font, otSwashes, horizontalScale, italic, language, lignatures, merge, opacity, otAllSmallCaps, otContextualFigures, otDenominator, otDiscretionaryLignatures, otFractions, otLiningFigures, otNone, otNumerator, otOrdinals, otPropotionalFigures, otSmallCaps, otStandardLignatures, otSubscript, otSuperscripta, otSwashes, otTabularFigures, otTitlingAlternates, outline, plain, shade, shadow, size, strikethru, subScript, superior, superscript, underline, value (contains the new Text), verticalScale, and wordUnderline. The Story object also contains some text-related properties: fitTextToBox, includeStylesheets, convertQuotes, and clearOldTeaxt.
Notes	• • *A FITTEXTTOBOX tag in an XML file is not applied correctly on render output* • The *FITTEXTTOBOX* tag depends on two preferences, *Allow Text to Grow* and *Font Size*. These preferences are set in the Modifier tab of the **Preferences** dialog box. Choose **QuarkXPress Server > Preferences** to display the **Preferences** dialog box. Choose **Modifier** and then check **Allow Text to Grow**. Specify minimum and maximum values in the **Font Size** area in the **Preferences** dialog box. Submit the request. For more information, please refer to *QuarkXPress Server 7 Guide.pdf*. Use the **Allow Text to Grow** • preferences to increase the text size to fit the text into the box.

Related topics:

About XML modify

Modifying box properties and content

Creating boxes

Deleting boxes

Modifying picture properties

Modifying picture properties

You can modify the properties (such as origin, scale, angle, skew, and orientation) of pictures in a QuarkXPress project with XML. To modify picture properties, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID
-
- PICTURE

Example

The following XML shows how some of these parameters work.

```
<PROJECT>
 <LAYOUT>
  <ID UID="1"/>
  <SPREAD>
   <ID UID="1"/>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="PEOPLE"/>
    <PICTURE SCALEACROSS="50" SCALEDOWN="50" OFFSETACROSS="20"
OFFSETDOWN="20"/>
   </BOX>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="MOUNTAINS"/>
    <PICTURE FIT="CENTERPICTURE" ANGLE="30" SKEW="30"
FLIPHORIZONTAL="false"/>
   </BOX>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="OFFICES"/>
    <PICTURE FIT="FITPICTURETOBOX" ANGLE="30" SKEW="30"
FLIPHORIZONTAL="false"/>
   </BOX>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="PRODUCTS"/>
    <PICTURE FIT="FITPICTURETOBOX" ANGLE="30" SKEW="30"
FLIPHORIZONTAL="false"/>
   </BOX>
   <BOX BOXTYPE="CT_PICT">
    <ID NAME="SERVICES"/>
    <PICTURE FIT="FITPICTURETOBOXPRO"/>
   </BOX>
  </SPREAD>
 </LAYOUT>
```

```
</PROJECT>
```

Response	Preview of a QuarkXPress project with image modifier tags applied on the picture boxes.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect XML file as a parameter or when you request a project that does not exist in the document pool. *What to do*: Enter the correct name and path of the XML file and ensure that the project given in the request exists in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request. *What to do*: Enter the correct XML file name and resubmit the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed if the picture box specified by the child text node of the *ID* tag in the XML file does not exist in the QuarkXPress project. *What to do*: Enter the correct box name or box id in the XML and resubmit the request.
	The value of Scale Across should be between 10% and 1000%.	HTTP Error #500 This alert is displayed if the value of the child text node of the *SCALEACROSS* tag in the XML file is not within the valid range. *What to do*: Enter a valid value in the *SCALEACROSS* tag in XML and resubmit the request.
	The Value of Scale Down should be between 10% and 1000%.	HTTP Error #500 This alert is displayed if the value of the child text node of the *SCALEDOWN* tag in the XML file is not within the valid range. *What to do*: Enter a valid value in the *SCALEDOWN* tag in XML and resubmit the request.
	The value of Offset Across is in invalid range.	HTTP Error #500 This alert is displayed if the value of the child text node of the *OFFSETACROSS* tag in the XML file is not within the valid range. *What to do*: Enter a valid value in the *OFFSETACROSS* tag in XML and resubmit the request.
	The value of Offset Down is in invalid range	HTTP Error #500 This alert is displayed if the value of the child text node of the *OFFSETDOWN* tag in the XML file is not within the valid range.

		What to do: Enter a valid value in the *OFFSETDOWN* tag in XML and resubmit the request.
	The value of Picture Angle must be between -360 and 360 degrees.	HTTP Error #500 This alert is displayed if the value of the child text node of the *ANGLE* tag in the XML file is not within the valid range. *What to do*: Enter a valid value in the *ANGLE* tag in XML and resubmit the request.
	The value of Picture Skew must be between -75 and 75 degrees.	HTTP Error #500 This alert is displayed if the value of the child text node of the *SKEW* tag in the XML file is not within the valid range. *What to do*: Enter a valid value in the *SKEW* tag in XML and resubmit the request.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert is displayed when you enter an incorrect value for a tag. For example, you enter a string value for a tag that accepts a numeric value. *What to do*: Enter a valid value in all tags of XML and resubmit the request.
	The specified box cannot be modified.	HTTP Error #500 This alert is displayed when you try to implement image modifier properties on boxes other than picture boxes. For example, you use the box ID of a text box in the XML. *What to do*: Check the box ID or name of the picture box in the project and use the same box ID or name in the XML.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 11:27:42 - jpeg/sample.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of an error log entry: 8/10/2005 10:39:07 - Error - Error Code: 10339 - The specified file failed to load in the picture box.	
Example GET URL	http://localhost:8080/sample.qxp?modify=file:C:\imageProperties.xml where the imageProperties.xml file exists in the *C:* drive of the server. **On MacOS** : The URL format is: http://localhost:8080/sample.qxp?modify=file:MacHD:xml:imageProperties.xml where the ImageProperties.xml file exists in the xml folder of MacHD. http://localhost:8080/sample.qxp?modify=<xml-string> where the *xml-string* tag consists of valid XML commands of image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="1"/><SPREAD>	

	<ID UID="1"/><BOX BOXTYPE="CT_PICT"><ID NAME="EVEREST"/> <PICTURE SCALEACROSS="50" OFFSETDOWN="20" ANGLE="30" FIT="CENTERPICTURE" SKEW="30" FLIPHORIZONTAL="false"/></BOX></SPREAD> </LAYOUT></PROJECT>

Example 1, Object Model	Request Object Names :
	•
	• ModifierRequest
	•
	• ModifierStreamRequest
	•
	• Project
	•
	• Box
	•
	• Picture
	•
	• Layout
	•
	• ModifierFileRequest:
	• member contents is used to set the file path
	QRequestContext rc = new sdk.QRequestContext();
	   if(!this.DocumentSettings1.documentName.Text.Equals(""))
	 rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the Image Modifier renderer request and embed it in
	ModifierRequest imgReq = new ModifierRequest();
	Project contents = new Project();
	Picture picture1 = new Picture();
	picture1.scaleAcross = this.scaleacross1.Text;
	picture1.scaleDown = this.scaledown1.Text;
	if(this.fitpicturebox1.Checked == true)
	 picture1.fitPictureToBox = "true";
	if(this.flipvertical1.Checked == true)
	 picture1.flipVertical = "true";
	if(this.fliphorizontal1.Checked == true)
	 picture1.flipHorizontal = "true";
	Box box1 = new Box();
	box1.UID = txtBox1;
	box1.picture = picture1;
	Layout layout1 = new Layout();
	layout1.name = layoutText;
	imgReq.contents = contents;
	contents.layouts = new Layout[]{layout1};
	layout1.boxes = new Box[]{box1};
	rc.request = imgReq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Example 2, Object Model	New Image Modifier: To edit the properties of an existing image box in a QuarkXPress project, the new object linking is shown below. ModifierRequest < Project < Layout < Spread < Box < Picture The Picture object contains the properties: angle, fit, flipHorizontal, flipVertical, fullRes, mask, offsetAcross, offsetDown, opacity, picColor, scaleAcross, scaleDown, shade, skew, and supressPict.
Notes	• • You cannot change an image with the Modifier XTensions software. You can only manipulate an image that already exists. • • If you specify FITPICTURETOBOX, FITBOXTOPICTURE, and FITPICTURETOBOXPRO for a picture, then only the first of them will be applied.

Related topics:

About XML modify
Modifying box properties and content
Creating boxes
Deleting boxes
Modifying text attributes

Importing data

Imports text or image data into a project.

To import text or image data into a project, use the following parameters in the Modifier DTD:

-
- BOX
-
- ID
-
- PICTURE
- Note: PICTURE is not a required element when importing data.
-
- TEXT
-
- STORY
-
- CONTENT

Example

The following XML shows how some of these parameters work.

```
<PROJECT>
    <ID NAME="Layout 1"/>
    <SPREAD>
      <ID UID="1"/>
<BOX BOXTYPE="CT_PICT">
        <ID NAME="ABOUT"/>
        <PICTURE/>
        <CONTENT>file:c:\docs\file1.jpg</CONTENT>
      </BOX>
<BOX BOXTYPE="CT_TEXT">
        <ID NAME="PRODUCTS"/>
        <CONTENT>file:c:\docs\file2.txt</CONTENT>
      </BOX>
<BOX BOXTYPE="CT_TEXT">
        <ID NAME="SERVICES"/>
        <TEXT>
         <STORY FILE="file:c:\docs\file3.doc" CONVERTQUOTES="true"
                INCLUDESTYLESHEETS="true"/>
        </TEXT>
      </BOX>
    </SPREAD>
  </LAYOUT>
</PROJECT>
```

Response	Preview of a QuarkXPress project with a value in the data import XML tags applied on the text boxes.	
Alerts	File not found.	HTTP Error #404

		QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect file name as a parameter or when you request a document that does not exist in the document pool. *What to do*: Enter the correct name and path of the file and make sure the document exists in the document pool.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Enter correct XML and resubmit the request.
	There is no box with the specified identifier.	HTTP Error #500 This alert is displayed if the box, as specified by the child node of the ID tag in the XML file, does not exist in the QuarkXPress file. *What to do*: Enter a correct box name or box ID in the XML file and resubmit the request.
	The specified box is not a picture or text box.	HTTP Error #500 This alert is displayed when you request a box that is not a text box or a picture box. *What to do*: Use a correct text box name or picture box name in the request.
	A locked layer cannot be manipulated.	HTTP Error #500 This alert is displayed when you request data from a locked layer box. *What to do*: You cannot modify the content of a box placed on a locked layer. To get data into a box placed on a locked layer, open the project in QuarkXPress and open the Layers palette. Unlock the layer on which the box is placed. Save the project and submit the render request again.
	Unable to read picture (#106)	HTTP Error #500 QuarkXPress Server Error #-109 This alert is displayed when you request a text file in a picture box. *What to do*: Enter the correct file name and file type in the request.
	Bad filename/pathname	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when you request an invalid or non-existent file in a box. *What to do*: Enter the correct file name and file type in the request.
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, the render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/5/2005 18:11:54 - sample.qxp - Type: image/jpeg - Size: 65982 - Client: 127.0.0.1	

	If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/5/2005 18:01:59 - Error - Error Code: 10343 - A locked Layer cannot be manipulated.
Example GET URL	On Windows, the URL format is: http://localhost:8080/Sample.qxp?modify=file:c:\file.xml On MAC OS, the URL format is: http://localhost:8080/Sample.qxp?modify=file:HDD:file.xml To use an XML string in the URL: http://localhost:8080/sample.qxp?modify=<xml-string> where *xml-string* consists of valid XML of image properties. For example: http://localhost:8080/sample.qxp?modify= <PROJECT><LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/> <BOXBOXTYPE="CT_TEXT"><ID NAME="TREES"/> <CONTENT>file:c:\docs\file1.jpg</CONTENT> </BOX></SPREAD></LAYOUT></PROJECT> Path Parameter http://localhost:8080/Sample.qxp?textboxname@dataimport=file:c:\file.txt http://localhost:8080/Sample.qxp?pictureboxname@dataimport=c:\file.jpg Text Parameter http://localhost:8080/Sample.qxp?textboxname@dataimport=Newdata Stylesheets http://localhost:8080/Documentname?textboxname@dataimport= file:c:\file.doc&textboxnameincludestylesheets@dataimport=yes Convert quotes http://localhost:8080/Documentname?textboxname@dataimport= file:c:\file.doc&textboxnameconvertquotes@dataimport=yes

Example, Object Model	**Request Object Names:**
	•
	• ModifierRequest
	•
	• ModifierStreamRequest
	•
	• Project
	•
	• Box
	•
	• Content
	•
	• Layout
	•
	• ModifierFileRequest: member contents is used to set the file path or send the XML itself.
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the data import request and

	embed it in request context
	ModifierRequest request = new ModifierRequest();
	Project
	requestContents = new Project();
	Content boxContent1 = new Content();
	Box box1 = new Box();
	box1.UID = txtBox1;
	box1.content = boxContent1;
	Layout layout1 = new Layout();
	layout1.name
	= layoutText;
	if(!this.content1.Text.Equals(""))
	{
	boxContent1.value = this.content1.Text;
	Text text1 = new Text();
	text1.font = this.fontname1.Text;
	box1.text = text1;
	if(this.includestylesheets1.Checked == false)
	boxContent1.includeStylesheets = "false";
	if(this.convertquotes1.Checked == false)
	boxContent1.convertQuotes = "false";
	}
	else if (null != uplTheFile.PostedFile)
	{
	Stream theStream = uplTheFile.PostedFile.InputStream;
	StreamReader reader = new StreamReader(theStream);
	boxContent1.value = reader.ReadToEnd();
	}
	layout1.boxes = new Box[]{box1};
	requestContents.layouts = new Layout[]{layout1};
	request.contents = requestContents;
	rc.request = request;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • Advantages of using the Modifier XTensions software over the BoxParam XTensions software • Using the Modifier XTensions software, you can load contents to boxes from files located anywhere on the computer or at any accessible network location. This is not possible with the BoxParam XTensions software. With the BoxParam XTensions software, you can only load contents into boxes from files that are located in the document pool of the server. • • You can use Modifier XTensions software to import any file format supported by QuarkXPress. The file can be a text file or image file. • • You can use Modifier XTensions software to import an XTags file generated by QuarkXPress.

Exporting Job Jackets files during project deconstruction

While using the xml namespace to deconstruct a QuarkXPress project, you can specify the jjname parameter in the same request to output the Job Jackets file to the document pool.

For example:

http://localhost:8080/xml/project.qxp?jjname=jjfilename.xml

You can then use the

construct namespace to create new QuarkXPress projects that are based on that Job Jackets file's resources and layout specifications.

Note: The jjname parameter exports QuarkXPress project resources and layout specifications to a Job Ticket. Resources defined at Job Jackets level are not exported to the Job Ticket created using the jjname parameter.

About XML deconstruct/reconstruct

Versions of QuarkXPress Server prior to 7.2 allowed you to retrieve a human-readable XML representation of a QuarkXPress project from the server using the
deconstruct namespace, but these versions did not provide an easy way to turn that XML back into a QuarkXPress project.

QuarkXPress Server includes the xml namespace (which deconstructs a project according to the Modifier DTD) and the
construct namespace (which lets the server turn an XML representation of a QuarkXPress project back into a QuarkXPress project). This means you can deconstruct a project into an XML representation, change the XML in accordance with the Modifier DTD, and then have the server generate an updated version of the QuarkXPress project. You can even create new QuarkXPress projects from scratch using XML.

In addition, you can use the new
construct namespace to:

-
- Create a page based on a master page
-
- Create a project from XML, using a Job Jackets file as the basis for the project
-
- Modify text font and style, including OpenType styles
-
- Apply style sheets and local formatting to text
-
- Create and populate tables
-
- Import pictures into picture boxes and specify picture attributes

The DTD used for XML construction and deconstruction is completely Unicode-compliant, making it ideal for use in international publishing. Furthermore, the use of this DTD ensures that the schema of XML output created by Constructor does not change when server preferences change.

Note: Some minor QuarkXPress features are not available through the Modifier DTD. However, this DTD represents the majority of all user-editable aspects of a QuarkXPress project.

Note: The
deconstruct namespace/request no longer exists. If you try to use it in QuarkXPress Server 8.0, an error will be returned.

The xml and construct namespaces

The
xml namespace returns an XML representation of the target project. To use this namespace, use a URL like the following:

http://QXPServer8:8080/xml/project1.qxp

When you use the
xml namespace, QuarkXPress Server returns an XML file that represents the deconstructed project. This XML file adheres to the Modifier DTD. This chapter provides a general introduction to working with this DTD, and goes into detail about some of the less obvious aspects of the process.
The
construct namespace takes two arguments: The name of the project to be created and a modify parameter that points to the XML file or string that describes how to create the project. These two arguments display as follows:
http://QXPServer8:8080/construct/project1.qxp?modify=file:path to XML file on server
or:
http://QXPServer8:8080/construct/project1.qxp?modify=XML string
Note: There is a length limitation of 4096 characters on URLs, so you will probably want to use an XML file rather than an XML string.
Note: If you are using QuarkXPress Server Manager, you can send a similar command with a QuarkXPress Server Manager URL or through Web services.
Note that the modify parameter still lets you modify existing projects. If you are modifying an existing document, however, you should not use the construct namespace.
http://QXPServer8:8080/project1.qxp?modify=file:path to XML file on server
or:
http://QXPServer8:8080/project1.qxp?modify=XML string

Construct vs. modify

It's important to understand that although the construct namespace uses the same DTD that you use when you modify an existing project, the
construct namespace uses it differently.
When you use the construct namespace, the XML you pass simply contains a description of everything in the document you want to create — much as an HTML file describes a page you want to display in a browser. There is no need to use a command and create elements such as ADDCELLS, OPERATION, and MOVERIGHT; you simply describe each item in the layout with elements such as <BOX> and <TABLE>, and specify each item's position with the
<POSITION> element type.
When you use the modify attribute without the
construct namespace, however, the XML you pass must contain commands that show how you want QuarkXPress Server to modify the project.

Related topics:

The Modifier DTD
xml
construct
Deconstructing a project
Constructing a project
Working with pages and spreads
Working with layers
Working with boxes
Working with pictures
Working with text
Working with tables
Working with Composition Zones
Using XSL transformation

xml

Creates an XML file from a QuarkXPress project. The XML is returned in a fixed format that adheres to the Modifier DTD. You can easily use the returned XML to create or modify a QuarkXPress document using the construct namespace or modify parameter.

Namespace	xml	
DTD	Modifier DTD	
Parameters	box	Returns XML only for the given box ID or name.
	boxes	Returns XML only for the box IDs or names

		given as a commaseparated list.
	XSL	Specifies a path to an XSL file that describes how to transform the returned XML. The path for the

		XSL file is specified by the *file:* indicator.
	layout	Specifies the layout name or number to render. Layout numbers start with 1; layout=1 refers to the

		first layout in the project. You can also specify the layout name with this parameter.
	Refer to the Modifier DTD	

Response	The following XML code breaks down the page elements of a project into the XML code for the individual layers, text boxes, and picture boxes.
	<?xml version="1.0" encoding="UTF-8" standalone="no"?>
	<PROJECT JOBJACKET="Macintosh HD:QuarkXPress DocPool:
	default job jackets:New Job Jacket.xml"
	        JOBTICKET="Default Job Ticket"
	        PROJECTNAME="project1.qxp">
	  <LAYOUT>
	    <ID NAME="Layout 1" UID="1"/>
	    <LAYER KEEPRUNAROUND="false" LOCKED="false" SUPPRESS="false" VISIBLE="true">
	      <ID NAME="Default" UID="-1"/>
	      <RGBCOLOR BLUE="231" GREEN="231" RED="231"/>
	    </LAYER>	

```
   <SPREAD>
     <ID UID="1"/>
     <PAGE MASTER="3"
POSITION="RIGHTOFSPINE">
       <ID UID="1"/>
     </PAGE>
      <BOX BOXTYPE="CT_TEXT"
COLOR="None" OPACITY="100%"
SHADE="100%">
       <ID NAME="Introduction" UID="5"/>
       <GEOMETRY LAYER="Default" PAGE="1"
SHAPE="SH_RECT">
         <POSITION>
          <TOP>39.064</TOP>
          <LEFT>39.026</LEFT>
          <BOTTOM>63.951</BOTTOM>
          <RIGHT>214.611</RIGHT>
         </POSITION>

<SUPPRESSOUTPUT>false</SUPPRESSOUTPU
T>
         <RUNAROUND TYPE="NONE"/>
       </GEOMETRY>pre
       <FRAME GAPCOLOR="White"
GAPOPACITY="100%" GAPSHADE="100%"
OPACITY="100%" SHADE="100%"
STYLE="Solid" WIDTH="0 pt"/>
       <TEXT>
         <STORY>
          <COPYFIT FITAMOUNT="0.033&quot;"
NUMBEROFCHARACTERS="6"
NUMBEROFLINES="1"
NUMBEROFWORDS="1" STATE="underFit"/>
          <PARAGRAPH PARASTYLE="launch">
            <RICHTEXT
CHARSTYLE="launch">LAUNCH</RICHTEXT>
          </PARAGRAPH>
         </STORY>
       </TEXT>
      </BOX>
      <BOX BOXTYPE="CT_PICT"
COLOR="None" OPACITY="100%"
SHADE="100%">
       <ID NAME="Sunrise" UID="6"/>
       <PICTURE SCALEACROSS="100%"
SCALEDOWN="100%"/>
       <CONTENT>Macintosh HD:QuarkXPress
DocPool:sunrise.tif</CONTENT>
      <GEOMETRY LAYER="Default" PAGE="1"
SHAPE="SH_RECT">
```

	<POSITION> <TOP>0</TOP> <LEFT>0</LEFT> <BOTTOM>800</BOTTOM> <RIGHT>600</RIGHT> </POSITION> <SUPPRESSOUTPUT>false</SUPPRESSOUTPUT> <RUNAROUND BOTTOM="0" LEFT="0" RIGHT="0" TOP="0" TYPE="ITEM"/> </GEOMETRY> <FRAME GAPCOLOR="White" GAPOPACITY="100%" GAPSHADE="100%" OPACITY="100%" SHADE="100%" STYLE="Solid" WIDTH="0"/> <PICTURE/> </BOX> </SPREAD> </LAYOUT> </PROJECT>
Alerts	
Logs	If the project is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, project name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 17:16:11 - xml/sample.qxp - Type: text/xml - Size: 2364 - Client: 127.0.0.1
Example GET URL	http://localhost:8080/xml/sample.qxp Note: You can also deconstruct QuarkCopyDesk articles. To deconstruct a QuarkCopyDesk article, use the following: http://localhost:8080/xml/copydesk/abc.qcd

Related topics:

The Modifier DTD
About XML deconstruct and construct
construct
Deconstructing a project
Constructing a project

construct

The construct namespace lets you create a QuarkXPress project using XML.

Namespace	construct		
DTD	Modifier DTD		
Parameters	modify	string	Specifies the XML file or string that describes how to create the project. The XML file is specified by the *file:* indicator. Note: The XML file must adhere to the Modifier DTD and be present in the specified location.
	qxpdocver	7 | 8	Returns a QuarkXPress document. For example: http://QXPServer8:8080/construct/qxpdoc/project1.qxp?qxpdocver=7
Example GET URL	http://QXPServer8:8080/construct/project1.qxp?modify=file:sample.xml		
Example XML	<?xml version="1.0" encoding="UTF-8"?> <PROJECT JOBJACKET="C:\XML\New Job Jacket 3.xml" JOBTICKET="Default Job Ticket" PROJECTNAME="project1.qxp"> <LAYOUT> <ID NAME="Layout 1"/> <SPREAD> <ID UID="1"/> <PAGE> <ID UID="1"/> </PAGE> </SPREAD> </LAYOUT> </PROJECT>		
Response	The new QuarkXPress project		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you give an incorrect XML file as a parameter or when you request a document that does not exist in the document pool. *What to do*: Enter the correct name and path of the XML file and ensure that the document given in the request exists in the document pool.	
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #-37 This alert is displayed when an invalid file name is entered in the request. *What to do*: Enter the correct XML file name and resubmit	

		the request.
	The XML document is not valid or well formed.	HTTP Error #500 This alert is displayed when XML tags are not correctly formed. *What to do*: Provide correct XML and resubmit the request.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert is displayed when you enter an incorrect value for a tag. For example, you enter a string value for a tag that accepts a numeric value. *What to do*: Enter a valid value in all tags of XML and resubmit the request.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, project name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: *8/3/2005 11:27:42 - jpeg/construct/table.qxp - Type: image/jpeg - Size: 31715 - Client: 127.0.0.1* If an alert is displayed, an error message is written to the QuarkXPress Server error log. The following is a sample of the error log entry: *8/10/2005 10:39:07 - Error - Error Code: 10339 - The specified file failed to load in the picture box.*	

Example, Object Model	**Request Object Names:**
	•
	• XMLRequest
	•
	• ConstructRequest
	•
	• ConstructFileRequest
	•
	• ConstructStreamRequest
	Steps to construct a new QuarkXPress project by editing an existing document
	a. Deconstruct a QuarkXPress project using the following code snippet:
	XMLRequest dcnstrq = new XMLRequest();
	rc.request = dcnstrq;
	b. Alter the project by manipulating the XML.
	c. Pass the modified XML document to ConstructStreamRequest to create a new QuarkXPress project, using the following code snippet:
	ConstructStreamRequest cnstrq = new ConstructStreamRequest();
	cnstrq.modify = Buffer;
	rc.request = cnstrq;
	QuarkXPressRenderRequest qxprq = new QuarkXPressRenderRequest();
	cnstrq.request = qxprq;
	Where Buffer contains the Byte[] for the modified XML document.
	Alternate steps to construct a new QuarkXPress project from an existing document	

	a. Deconstruct a QuarkXPress project using the following code snippet:
	QManagerSDKSvcService svc = new QManagerSDKSvcService()
	Project proj = svc.getDOM("document.qxp");
	b. Alter the project by manipulating the XML.
	c. Pass the modified Project instance to
	ConstructRequest to create a new QuarkXPress project using the following code snippet:
	ConstructRequest cnstrq = new ConstructRequest();
	cnstrq.project = proj;
	QRequestContext rc = new QRequestContext();
	rc.request = cnstrq;
	QuarkXPressRenderRequest qxprq = new QuarkXPressRenderRequest();
	cnstrq.request = qxprq;

Notes	The construct namespace takes two arguments: The name of the project to be created and a modify parameter that points to the XML file or string that describes how to create the project: http://localhost:8080/construct/project1.qxp?modify=file:path to XML file on server http://localhost:8080/construct/project1.qxp?modify=<xml-string>

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

Deconstructing a project

Constructing a project

Deconstructing a project

An XML file that represents a deconstructed project does not contain all of the information necessary to reconstruct the project. The definitions of the project's resources (such as style sheets, colors, and master page definitions) are stored in a Job Jackets file.

For example, you will see later in this section that you can apply a style sheet to a paragraph by indicating the style sheet’s name, like so:

```
<PARAGRAPH PARASTYLE="BodyText">
  <RICHTEXT>The sun has risen.</RICHTEXT>
</PARAGRAPH>
```

The above information is included in the deconstructed project’s XML file. The definition of the “BodyText” style sheet, however, is stored in the Job Jackets file.

The URL of a deconstructed Job Jackets file is indicated by the PROJECT@JOBJACKET attribute. If you need access to new colors, style sheets, master pages, or other resources, add them to the Job Jackets file indicated by this URL.

Note: Projects can also refer to resources defined with the QuarkXPress Server's Document Controls submenu (
Server/QuarkXPress Server menu). QuarkXPress Server looks for resources first in the Job Jackets file and then in the server-defined resources.

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Constructing a project

Constructing a project

Every project created with the construct namespace must be based on a Job Ticket in a Job Jackets file. Using construct to create a project is roughly equivalent to using the
File > New > Project from Ticket command in QuarkXPress.

When you create a project using the construct namespace, you must supply the path to the Job Jackets file that will supply the project's resources. To do so, indicate the URL of the Job Jackets file in the PROJECT@JOBJACKET attribute and the name of the Job Ticket in the PROJECT@JOBTICKET attribute. (
<PROJECT> is the root element of the Modifier DTD.)

For example, to create a project from a Job Ticket named "Tall US Brochure Ticket" in a Job Jackets file named “BrochureJJ.xml,” you could use the following XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<PROJECT JOBJACKET="MacintoshHD:brochures:BrochureJJ.xml"
      JOBTICKET="Tall US Brochure Ticket"
      PROJECTNAME="project1.qxp">
```

Related topics:

The Modifier DTD
About XML deconstruct and construct
Exporting Job Jackets files during project deconstruction
xml
construct
Deconstructing a project

Working with pages and spreads

The root element of a deconstructed QuarkXPress project is <PROJECT>. Within each <PROJECT> element are one or more <LAYOUT> elements. Each layout contains one or more <SPREAD> elements, and each <SPREAD> contains one or more <PAGE> elements. Each layout, spread, and page has a unique name, indicated by its
<ID> element.

Each layout can have a unique name, indicated by its <ID> element's NAME attribute. You can use a layout's name when referring to that layout in a non-construct call that uses the MODIFY attribute. The ID@NAME attribute is ignored for <SPREAD> and <PAGE> elements, but you can refer to them numerically with their <ID> element's UID attribute, with "1" being the first,
"2" being the second, and so forth.

Note: With most element types, it is best to assign an ID@NAME value to an element and use that to refer to the element, because ID@UID values are defined by QuarkXPress Server and thus ignored for construct calls. <PAGE> and
<SPREAD> are exceptions to this rule.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<PROJECT JOBJACKET=" MacintoshHD:brochures:BrochureJJ.xml"
     JOBTICKET="Tall US Brochure Ticket"
     PROJECTNAME="project1.qxp">
   <LAYOUT>
     <ID NAME="Layout 1" />
     <SPREAD>
        <ID UID="1" />
        <PAGE POSITION="RIGHTOFSPINE" MASTER="3">
          <ID UID="2" />
        </PAGE>
        ...
```

Each page has a POSITION attribute that indicates which side of the spine it is on. (In single-sided layouts, every page is given a POSITION of
RIGHTOFSPINE)

Assigning items to specific pages

You can assign items to a page using the GEOMETRY element, which is a child of the BOX and TABLE elements. For example:

```
<BOX BOXTYPE="CT_TEXT" COLOR="White">
     <ID NAME="Title Box" />
   <GEOMETRY LAYER="Default" PAGE="1"
 SHAPE="SH_RECT">
      <POSITION>
         <TOP>90</TOP>
         <LEFT>95</LEFT>
      <BOTTOM>190</BOTTOM>
      <RIGHT>195</RIGHT>
    </POSITION>
  </GEOMETRY>
</BOX>
```

Creating pages from master pages

Master pages are stored in a deconstructed project’s Job Jackets file. To create a page from this master page, insert a MASTER attribute into the PAGE element and indicate the number of the target master page. Master page numbering is as follows:

1 = blank single page
2 = blank facing-page
3 = the first user-defined master page in the Job Jackets file (by default, the master page named "A-Master A")

For example, to create a master page based on the first user-defined master page in the Job Jackets file, you could use XML like the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<PROJECT JOBJACKET=" file://brochures/BrochureJJ.xml"
     JOBTICKET="Tall US Brochure Ticket"
     PROJECTNAME="project1.qxp">
  <LAYOUT>
    <ID NAME="Layout 1"/>
    <SPREAD>
      <ID UID="1" />
      <PAGE MASTER="3" POSITION="LEFTOFSPINE">
        <ID UID="2" />
      </PAGE>
      ...
```

Note that each page has a POSITION attribute that indicates where that page falls with regard to the spine.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

Working with layers

To create a layer in XML, use the LAYER element. For example:

```
<LAYER KEEPRUNAROUND="true" LOCKED="false"
    SUPPRESS="false" VISIBLE="true">
  <ID NAME="Layer 1" />
</LAYER>
```

The RGBCOLOR element defines the layer's color as displayed in the Layers palette.

Assigning items to layers

You can assign items to a layer using the GEOMETRY element, which is a child of the BOX and TABLE elements. For example:

```
BOX BOXTYPE="CT_TEXT" COLOR="White">
    <ID NAME="Main Layer" />
  <GEOMETRY LAYER="Default"
 PAGE="1" SHAPE="SH_RECT">
    <POSITION>
      <TOP>90</TOP>
      <LEFT>95</LEFT>
      <BOTTOM>190</BOTTOM>
      <RIGHT>195</RIGHT>
    </POSITION>
  </GEOMETRY>
</BOX>
```

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Deconstructing a project

Constructing a project

Working with boxes

To add text and pictures to a project, you must add text boxes and picture boxes to the project’s <SPREAD> element. Both are represented by <BOX> elements, but text boxes have a BOXTYPE attribute of CT_TEXT, and picture boxes have a BOXTYPE attribute of CT_PICT. You can read about how <BOX> elements are put together in the Modifier DTD, but for purposes of illustration, the sample XML below describes a spread that contains a text box and a picture box.

```
<SPREAD>
  <PAGE MASTER="3" POSITION="LEFTOFSPINE">
    <ID UID="1" />
     <!-- TEXT BOX -->
    <BOX BOXTYPE="CT_TEXT" COLOR="White">
      <ID NAME="Headline Box" />
      <GEOMETRY LAYER="Default" PAGE="1" SHAPE="SH_RECT">
        <POSITION>
          <TOP>200</TOP>
          <LEFT>80</LEFT>
          <BOTTOM>450</BOTTOM>
          <RIGHT>475</RIGHT>
        </POSITION>
      </GEOMETRY>
      <TEXT>
        <STORY>
          <PARAGRAPH PARASTYLE="Normal">
            <RICHTEXT>This is text in a box.</RICHTEXT>
          </PARAGRAPH>
        </STORY>
      </TEXT>
    </BOX>
     <!-- PICTURE BOX -->
    <BOX BOXTYPE="CT_PICT">
      <ID NAME="Main Story Photo" />
      <GEOMETRY LAYER="Default" PAGE="1" SHAPE="SH_RECT">
        <POSITION>
          <TOP>90</TOP>
          <LEFT>95</LEFT>
          <BOTTOM>190</BOTTOM>
          <RIGHT>195</RIGHT>
        </POSITION>
      </GEOMETRY>
      <PICTURE ANGLE="0°" FLIPHORIZONTAL="false"
        FLIPVERTICAL="false" FULLRES="false" MASK="None"
        OFFSETACROSS="0 OFFSETDOWN="0" OPACITY="100%"
        SCALEACROSS="100%" SCALEDOWN="100%" SHADE="100%"
        SKEW="0°" SUPRESSPICT="false"/>
      <CONTENT>Macintosh HD:DocPool:flower1.jpg</CONTENT>
    </BOX>
   </PAGE>
</SPREAD>
```

This example will work for a construct request. For a modify request, add the attribute value OPERATION="CREATE" in the
BOX element.

All BOX elements can contain a GEOMETRY element that indicates the position and size of the box, a FRAME element that describes the box's frame (if any), and a SHADOW element that describes the box's drop shadow. Additional
BOX elements are described in the following sections.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

Working with pictures

The
<PICTURE> element supports a variety of new features, including the ability to specify runaround, opacity, and drop shadow characteristics. For more information, see the Modifier DTD .

```
<PROJECT>
 <LAYOUT>
  <ID NAME="Layout 1"/>
  <SPREAD>
   <ID UID="1"/>
   <BOX COLOR="Magenta" SHADE="50%" OPACITY="100%">
    <ID NAME="pict1"/>
    <PICTURE MASK="Test Alpha1"/>
    <FRAME STYLE="Triple" WIDTH ="5" COLOR="Cyan" SHADE="100%"
OPACITY="100%" GAPCOLOR="Yellow" GAPSHADE="80%" GAPOPACITY="100%"/>
   </BOX>
   <BOX>
    <ID NAME="pict2"/>
    <PICTURE SUPRESSPICT="true" FULLRES="true" PICCOLOR="Cyan" SHADE="90"
OPACITY="90"/>
    <SHADOW COLOR="Cyan" SHADE="90" ANGLE="130" OPACITY="100" DISTANCE="5"
SKEW="10" SCALE="90" BLUR="3"/>
   </BOX>
   <BOX>
    <ID NAME="pict3"/>
    <GEOMETRY>
<RUNAROUND TYPE="NONWHITEAREAS"  OUTSET="10" NOISE="5" SMOOTHNESS="5"
THRESHOLD="10" INVERT="true" OUTSIDEONLY="true" RESTRICTTOBOX="true"/>
    </GEOMETRY>
    </BOX>
    <BOX>
     <ID NAME="pict4"/>
     <PICTURE FIT="FITPICTURETOBOX" SCALEACROSS="40" SCALEDOWN="50"
FLIPVERTICAL="true" FLIPHORIZONTAL="false" ANGLE="40" SKEW="20"/>
   </BOX>
  </SPREAD>
 </LAYOUT>
</PROJECT>
```

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Deconstructing a project

Constructing a project

Working with text

Every <BOX> element for text contains a <TEXT> element, and every <TEXT> element contains a <STORY> element. A <STORY> element can contain <PARAGRAPH> elements, each of which contains <RICHTEXT> elements. A <STORY> element can also simply contain <RICHTEXT> elements.

A text <BOX> element can also contain a <CONTENT> element that indicates the origin of the text in that box.

Finally, a text <BOX> element in a deconstructed project can contain <PLACEHOLDER> elements, which allow XML Import XTensions software to insert text from a different XML source. Note that <PLACEHOLDER> elements are ignored by the construct namespace and the modify parameter; placeholders must be inserted in QuarkXPress using XML Import XTensions software.

Applying style sheets

Like other resources, style sheets are defined in a deconstructed project's Job Jackets file. To apply a paragraph style sheet to text, use the PARASTYLE attribute of the <PARAGRAPH> element. For example, to apply the paragraph style sheet named “BodyText” to a paragraph, you could use XML like the following:

```
<PARAGRAPH PARASTYLE="BodyText">
   <RICHTEXT MERGE="true">The sun has risen.</RICHTEXT>
</PARAGRAPH>
```

To apply a character style sheet to text, use the CHARSTYLE attribute of the < RICHTEXT> element. For example, to apply the character style sheet named “Emphasis” to a word, you could use XML like the following:

```
<PARAGRAPH PARASTYLE="BodyText">
   <RICHTEXT>The </RICHTEXT>
   <RICHTEXT CHARSTYLE="Emphasis">sun</RICHTEXT>
   <RICHTEXT> has risen.</RICHTEXT>
</PARAGRAPH>
```

Applying local formatting

To apply local formatting to text, use the attributes of the <RICHTEXT> element. For example:

```
<PARAGRAPH>
   <RICHTEXT
   SIZE="10" COLOR="Magenta" BOLD="true" OPACITY="50%"
   >The sun has risen.</RICHTEXT>
</PARAGRAPH>
```

To apply paragraph formatting, use a <FORMAT> element, like so:

```
<PARAGRAPH>
   <FORMAT SPACEBEFORE="6" SPACEAFTER="2" LEADING="24"
       ALIGNMENT="LEFT” KEEPWITHNEXT="true">
```

```
    <RICHTEXT>The sun has risen.<RICHTEXT>
  </FORMAT>
</PARAGRAPH>
```

The MERGE attribute lets you control whether formatting from one <RICHTEXT> or <PARAGRAPH> element is carried forward to the next. For example, the following XML would result in "has risen" being italicized:

```
<PARAGRAPH PARASTYLE="BodyText">
  <RICHTEXT SIZE="10">The </RICHTEXT>
  <RICHTEXT SIZE="12"ITALIC="TRUE">sun</RICHTEXT>
  <RICHTEXT MERGE="true" SIZE="10"> has risen.</RICHTEXT>
</PARAGRAPH>
```

However, this XML would result in "has risen" being plain:

```
<PARAGRAPH PARASTYLE="BodyText">
  <RICHTEXT SIZE="10">The </RICHTEXT>
  <RICHTEXT SIZE="12" ITALIC="TRUE">sun</RICHTEXT>
  <RICHTEXT MERGE="false" SIZE="10"> has risen.</RICHTEXT>
</PARAGRAPH>
```

The default value for < MERGE> is "false."

Formatting across paragraph boundaries

You can use two methods to describe a run of formatting that crosses a paragraph boundary. The first is to simply close the first <PARAGRAPH> element and then open a new one. For example:

```
<PARAGRAPH>
  <RICHTEXT SIZE="10">The sun has risen.</RICHTEXT>
</PARAGRAPH>
<PARAGRAPH>
  <RICHTEXT SIZE="10">The sun has set.</RICHTEXT>
</PARAGRAPH>
```

The second is to use a &harRetrun; entity to create the paragraph break. For example:

```
<PARAGRAPH>
  <RICHTEXT SIZE="10"
  >The sun has risen.&harRetrun;The sun has set.</RICHTEXT>
</PARAGRAPH>
```

Combining style sheets with local formatting

To combine local formatting with style sheets, simply add attributes to the <RICHTEXT> elements within a <PARAGRAPH> element. For example:

```
<PARAGRAPH PARASTYLE="BodyText">
  <RICHTEXT COLOR="Red">The </RICHTEXT>
  <RICHTEXT COLOR="Yellow" CHARSTYLE="Emphasis">sun</RICHTEXT>
  <RICHTEXT COLOR="Red"> has risen.</RICHTEXT>
</PARAGRAPH>
```

Retrieving copyfitting information

In deconstructed projects, a <BOX> element can contain a <LINKEDBOX> element. The <LINKEDBOX> element indicates the point where text has overflowed the current box and identifies the box where the text continues. The
<LINKEDBOX> element also contains attributes that indicate where in the text the break occurs.
In a <STORY> element, the <OVERMATTER> element indicates where the current box overflows when there is no subsequent box for text to flow into. A <STORY> element also contains a <COPYFIT> element indicating how many words, characters, and lines should be allowed to fit in that box and whether the text currently fits in the box, is too short, or is too long. This information can be useful for on-the-fly copyfitting.
Note: The elements described in this section occur only in deconstructed project XML generated by the xml namespace. Do not use these elements when using the
construct namespace.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

Working with tables

To construct tables in XML, use a structure like the following:

```
<TABLE COLUMNS="2" ROWS="2">
   <ID NAME="MyTable"/>
   <GEOMETRY PAGE="1">
      <POSITION>
         <TOP>100</TOP>
         <LEFT>100</LEFT>
         <BOTTOM>600</BOTTOM>
         <RIGHT>400</RIGHT>
      </POSITION>
   </GEOMETRY>
   <COLSPEC>
      <COLUMN AUTOFIT="false" COLUMNCOUNT="1" COLUMNWIDTH="134.667">
         <GRIDLINE COLOR="Black" GAPCOLOR="none" OPACITY="100%" SHADE="100%"
STYLE="Solid" TYPE="LEFT" WIDTH="1"/>
         <GRIDLINE COLOR="Black" GAPCOLOR="none" OPACITY="100%" SHADE="100%"
STYLE="Solid" TYPE="RIGHT" WIDTH="1"/>
      </COLUMN>
      <COLUMN AUTOFIT="false" COLUMNCOUNT="2" COLUMNWIDTH="134.667">
         <GRIDLINE COLOR="Black" GAPCOLOR="none" OPACITY="100%" SHADE="100%"
STYLE="Solid" WIDTH="1"/>
      </COLUMN>
      <COLUMN AUTOFIT="false" COLUMNCOUNT="3" COLUMNWIDTH="134.667">
         <GRIDLINE COLOR="Black" GAPCOLOR="none" OPACITY="100%" SHADE="100%"
STYLE="Solid" WIDTH="1"/>
      </COLUMN>
   </COLSPEC>
   <ROW ROWCOUNT="1">
      <CELL COLUMNCOUNT ="1">
         ...
      </CELL>
      <CELL COLUMNCOUNT ="2">
         ...
      </CELL>
   </ROW>
</TABLE>
```

Note that the position of each row and column within the table is indicated by the ROWCOUNT and COLUMNCOUNT attributes, respectively.

<CELL> elements can describe text cells or picture cells; see the following sections for details.

Adding text cells

To add a text cell, use XML like the following:

```
<CELL BOXTYPE="CT_TEXT" COLUMNCOUNT ="1">
```

```
    <TEXT>
      <STORY>
        <RICHTEXT>Text goes here.</RICHTEXT>
      </STORY>
    </TEXT>
</CELL>
```

Note that the <TEXT> element must always contain a <STORY> element. A <STORY> element can contain <PARAGRAPH> elements or simply
<RICHTEXT> elements.

Adding picture cells

To add a picture cell, use XML like the following:

```
<CELL BOXTYPE="CT_PICT" COLUMNCOUNT ="1">
    <CONTENT>MacintoshHD:DocPool:flower1.jpg</CONTENT>
    <PICTURE FIT="CENTERPICTURE" />
</CELL>
```

Merge and split table cells

To merge table cells, use XML like the following:

```
<TABLE>
<ID NAME=”table1”/>
<ROW ROWCOUNT="1" MERGEROWSPAN=”1” >
<CELL COLCOUNT=”1”><TEXT>…</TEXT></CELL>
<CELL COLCOUNT=”2”> <TEXT>…</TEXT></CELL>
</ROW>
<ROW ROWCOUNT="2">
<CELL COLCOUNT=”1”> <TEXT>…</TEXT></CELL>
<CELL COLCOUNT=”2”> <TEXT>…</TEXT></CELL>
</ROW>
<ROW ROWCOUNT="3">
<CELL COLCOUNT=”1”> <TEXT>…</TEXT></CELL>
<CELL COLCOUNT=”2”> <TEXT>…</TEXT></CELL>
</ROW>
</TABLE>
```

To split table cells, use XML like the following:

```
<TABLE>
<ID NAME=”table1”/>
<ROW AUTOFIT="false" ROWCOUNT="5" ROWHEIGHT="60.9">
<CELL BOXTYPE="CT_TEXT" COLUMNCOUNT="2" SPLIT="true"/>
</ROW>
</TABLE>
```

Break a table across pages

To break a table across pages, use XML like the following:

```
<SPREAD>
<ID UID="1"/>
<PAGE MASTER="A-Master A" POSITION="RIGHTOFSPINE">
<ID UID="1"/>
```

```
</PAGE>
<TABLE COLOR="none" COLUMNS="2" MAINTAINGEOMETRY="false" ROWS="2"
AUTOFIT="rows">
<ID NAME="Table1"/>
<TABLEBREAK BREAKHEIGHT="140.251" KEEPATTRIBUTE="true"
MAINTAINLINK="true">
<HEADER>
<ROW ROWCOUNT="1" ROWHEIGHT="68.625">
...
</ROW>
</HEADER>
</TABLEBREAK>
<ROW ROWCOUNT="1" ROWHEIGHT="68.625">
...
</ROW>
<ROW ROWCOUNT="2" ROWHEIGHT="68.625">
...
</ROW>
<FRAME .../>
<GEOMETRY LAYER="Default" PAGE="1" SHAPE="SH_RECT">
...
</GEOMETRY>
<COLSPEC>
...
</COLSPEC></TABLE>
</SPREAD>
```

Specify table lines

To specify horizontal and vertical lines in a table, use XML like the following:

```
<TABLE>
    <GRID TYPE=”ALLGRID”>
<LINE COLOR="Black" GAPCOLOR="none"   OPACITY="100%" SHADE="100%"
STYLE="Solid" WIDTH="0"/>
     </GRID>
...
</TABLE>
```

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Deconstructing a project

Constructing a project

Working with Composition Zones

A Composition Zones item in a deconstructed project is represented in XML by a <COMPOSITIONZONE> element. Like the <BOX> element type, this element type supports the <GEOMETRY>, <SHADOW>, and <FRAME> elements. In addition, a <COMPOSITIONZONE> element includes a <TYPE> attribute that identifies it as an internal or external Composition Zones item. For external Composition Zones, the
PATH attribute indicates the location of the associated composition layout.
Composition Zones items must be created in QuarkXPress. <COMPOSITIONZONE> elements are ignored by the construct namespace and the
modify parameter.

```
<PROJECT>
  <LAYOUT>
    <ID UID="Layout 1"/>
    <SPREAD>
      <ID/>
      <COMPOSITIONZONE TYPE="INTERNAL
">
        <ID/>
      </COMPOSITIONZONE>
    </SPREAD>
  </LAYOUT>
</PROJECT>
```

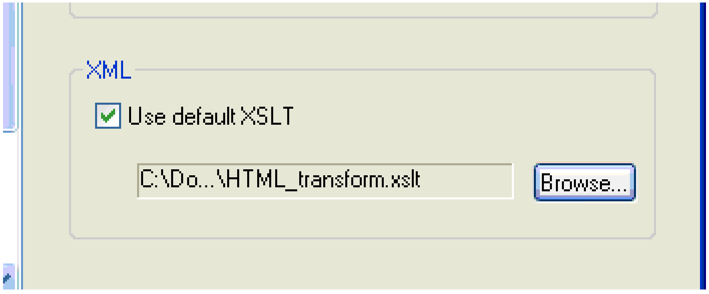
Related topics:

Modifier DTD
Modifier DTD (annotated)
construct
modify
xml

Using server XSLT

You can use an XSLT file to transform the XML returned by the xml namespace into other formats. You might find this feature useful if you want the xml namespace to return an XML representation that uses a different schema or a subset of the returned data.

There are two ways to using this feature. The first way is to select Use Default XSLT and specify the path to an XSLT file in the preferences for QuarkXPress Server (Server/QuarkXPress Server > Preferences > Modifier pane). If you choose this approach, the XSLT file is applied to all XML returned by the xml namespace.



☑ Use default XSLT

C:\Do...\HTML_transform.xslt

XML area of the Modifier pane of the Preferences dialog box

The second way to use XSL is to use the XSL parameter in the request URL. If the XSL parameter specifies the absolute path to an XSLT file on the server, QuarkXPress Server uses that XSLT file to transform the response to that call. For example: http://QXPServer8:8080/xml/project1.qxp?modify=file:path to XML file on server&XSL= path to XSLT file on server

To make returned XML use the Modifier DTD, uncheck Use default XSLT and do not use the XSL parameter in your calls to the construct namespace.

Note: QuarkXPress Server currently supports only XML output from XSL transformation.

Working with lists

The <LISTS> element allows you to construct and deconstruct QuarkXPress lists. Lists allow a user to automatically create a table of contents (TOC) or list of figures. For more information, see the Modifier DTD .

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PROJECT JOBJACKET="Project2 Job Jacket" JOBTICKET="Default Job Ticket 1:Project2"
 PROJECTNAME="lis1.qxp" XMLVERSION="8.0">
 <LAYOUT POINTSPERINCH="72">
  <ID NAME="Layout 1"/>
  <LAYER>
   <ID NAME="Default"/>
   <RGBCOLOR BLUE="90" GREEN="90" RED="90"/>
  </LAYER>
  <SPREAD>
   <ID UID="1"/>
   <PAGE MASTER="A-Master A" POSITION="RIGHTOFSPINE">
    <ID UID="1"/>
   </PAGE>
   <BOX BOXTYPE="CT_TEXT" COLOR="none">
    <ID NAME="Box5"/>
    <GEOMETRY>
     <POSITION>
      <TOP>56</TOP>
      <LEFT>56</LEFT>
      <BOTTOM>200</BOTTOM>
      <RIGHT>300</RIGHT>
     </POSITION>
    </GEOMETRY>
    <TEXT>
     <STORY>
      <LIST LISTSTYLE="New List" OPERATION="CREATE">
      </LIST>
     </STORY>
    </TEXT>
   </BOX>
  </SPREAD>
</LAYOUT>
</PROJECT>
```

LIST is a child of the STORY element. The value of LISTSTYLE will be the name of the list that had been created in QuarkXPress. When a project containing a list is deconstructed in XML, the XML will contain the text of the list, as well as a reference back to the LIST.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

Working with anchored boxes

To create an anchored box within a text box, use a structure like the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PROJECT JOBJACKET="Macintosh HD:Server:Project1 Job Jacket" JOBTICKET="Default Job
Ticket 1:Project2"
  PROJECTNAME="anchor.qxp" XMLVERSION="8.0">
  <LAYOUT POINTSPERINCH="72">
    <ID NAME="Layout 1"></ID>
    <LAYER>
      <ID NAME="Default"/>
      <RGBCOLOR BLUE="90" GREEN="90" RED="90"/>
    </LAYER>
    <SPREAD>
      <ID UID="1"/>
      <PAGE MASTER="A-Master A" POSITION="RIGHTOFSPINE">
        <ID UID="1"/>
      </PAGE>
      <BOX BOXTYPE="CT_TEXT" COLOR="none">
        <ID NAME="Box5"/>
        <GEOMETRY LAYER="Default" PAGE="1">
          <POSITION>
            <TOP>36</TOP>
            <LEFT>36</LEFT>
            <BOTTOM>112</BOTTOM>
            <RIGHT>210</RIGHT>
          </POSITION>
        </GEOMETRY>
        <TEXT>
          <STORY>
            <PARAGRAPH MERGE="false" PARASTYLE="Normal">
              <RICHTEXT MERGE="false">Hello </RICHTEXT>
              <ANCHOREDBOXREF ALIGNWITHTEXT="BASELINE"
OFFSET="0">Box7</ANCHOREDBOXREF>
              <RICHTEXT MERGE="false">, world</RICHTEXT>
            </PARAGRAPH>
          </STORY>
        </TEXT>
      </BOX>
      <BOX ANCHOREDIN="Box5" BOXTYPE="CT_TEXT" COLOR="none">
        <ID NAME="Box7" UID="7"/>
        <GEOMETRY PAGE="1" SHAPE="SH_RECT">
          <POSITION>
            <TOP>0</TOP>
            <LEFT>0</LEFT>
            <BOTTOM>50</BOTTOM>
```

```
                    <RIGHT>75</RIGHT>
                  </POSITION>
               </GEOMETRY>
               <TEXT>
                  <STORY>
                    <PARAGRAPH MERGE="false" PARASTYLE="Normal">
                       <RICHTEXT MERGE="false">anchored box</RICHTEXT>
                    </PARAGRAPH>
                  </STORY>
               </TEXT>
            </BOX>
         </SPREAD>
   </LAYOUT>
</PROJECT>
```

Note that there are two BOX elements. One is the parent box that has the element ANCHOREDBOXREF, which points to the name of the anchored box. The anchored box itself has the attribute
ANCHOREDIN, which points to the name of the parent box.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

Working with placeholders

Placeholders allow a region of text in a QuarkXPress project to hold non-printing metadata. You can use placeholders to store information from other systems, or to provide information to third-party XTensions software or other tools that operate on QuarkXPress projects.

Placeholders are used by technologies within QuarkXPress, such as XML import. Modifier XT allows placeholder data to be added to a QuarkXPress project from your application, and the placeholder data can be read from a project using the xml namespace.

Note: Unless a third-party XTensions software module for QuarkXPress is created to manage the placeholders inserted by your application using Modifier XML, a user is not prohibited from deleting placeholders from within the QuarkXPress user interface. In fact, users are not alerted to the presence of placeholders through the QuarkXPress user interface. You can use APIs in the QuarkXPress Server XTensions Software XDK to allow a suitable user interface for managing the placeholders inserted by your application. Contact QuarkAlliance for details about the XTensions software developer program.

There are two types of placeholders supported in Modifier XML: Text placeholders and Text Node placeholders. Text placeholders can be placed around a run of text to identify particular metadata with that text content.

```
<PROJECT>
  <LAYOUT>
    <ID UID="1"/>
    <SPREAD>
      <ID UID="1"/>
      <BOX>
        <ID NAME="name"/>
        <TEXT>
          <STORY CLEAROLDTEXT="true">
            <PARAGRAPH PARASTYLE="Normal"/>
            <RICHTEXT>This is text that</RICHTEXT>
            <TEXTPH NAME="SOURCE_UID" OWNER="1347639377">
              <RICHTEXT>has a placeholder</RICHTEXT>
            </TEXTPH>
          </STORY>
        </TEXT>
      </BOX>
    </SPREAD>
  </LAYOUT>
</PROJECT>
```

When a Text placeholder spans multiple paragraphs, the PARAGRAPH and RICHTEXT hierarchy is flattened. A new paragraph can be started using an empty PARAGRAPH element.

Text Node placeholders can represent a hierarchical structure of meta-tagging around text. This can allow more complex meta-tagging of data placed into a QuarkXPress project. Also, it allows some structure to be preserved within the QuarkXPress project format.

```
<PROJECT>
  <LAYOUT>
```

```
      <ID UID="1"/>
      <SPREAD>
        <ID UID="1"/>
         <BOX>
           <ID NAME="name"/>
            <TEXT>
              <STORY CLEAROLDTEXT="true">
                 <PARAGRAPH PARACHAR="HARDRETURN"/>
                 <TEXTNODEPH NAME="ARTICLE" OWNER="1347639377">
                    <TEXTPH NAME="HEADLINE">
                        <PARAGRAPH PARASTYLE="Headline"/>
                        <RICHTEXT>Quark Announces Availability of QuarkXPress Server 8.0
Updater</RICHTEXT>
                     </TEXTPH>
                    <TEXTPH NAME="STANDFIRST">
                        <PARAGRAPH PARACHAR="HARDRETURN" PARASTYLE="1st
para"/>
                        <RICHTEXT>Latest Maintenance Release of QuarkXPress 8.0 is Certified for
Windows Vista, Optimized for Mac OS X Leopard and Introduces Key Performance
Enhancements</RICHTEXT>
                     </TEXTPH>
                    <TEXTPH NAME="BODY">
                         <PARAGRAPH PARACHAR="HARDRETURN" PARASTYLE="Body"/>
                          <RICHTEXT>DENVER - 11-21-2007 - Quark Inc. today announced
availability of the QuarkXPress Server 8.0 updater, a maintenance release of the company’s
industry-leading professional design software.</RICHTEXT>
                    </TEXTPH>
                    <METADATA>
                      <VALUE KEY="ARTICLE_ID">1145</VALUE>
                      <VALUE KEY="ARTICLE_TYPE">Press Release</VALUE>
                      <VALUE KEY="AUTHOR">M.Gutherie</VALUE>
                    </METADATA>
                 </TEXTNODEPH>
              </STORY>
           </TEXT>
        </BOX>
      </SPREAD>
   </LAYOUT>
</PROJECT>
```

Note: To avoid hierarchy conflicts between the placeholder hierarchy and the paragraph hierarchy, the paragraph structure is flattened, which means that PARAGRAPH and RICHTEXT elements become siblings. In this case, the PARACHAR attribute is not applied, and the Modifier XML should include the
&hardReturn; entity to represent paragraph break characters.

Note: The OWNER attribute of the TEXTPH and TEXTNODEPH elements refers to the ID of the XTensions software that is responsible for the placeholder. The
xml namespace returns all placeholders from all XTensions software. The default value for placeholders is “1347639377” (this is the XTension ID of PlaceholderSXT XT). If you want to create placeholders for your own XTensions software, use that XTensions software ID here.

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Deconstructing a project

Constructing a project

Working with metadata

You can now attach box-level metadata to a QuarkXPress project created from XML using the Modifier DTD. For example, if you populate a picture box with a picture from a content management system, you can capture the unique ID of that picture (and other information, such as the last-modified date) to the box containing that picture. When you deconstruct such a project, you can read the metadata and use it in any way you like (such as, for example, for tracking the usage of licensed pictures).

Items to which metadata can be attached include picture boxes, text boxes, tables, lines, and text paths. QuarkXPress Server metadata takes the form of key/value pairs. For more information, see the Modifier DTD.

Constructing and modifying a project with metadata

To create a new box with metadata:

```
<BOX OPERATION="CREATE" BOXTYPE="CT_TEXT">
  <ID NAME="box1"/>
  <METADATA>
    <VALUE KEY=”Asset” ><![CDATA[1234567890]]>
    </VALUE>
    <VALUE KEY=”Date” ><![CDATA[08.06.07]]>
    </VALUE>
    <VALUE KEY=”Password” ><![CDATA[Hello World]]>
    </VALUE>
  </METADATA>
  <GEOMETRY SHAPE="SH_RECT">
    <POSITION>
      <TOP>5</TOP>
      <LEFT>5</LEFT>
      <BOTTOM>10</BOTTOM>
      <RIGHT>10</RIGHT>
    </POSITION>
  </GEOMETRY>
</BOX>
```

In this example, a box called ‘box1’ will be created and have the following metadata associated with it: Asset, 1234567890; Date, 08.06.07 and Password, Hello World.

Deleting metadata associated with a box

To delete metadata:

```
<BOX>
<ID NAME="BoxWithMetadata"/>
<METADATA>
<VALUE KEY=”Asset”></VALUE>
      </METADATA>
</BOX>
```

Related topics:

The Modifier DTD

About XML deconstruct and construct

xml

construct

Deconstructing a project

Constructing a project

Working with hidden text

Hidden text, represented in Modifier XML by the HIDDEN element, allows XTensions software developers to insert custom, non-printing data into a text flow. This hidden text can be retained when QuarkXPress Server deconstructs and reconstructs QuarkXPress projects. For more information, see the Modifier DTD .

```
<PARAGRAPH MERGE="false" PARACHAR="HARDRETURN" PARASTYLE="001-TEXT">
  <RICHTEXT MERGE="false">
     The population of Iceland is 500,000,000.
  </RICHTEXT>
  <HIDDEN DATALEN="100" OPCODE="51434410" OWNER="514344"
TYPE="CHARACTERTYPE">
    <RICHTEXT LANGUAGE="USEnglish" MERGE="false">
       VGhpcyBpcyB0aGUgdGV4dCBvZiBhIENvcHlEZXNrIG5vdGU=
     </RICHTEXT>
  </HIDDEN>
  <RICHTEXT MERGE="false">
     Iceland is located north of the Equator.
  </RICHTEXT>
</PARAGRAPH>
```

The example XML extract above shows the output from the xml namespace of text that contains a note inserted by the Notes XT XTensions software. Developed by Quark for QPS, the Notes XT XTensions software stores its data as hidden text. The note contains "This is the text of a CopyDesk note," which is represented as "VGhpcyBpcyB0aGUgdGV4dCBvZiBhIENvcHlEZXNrIG5vdGU=" in the sample above.

If this text is passed back to QuarkXPress Server in a Modify or Construct request, the hidden text inserted by the Notes XT XTensions software is preserved. Also, the hidden text can be read by the Notes XT XTensions software if the project is opened in QuarkXPress.

Notes for XTensions software developers

The data within the RICHTEXT element inside a HIDDEN element is a Base 64-encoded representation of the raw data that is stored within the hidden text. Considering that hidden text in QuarkXPress can contain any type of data, and the structure of that data is specified by the XTensions software that creates it, this method ensures that the data can be safely represented in XML. Also, this data can be converted back into the same raw data structure so that it can be read by the destination XTensions software. If the content is edited, the destination XTensions software may not be able to interpret it. Only XTensions software developers should attempt to interpret data from their own XTensions software.

For the <HIDDEN> element, the OPCODE attribute is a decimal representation of the XTension ID of the XTensions software that inserted this hidden text. The OWNER attribute is a decimal representation of the QuarkAlliance developer ID of the XTensions software developer who inserted this hidden text.

By default, hidden text is not output from the xml namespace. To output hidden text, specify the “opcode=” parameter in your request as follows:

http://server:port/xml/projectname.qxp?opcode=51433410

Note: You can also specify "...

opcode=*" to specify all hidden text in the XML output.
This example URL outputs all of the hidden text inserted by the XTensions software with this ID. To avoid byte order issues when cross-platform rendering is enabled, the XTID is represented decimally, rather than with the usual char[4] representation.

Related topics:

The Modifier DTD
About XML deconstruct and construct
xml
construct
Deconstructing a project
Constructing a project

About administrative request handlers

A request handler is the most powerful way to change the behavior of QuarkXPress Server. With this method, the QuarkXPress Server XTensions software can completely handle a subset of requests received by the server. A subset of requests is identified by a namespace string. Every request handler has a namespace string associated with it.

Namespace	getdocinfo, getdocpoollist, getprojinfo, fileinfo, addfile, delete, literal, flush, flushall, shutdown, getprefs, setprefs, and getserverinfo
Response	A QuarkXPress Server response, depending on the specific request handler.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/4/2004 10:10:43 - getprefs - Type: text/xml - Size: 2636 - Client: 127.0.0.1
Example GET URL	http://localhost:8080/getserverinfo
Notes	You can add a new request handler. During the DDSSETUPCBCODE callback, the QuarkXPress Server XTensions software registers itself as a request handler via AddCustomRequestHandler, the QuarkXPress Server XTensions API. The first parameter of this API is a pointer to a request handler function implemented in the QuarkXPress Server XTensions software. The second parameter is a namespace string that identifies the request. Whenever a user submits a request that has the same namespace string as a suffix to the request URL, the request handler function is called by QuarkXPress Server with all the user-specified parameters filled in the ServerRequest structure. The request handler function then processes the request and submits the reply in the ServerReply structure, which QuarkXPress Server communicates back to the user-agent.

addfile

Use the addfile request handler to place a document or image file in the document pool. The addfile request is always a POST request. It cannot be a GET request because binary content is attached with the request.

If you send an addfile request to Server Manager, using HTTP or the Web services interface, the file is uploaded to all registered QuarkXPress servers if the common doc pool switch is set to off in the admin client. If the common doc pool is been enabled in the admin client, the file is uploaded to any one registered QuarkXPress server.

Namespace	addfile		
Parameters	uploadfile	Binary file or MIME-type file	Contains the actual binary content of the QuarkXPress file, a Word file, a text file, or MIME-types files, such as EPS, JPEG, PNG, and PICT.
Response	QuarkXPress Server responds with the message "File upload completed."		
Alerts	The file system document pool is not enabled.	HTTP Error #404 This alert is displayed when you attempt to upload a document but the file system document pool is not enabled in the **Server Configuration** dialog box. *What to do*: Check **Enable File System Document Pool** in the **Server Configuration** dialog box. Click **OK** and resubmit the *addfile* request.	
	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when you enter an invalid administrator user name and password. *What to do*: Identify the correct user name and password set in the **Server Configuration** dialog box, and then resubmit the *addfile* request with the correct user name and password.	
	Cannot find required volume or folder	HTTP Error #500 QuarkXPress Server Error #120 This alert is displayed when you attempt to upload a document that exists in a subfolder and this subfolder does not exist in the document pool. The *addfile* request does not create a subfolder in the document pool if **Generate Hierarchy on Document Upload** is unchecked in the **Server Configuration** dialog box. *What to do*: Check **Generate Hierarchy on Document Upload** in the **Server Configuration** dialog box. Click **OK** and resubmit the addfile request.	
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, document name, type of response produced by server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2004 19:24:12 - addfile/p1.qxp - Type: text/html - Size: 22 - Client: 127.0.0.1		

	If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 20:08:45 - Error - Error Code: 10100 - The file system document pool is not enabled.
Example GET URL	To post a binary file in the root folder http://localhost:8080/addfile/abc.qxp To post a binary file in a subfolder http://localhost:8080/addfile/sub1/abc.qxp

Example, Object Model	Request Object Name : AddFileRequest
	//STEP 1 (COMMON FOR ALL REQUESTS):
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	Stream theStream = uplTheFile.PostedFile.InputStream;
	long length = theStream.Length;
	Byte[] Buffer = new Byte[length];
	const int BUFFER_SIZE = 10000;
	int nBytesRead = 0,iCount = 0;
	long remainingBytes = length - BUFFER_SIZE;
	if(remainingBytes > BUFFER_SIZE)
	{
	nBytesRead = theStream.Read(Buffer,iCount * BUFFER_SIZE,BUFFER_SIZE);
	while(0 != nBytesRead)
	{
	iCount++;
	remainingBytes = length - (iCount * BUFFER_SIZE);
	if(remainingBytes > BUFFER_SIZE)
	nBytesRead = theStream.Read(Buffer,iCount * BUFFER_SIZE,BUFFER_SIZE);
	else
	{
	nBytesRead = theStream.Read(Buffer,iCount * BUFFER_SIZE,(int)remainingBytes);
	break;
	}
	}
	}
	else
	nBytesRead = theStream.Read(Buffer,iCount * BUFFER_SIZE,(int)remainingBytes);
	AddFileRequest addfilereq = new AddFileRequest();
	addfilereq.fileData = Buffer;
	rc.request = addfilereq;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	QContentData qc = svc.processRequest(rc);
	Note: The object model uses SOAP to transfer data. SOAP encoding is not the most efficient way to transfer binary data. If you have to add a file using

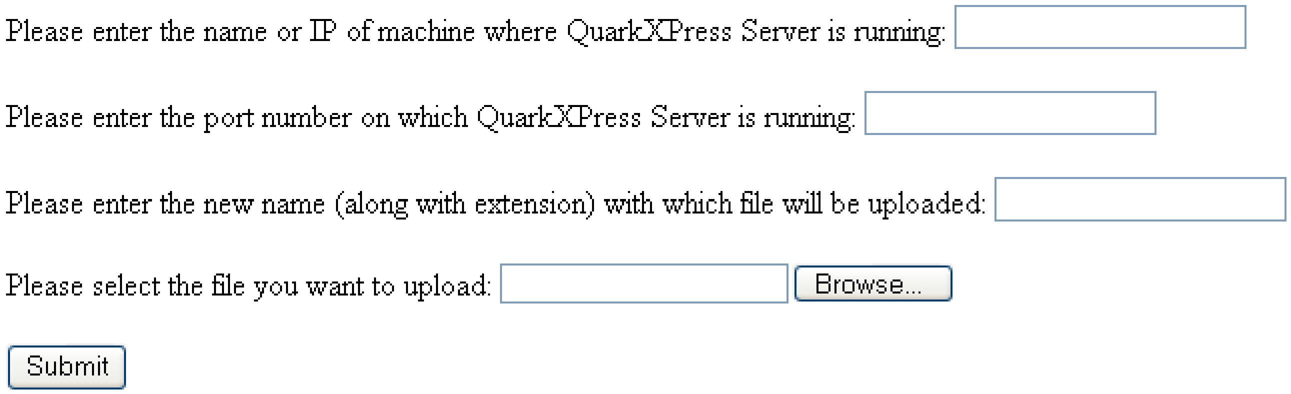
	QuarkXPress Server Manager, the best way is to use a POST request in a QuarkXPress Server Manager URL. You might use QuarkXPress Manager to add a file if you wanted to add the file to all registered QuarkXPress Server instances at one time (assuming the instances are not sharing a single document pool).
Notes	This cannot be a GET request since binary content is attached. Therefore, the request has to be a POST request. The following is a sample of a post request HTML form. <HTML> <HEAD><TITLE>Test Addfile</TITLE></HEAD> <BODY> File will always be uploaded with name new.qxp <FORM ACTION="http://localhost:8080/addfile/new.qxp" METHOD = "post" ENCTYPE="multipart/form-data"> Please select the file you want to upload: <INPUT TYPE=file NAME="uploadFile"><br> <br><INPUT TYPE=submit VALUE="Submit"> </FORM> </BODY> </HTML>

Example

To view the HTML, click here

.

The following is a sample POST request using the addfile request handler:



Please enter the name or IP of machine where QuarkXPress Server is running:

Please enter the port number on which QuarkXPress Server is running:

Please enter the new name (along with extension) with which file will be uploaded:

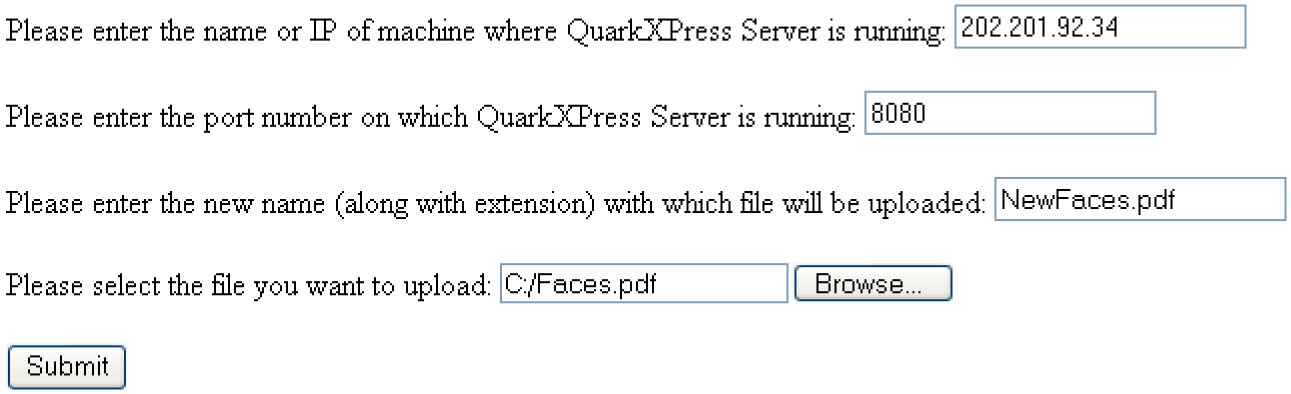
Please select the file you want to upload:

Using This Form

1.
1. Enter the name or IP address of the computer on which QuarkXPress Server is running.
2.
2. Enter the port number in the port number field.
3.
3. Enter the file name along with the extension in the file field. The file will be uploaded with this name. Click
3. Browse if you need to find the file on your computer.
4.
4. Click
4. Submit.

The file uploads to the document pool of the specified server. After the file is successfully uploaded, the "File upload completed." alert is displayed.

For example, suppose you uploaded the file Faces.pdf located on the C: drive of Windows to QuarkXPress Server running at IP 202.201.92.34 and port 8080, and you uploaded it with the name "NewFaces.pdf." The above HTML form would look like this:



Please enter the name or IP of machine where QuarkXPress Server is running: 202.201.92.34

Please enter the port number on which QuarkXPress Server is running: 8080

Please enter the new name (along with extension) with which file will be uploaded: NewFaces.pdf

Please select the file you want to upload: C:/Faces.pdf

The HTML code to generate the above sample file is:

```
<HTML>
<HEAD>
 <TITLE>Test Addfile</TITLE>
 <SCRIPT LANGUAGE="JavaScript">
 function UploadDocument() {
  var URL;
  URL = "http://" + UploadForm.MachineIP.value + ":"  +
   UploadForm.Port.value + "/addfile/" + UploadForm.NewName.value;
  UploadForm.action = URL;
 }
 </SCRIPT>
</HEAD>
<BODY>
 <FORM ID="UploadForm" METHOD = "post" ENCTYPE="multipart/form-data"
onSubmit="UploadDocument()">
 Please enter the name or IP of machine where QuarkXPress Server is running:
  <INPUT TYPE="TextBox" NAME="MachineIP"><br><br>
 Please enter the port number on which QuarkXPress Server is running:
  <INPUT TYPE="TextBox" NAME="Port"><br><br>
 Please enter the new name (along with extension) with which file will be uploaded:
  <INPUT TYPE="TextBox" NAME="NewName"><br><br>
 Please select the file you want to upload: <INPUT TYPE=file NAME="uploadFile"><br><br>
 <INPUT TYPE=submit VALUE="Submit">
</FORM>
</BODY>
</HTML>
```

The information entered in the form is created with the following tags:

```
<FORM ID="UploadForm" METHOD = "post" ENCTYPE="multipart/form-data"
onSubmit="UploadDocument()">
 Please enter the name or IP of machine where QuarkXPress Server is running:
 <INPUT TYPE="TextBox" NAME="MachineIP"><br><br>
 Please enter the port number on which QuarkXPress Server is running:
 <INPUT TYPE="TextBox" NAME="Port"><br><br>
 Please enter the new name (along with extension) with which file will be uploaded:
  <INPUT TYPE="TextBox" NAME="NewName"><br><br>
  Please select the file you want to upload:
  <INPUT TYPE=file NAME="uploadFile"><br><br>
  <INPUT TYPE=submit VALUE="Submit">
</FORM>
```

The FORM tag specifies that the method of the request is POST. This request is a "Multipart/form-data" request. When you submit the form, the
UploadDocument() function is called.
Use the INPUT tag to create the text box and the
Browse button.

-
- <INPUT TYPE="TextBox": To create text boxes only.
-
- <INPUT TYPE=file: To create a combination of text box and the Browse button in the form. When you click Browse and choose any file, the file path of the selected file displays in the text box linked with the
- Browse button.

You can use the INPUT tag to create the Submit button. <INPUT TYPE=submit VALUE="Submit"> creates the
Submit button on the form.
When you click Submit, the UploadDocument() function is called. This function is defined inside a script tag. This function combines the information entered in the form to create a URL for the addfile request and sends this URL to QuarkXPress Server for processing. The code of the
UploadDocument() function is as follows:

```
<SCRIPT LANGUAGE="JavaScript">
  function UploadDocument() {
    var URL;
    URL = "http://" + UploadForm.MachineIP.value + ":"
      +  UploadForm.Port.value + "/addfile/" + UploadForm.NewName.value;
    UploadForm.action = URL;
  }
  </SCRIPT>
```

Declare a variable URL. This variable combines the information entered in the form to create an addfile request. The UploadForm.MachineIP.value statement retrieves the IP address or the name of your computer. The other information is also retrieved and combined. Finally, the statement UploadForm.action = URL; sends the URL to QuarkXPress Server for processing.

cplatform

Used to handle Unicode language support in QuarkXPress Server. It tells the server that the client browser/machine is running on a specified platform (Windows or Mac OS).

Note: In QuarkXPress Server Manager, this parameter is deprecated and its use is strongly discouraged. Please use UTF-8 to post data to Server Manager to avoid encoding issues.

Parameters	cplatform	string	Tells the server the client platform on which on the request was generated. The value for Windows is *win*. The value for Mac OS is *mac*.
Response	Preview of the document.		
Alerts			
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, document name, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 12/2/2005 13:50:49 - project1.qxp - Type: image/jpeg - Size: 10766 - Client: 127.0.0.1 If an error occurs, the error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 12/2/2005 11:32:32 - Error - Error Code: -43 - File not found.		
Example GET URL	http://localhost:8080/sample.qxp?Story=f?ßßµ?&clang=EL&cplatform=win where some Greek characters are flowed into the text box named *Story* and the request is given from the Windows platform.		

Example, Object Model	Request Object Names : RequestSettings
	sdk.QRequestContext rc = new sdk.QRequestContext();
	    if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	 //STEP 2(SPECIFIC TO REQUESTS):Create the RequestSettings renderer request and embed it in request context
	RequestSettings requestSetting = new RequestSettings();
	requestSetting.clientPlatform = clientplatformValue;
	rc.request = requestSetting;
	 //Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);		

Notes	*What languages are supported with the Unicode language feature of QuarkXPress Server?* The Unicode languages supported by QuarkXPress Server are Finnish, Portuguese, Brazilian Portuguese, Spanish, Slovakian, Hungarian, Polish, Czech, Greek, Russian, Turkish, and Romanian.		

delete

Removes a specified document from the document pool. The *Delete* request can also delete folders.

If this request is sent to Server Manager using either HTTP or Web services, the specified file is deleted from all registered QuarkXPress servers if the common doc pool has been switched off in the admin client. If the common doc pool has been turned on in the admin client, the delete request is sent to any one registered QuarkXPress server.

Namespace	delete	
Response	QuarkXPress Server responds with the message "File deleted successfully."	
Alerts	File not found	HTTP Error #404 QuarkXPress Server Error #-43 This alert is displayed when you try to delete a file that does not exist in the document pool. *What to do*: Specify a valid file name in the delete request.
	Folder cannot be deleted. It may still contain files.	HTTP Error #405 This alert is displayed when you try to delete a folder that is not empty. *What to do*: You cannot delete a folder that is not empty. First, delete all the files in the folder, and then resubmit the delete request to delete the folder.
	I/O error trying to read or write to disk.	HTTP Error #500 QuarkXPress Server Error #-36 This alert is displayed when you try to delete an open file. *What to do*: Close the opened file and resubmit the delete request.
	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when you provide an invalid administrator user name and password. *What to do*: Find out the correct user name and password that were set in the server configuration and then resubmit the delete request with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 20:37:57 - delete/2000.qxp - Type: text/html - Size: 26 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2004 21:49:13 - Error - Error Code: 10098 - Folder cannot be deleted. It may still contain files.	
Example GET URL	http://localhost:8080/delete/sample.qxp	

Example, Object Model	Request Object Name : DeleteRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new DeleteRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • You cannot delete a folder that is not empty. First, delete all the files in the folder, and then resubmit the delete request. • • The *Delete* parameter request requires an administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *delete* request is submitted to the browser, it asks for the user name and password. Enter the same user name and password as set in the **Server Configuration** dialog box and click **OK** • .

clang

Handles Unicode® language support in QuarkXPress Server. It tells the server that the client browser/machine is running in the specified (Unicode) language. It also specifies that the values given in the HTTP request are in the specified (Unicode) language.

Note: In QuarkXPress Server Manager, this parameter is deprecated and its use is strongly discouraged. Please use UTF-8 to post data to Server Manager to avoid encoding issues.

Parameters	clang	string	Specifies that values given in the HTTP request are in the specified (Unicode) language. Parameter values: CS Czech EL Greek ES Spanish FI Finnish HU Hungarian PL Polish PT Portuguese PT-BR Brazilian Portuguese RO Romanian RU Russian SK Slovakian TR Turkish
Response	Preview of document		
Alerts	The content language/script is unknown. Please provide the content language parameter to be served by your QuarkXPress Server.	HTTP Error #500 This alert is displayed when an invalid value is given with the *clang* parameter. *What to do*: Provide a valid value with the *clang* parameter.	
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 12/1/2005 14:57:37 - p1.qxp - Type: image/jpeg - Size: 11981 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server Error Log. The transaction entry in the error log contains the date and time of the request, the error code, and the error message. The following is a sample of an error log transaction entry: 12/1/2005 14:56:21 - Error - Error Code: 10136 - The content language/script is unknown. Please provide the content language parameter to be served by your QuarkXPress Server.		

Example GET URL	http://localhost:8080/sample.qxp?Story=abcdefghijklmnopqrstuvwxyz&clang=EL where some Greek characters are flown into the text box named *Story*.

Example, Object Model	Request Object Names : RequestSettings
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	//STEP 2(SPECIFIC TO REQUESTS):Create the RequestSettings renderer request and embed it in request context
	RequestSettings requestSetting = new RequestSettings();
	requestSetting.clientLanguage = clientLanguageValue;
	rc.request = requestSetting;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	• • *What languages are supported by the Unicode language feature of QuarkXPress Server?* • The Unicode languages supported by QuarkXPress Server are Finnish, Portuguese, Brazilian Portuguese, Spanish, Slovakian, Hungarian, Polish, Czech, Greek, Russian, Turkish, and Romanian. The Unicode character set is supported. • • *Does the user interface change when QuarkXPress Server is launched in a Unicode language OS?* • There is no change in the user interface of QuarkXPress Server when it is launched in a Unicode language. The user interface of QuarkXPress Server remains in English. • • *What functionality is supported in the Unicode language feature?* • You can specify content in text boxes and the document pool path in the Unicode language. You can also import Unicode text in a text box. You can specify Unicode text inside the TextModifier, DataImport, and Modifier XML, and you can even provide file paths and box names in the Unicode language in the following Server XTensions XML files: ModifierXT and XML Import. • • *What happens if a document is created with Unicode text on one platform and the document is uploaded to another platform? For example, suppose the document is created on a Mac OS computer and is uploaded to Windows.* • When a document is uploaded to another platform, some Unicode text may get changed and may not appear the same. • • *In what encoding format should I save a text file or XML file if it contains Unicode text?* • A file containing Unicode text must be saved in UTF-8 or UTF-16 format. • • *I have flowed some Unicode language text (using the BoxParam XTensions) in a text box. But the text in the document looks different*

from the text specified in the request. What could be the problem?

- This may be due to the font applied on the text box. If you create a request to import or flow Unicode text in a text box, and the font for the specific Unicode language is not applied to it, then text flowed in the text box may appear different than what was specified in the request. To get correct output, replace the font of the text box with the font for the specific Unicode language. For example, suppose a document is created with a text box to which Arial font is applied. Upload or Save the document in the document pool. Now create a request to flow Greek or Russian text in this text box. The preview generated from QuarkXPress Server will not show the same text in the text box as was specified in the request. To obtain the correct output, change the font of text box to Arial Greek (for Greek) or Arial CYR (for Russian) in the document. Save the document and upload the document to the server. Create a request to flow the Unicode text in the text box. Correct characters will now be displayed in the document.
- This problem can also occur if the *font fallback*
- feature is turned OFF in QuarkXPress Server. If this feature is turned ON, then QuarkXPress Server automatically applies the correct fonts, depending on the script being used. This feature is turned ON by default.
-
- *How many render types work with the clang parameter?*
- clang works for all render types supported in QuarkXPress Server.
-
- How can else can Unicode data be sent to the server? You can also send Unicode data to the server in the following ways, which do not require clang and cplatform to be in the URL:
  -
  - "URL encoded UTF-8" string in the URL
  -
  - UTF-8 encoded XML files (in construct/modify)
  -
  - UTF-8 text file in data import
  -
  - Word, RTF files

fileinfo

Returns the creation date, modification date, and file size of the specified document in XML format.

Namespace	fileinfo	
Response	The following XML code displays the creation date, modification date, and size of the document. <?xml version="1.0" encoding="UTF-8" ?> <FILEINFO> <CREATIONDATE>08-01-2004 06:14:07 UTC </CREATIONDATE> <MODIFICATIONDATE>08-01-2004 11:56:56 UTC </MODIFICATIONDATE> <SIZE>1519616</SIZE> </FILEINFO>	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. *What to do*: Find out the correct user name and password set in the **Server Configuration** dialog box, and then resubmit the *fileinfo* request with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 18:26:48 - fileinfo/Brochure_Base.qxd - Type: text/xml - Size: 191 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/fileinfo/sample.qxp	

Example, Object Model	Request Object Name : FileInfoRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	        if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	 rc.request = new FileInfoRequest();
	 //Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	The *fileinfo* parameter request requires an administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *fileinfo* request is submitted to the browser, it asks for the user name and password. Enter the same user name and password that were set in the **Server Configuration** dialog box and click **OK**.	

flush

Flushes a document from the cache.

Namespace	flush	
Response	QuarkXPress Server responds with the message "CACHE FLUSH COMPLETED."	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when the wrong administrator user name and password are specified. *What to do*: Identify the correct user name and password that were set in the **Server Configuration** dialog box, and then resubmit the *Flush* request handler with the correct user name and password.
Logs	If the document is cleared from the cache, a transaction success message is written in the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 11/30/2005 17:32:45 - flush/project1 - Type: text/html - Size: 21 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/flush/sample.qxp	

Example, Object Model	Request Object Name : FlushRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	       if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	 rc.request = new FlushRequest();
	 //Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	*What happens if user verification is set to "On" for administrator requests?* The *flush* request requires the administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *flush* request is submitted to the browser, it asks for a user name and password. Enter the user name and password that were set in the **Server Configuration** dialog box and click **OK**.	

flushall

Flushes all documents from the cache.

When this request is sent to Server Manager using either HTTP or Web services, the cache of all registered QuarkXPress servers is flushed.

Namespace	flushall	
Response	QuarkXPress Server responds with the message "CACHE FLUSH COMPLETED"	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when the wrong administrator user name and password are specified. *What to do*: Identify the correct user name and password that were set in the **Server Configuration** dialog box, and then resubmit the Flushall request handler with the correct user name and password.
Logs	If the cache is successfully flushed, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 11/30/2005 17:37:46 - flushall - Type: text/html - Size: 21 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password	
Example GET URL	http://localhost:8080/flushall	

Example, Object Model	Request Object Name : FlushAllRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new FlushAllRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	*What happens if user verification is set to "On" for an administrator request?* The *Flushall* request requires the administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *Flushall* request is submitted to the browser, it asks for a user name and password. Enter the same user name and password that were set in the **Server Configuration** dialog box and click **OK**. *Will the **Memory Usage** field in the status monitor field change when the Flushall request is issued to QuarkXPress Server?*	

	The value of memory usage becomes zero in the status monitor when you issue the *Flushall* request.

getdocinfo

Returns an information block on a specific QuarkXPress project loaded into the QuarkXPress Server document pool. The information is returned in XML format and contains the project version, platform, layers, page properties, the length and width of the project page expressed in points, the number of pages in the project, a list of linked high-resolution graphics, the names of graphic images that are used in the QuarkXPress project or template, any required fonts, any required server XTensions modules, and the relevant XTensions software or server XTensions module IDs. The handler displays synchronized text only in the case of QuarkXPress 6.0 projects.

Namespace	getdocinfo	
Response	The following XML code displays detailed document information regarding version, platform, layers, page properties, length and width of document pages, number of pages, and the names of graphic images used in the document. <? xml version="1.0" encoding="UTF-8" ?> <PROJINFO> <PLATFORM>WINDOWS</PLATFORM> <VERSION>7.0</VERSION> <NAME>Sample.qxp</NAME> <REQUIREDXTENSIONS /> <FONTUSAGE> <FONT> <NAME>ArialMT</NAME> </FONT> </FONTUSAGE> <LAYOUT> <NAME>Layout 1</NAME> <TYPE>Print</TYPE> <PAGES>4</PAGES> <PAGEPROPERTIES> <WIDTH>432</WIDTH> <LENGTH>756</LENGTH> </PAGEPROPERTIES> <LAYERS> <LAYER> <NAME>Default</NAME> </LAYER> </LAYERS> <HIRESGRAPHICS> <GRAPHICLINK> <FILEPATH>E:\pics\Jpeg\Autumn.jpg</FILEPATH> <USAGE PAGE="1" UNIQUEID="8" X="126.003" Y="116.967" /> </GRAPHICLINK> </HIRESGRAPHICS> </LAYOUT> </PROJINFO>	
Alerts	Incorrect administration realm	HTTP Error #401 This alert is displayed when an invalid administrator user name

	user name and password.	and password are specified. *What to do*: Identify the correct username and password that were set in the **Server Configuration** dialog box, and then resubmit the g*etdocinfo* request with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 17:37:56 - getdocinfo/sample.qxp - Type: text/xml - Size: 590 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/getdocinfo/sample.qxp	

Example, Object Model	Request Object Name : GetDocInfoRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new GetDocInfoRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	The *getdocinfo* parameter request requires an administrator user name and password if those were set in the **Server Configuration** dialog box. When the *getdocinfo* request is submitted to the browser, it asks for the user name and password. Enter the same user name and password that were set in the **Server Configuration** dialog box and click **OK**.	

getprefs

Returns the current preference settings of the server in XML format.

Namespace	getprefs	
Response	XML display of server preference settings	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. *What to do*: Identify the correct user name and password that were set in the **Server Configuration** dialog box and then resubmit *getprefs* with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/4/2004 10:10:43 - getprefs - Type: text/xml - Size: 2636 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/getprefs	

Example, Object Model	Request Object Name : GetPreferencesRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	      if(!this.DocumentSettings1.documentName.Text.Equals(""))
	 rc.documentName = this.DocumentSettings1.documentName.Text;
	 rc.request = new GetPreferencesRequest();
	 //Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	• • The *getprefs* • parameter returns preference settings regarding Server Configuration and Status Monitor. It does not return preference other settings, such as Deconstruct and PDF Workflow settings. • • The *getprefs* parameter request requires an administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *Getprefs* request is submitted to the browser, it asks for a user name and password. Enter the same user name and password specified in the **Server Configuration** dialog box and click **OK** • .	

getprojinfo

Returns information about a specific QuarkXPress project loaded in the QuarkXPress Server document pool. The information is returned in XML format and identifies the operating system, the QuarkXPress version in which the project was created, the size of the project, the page properties of layouts, and information about named boxes and synchronized text.

Namespace	getprojinfo	

Response	The following XML code displays detailed document information about version, platform, page properties, length and width of pages, and number of pages. It also displays named boxes and synchronized text of the document.
	<? xml version="1.0" encoding="UTF-8" ?>
	<PROJINFO>
	  <PLATFORM>WINDOWS</PLATFORM>
	  <VERSION>6.0</VERSION>
	  <NAME>Sample.qxp</NAME>
	  <SIZE>1519616 Bytes</SIZE>
	  <SYNCHRONIZED/>
	  <LAYOUT>
	    <NAME>Layout 1</NAME>
	    <TYPE>Print</TYPE>
	    <PAGES>4</PAGES>
	    <PAGEPROPERTIES>
	      <WIDTH>432</WIDTH>
	      <LENGTH>756</LENGTH>
	    </PAGEPROPERTIES>
	  <NAMEDBOX>
	    <BOX>box2</BOX>
	    <BOX>box1</BOX>
	  </NAMEDBOX>
	  </LAYOUT>
	</PROJINFO>	

Alerts	The *getprojinfo* command can only be used for QuarkXPress 6 documents and later.	HTTP Error #500 This alert is displayed when the *getprojinfo* parameter requests a QuarkXPress 4.0 or 5.0 document. *What to do*: You cannot use the *getprojinfo* parameter with QuarkXPress 4.0 or 5.0 documents.
	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. *What to do*: Enter the correct user name and password that were set in the server configuration and then resubmit the *getprojinfo* request with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, document name, type of response	

	produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/3/2005 17:38:03 - getprojinfo/sample.qxp - Type: text/xml - Size: 386 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:45:02 - Error - Error Code: 10124 - The *getprojinfo* command can only be used for QuarkXPress 6 documents and later.
Example GET URL	http://localhost:8080/getprojinfo/sample.qxp

Example, Object Model	Request Object Name : GetProjectInfoRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new GetProjectInfoRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	QContentData qc = svc.processRequest(rc);

Notes	• • The *getprojinfo* • parameter only works with projects created in QuarkXPress 6.0. It does not work with documents created in QuarkXPress 5.0. • • The *getprojinfo* request requires an administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *getprojinfo* request is submitted to the browser, it asks for a user name and password. Enter the same user name and password specified in the **Server Configuration** dialog box and click **OK** • .

getserverinfo

Returns an information block about QuarkXPress Server. The information is returned in XML format and contains the platform on which the server is running, the version of QuarkXPress Server, a list of installed fonts and server XTensions modules, the relevant XTensions software or server XTensions module IDs, and the startup parameters with which the server is running. Any disabled server XTensions modules are not returned in the response.

Namespace	getserverinfo	
Response	XML containing list of server XTensions, required components, list of fonts available to server, and startup parameters.	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. What to do: Identify the correct user name and password that were set in the **Server Configuration** dialog box, and then resubmit the getserverinfo request with the correct user name and password.
Logs	If the document is successfully rendered, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/4/2005 10:34:48 - getserverinfo - Type: text/xml - Size: 62098 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/getserverinfo	

Example, Object Model	Request Object Name : GetServerInfoRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	rc.request = new GetServerInfoRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);	

Notes	The *getserverinfo* parameter request requires an administrator user name and password if those were set in the **Server Configuration** dialog box. When the *getserverinfo* request is submitted to the browser, it asks for a user name and password. Enter the user name and password that were set in the **Server Configuration** dialog box and click **OK**.	

setprefs

Sets server preferences.

Namespace	setprefs
Response	QuarkXPress Server responds with the message "Preferences successfully set."
Alerts	Incorrect administration realm user name and password. HTTP Error #401 This alert is displayed when the wrong administrator user name and password are specified. *What to do*: Identify the correct user name and password that were set in the server configuration, and then resubmit *setprefs* with the correct user name and password.
Logs	If the request is successfully processed, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, type of response produced by the server, size of the response returned in bytes, and client IP address. The following is a sample of a transaction entry: 1/13/2006 16:04:28 - setprefs - Type: text/plain - Size: 29 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.
Example GET URL	http://localhost:8080/setprefs?CacheSize=200

Example, Object Model	Request Object Name : GetPreferencesRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	if(!this.DocumentSettings1.documentName.Text.Equals(""))
	rc.documentName = this.DocumentSettings1.documentName.Text;
	NameValueParam nameValueParam = new NameValueParam();
	nameValueParam.paramName = this.pref1.Text;
	nameValueParam.textValue = this.prefvalue1.Text;
	SetPreferencesRequest request = new SetPreferencesRequest();
	request.serverpreferences = new NameValueParam[]{nameValueParam};
	rc.request = request;
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc);

Notes	*How can I set a preference with setprefs?* First issue the *getprefs* request handler. *gstprefs* returns the server preferences in the form of XML. Note the XML tags of the server preferences you want to modify. Submit the *setprefs* request for the preferences you want to modify. For example, suppose you want to turn off Memory Caching. First submit a *getprefs* request to the server. In the XML obtained by *getprefs*, note that the memory caching tag is "AllowMemoryCaching". Now submit this *setprefs* request to the server: http://localhost:8080/setprefs?AllowMemoryCaching=false

	The *setprefs* request requires an administrator user name and password if those were set in the **Server Configuration** dialog box. When the s*etprefs* request is submitted to the browser, it asks for a user name and password. Enter the same user name and password that were set in the **Server Configuration** dialog box, and click **OK**. *What happens if user verification is set on for administrator requests?* The *getprefs* request requires an administrator user name and password if those were set in the **Server Configuration** dialog box. When a *getprefs* request is submitted to the browser, it will ask for a user name and password. Give the same user name and password that were set in the **Server Configuration** dialog box, and click **OK**.

shutdown

Shuts down QuarkXPress Server.

When this request is sent to Server Manager using either HTTP or Web services, a shutdown request is sent to all registered QuarkXPress servers.

Namespace	shutdown	
Response	QuarkXPress Server responds with the message "Shutdown request posted." QuarkXPress Server then stops accepting new requests and shuts down after completing all pending transactions.	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when the wrong administrator user name and password are specified. *What to do*: Identify the correct user name and password that were set in the server configuration, and then resubmit the *Shutdown* request handler with the correct user name and password.
Logs	If the request is successfully processed, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, request type, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 11/30/2005 17:40:16 - shutdown - Type: text/html - Size: 24 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/shutdown	

Example, Object Model	Request Object Name : ShutdownRequest
	sdk.QRequestContext rc = new sdk.QRequestContext();
	        if(!this.DocumentSettings1.documentName.Text.Equals(""))
	  rc.documentName = this.DocumentSettings1.documentName.Text;
	 rc.request = new ShutdownRequest();
	//Create the service and call it with QRequestContext object
	QManagerSDKSvcService svc = new QManagerSDKSvcService();
	sdk.QContentData qc = svc.processRequest(rc)	

Notes	*What happens if user verification is set on for an administrator request?* The *Shutdown* request requires an administrator user name and password if the user name and password were set in the **Server Configuration** dialog box. When the *Shutdown* request is submitted to the browser, it asks for a user name and password. Enter the same user name and password specified in the **Server Configuration** dialog box and click **OK** . When issuing a shutdown request through Web services, the user name and	

	password to be used for realm verification should be specified using the "setUserName" and "setUserPassword" methods of the QContextRequest object that contains this ShutdownRequest object.

getdocpoollist

Returns an XML representation of all files names and folder names, the document type, size of the document, the modified date and time, and the absolute and relative path of the documents in the local document pool.

Namespace	getdocpoollist	
Response	XML file containing folder name, absolute and relative file path, file type, modified dates, and file size.	
Parameters	directory	You can get the list of files in a specific diretory in the document pool. For example: http://server:port/getdocpoollist?directory=images
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert is displayed when an invalid administrator user name and password are specified. What to do: Identify the correct user name and password that were set in the **Server Configuration** dialog box, and then resubmit the Getdocpoollist request with the correct user name and password.
Logs	If the request is successfully returned, a transaction success message is written to the QuarkXPress Server Transaction Log file. The transaction entry consists of the date and time of the request, render type, document name, type of response produced by the server, size of response returned in bytes, and client IP address. The following is a sample of a transaction entry: 8/4/2005 10:34:48 - getdocpoollist - Type: text/xml - Size: 62098 - Client: 127.0.0.1 If an alert is displayed, an error message is written to the QuarkXPress Server error log file. The following is a sample of an error log entry: 8/3/2005 17:49:23 - Error - Error Code: 10022 - Incorrect administration realm user name and password.	
Example GET URL	http://localhost:8080/getdocpoollist	
Example, Object Model	Request Object Name: GetDocPoolListRequest	
Notes	The getdocpoollist parameter request requires an administrator user name and password if they were specified in the Server Configuration dialog box. When the getserverinfo request is submitted to the browser, it asks for a user name and password. Enter the user name and password that were set in the Server Configuration dialog box and click OK.	

Modifier DTD (annotated)

This section provides the annotated version of the Modifier DTD. Details are provided for how to form XML code that uses the construct namespace, modify parameter, and xml namespace. The XML sent to or from these functions is case-sensitive and validated by the Modifier DTD, thereby providing well-formed XML code that is compatible between each function.

Note: Measurement values do not require units. For example, "25pt" should be submitted as "25".

Modifier DTD	**Construct** (construction of a QuarkXPress project using the construct namespace)	**Modify** (modification of a QuarkXPress project using the modify parameter)	**XML** (deconstruction of a QuarkXPress project using the xml namespace)
<?xml version="1.0" encoding="UTF-8"?>	XML declaration	XML declaration	XML declaration
<!ENTITY eot "&#38;#x0004;"> <!ENTITY enq "&#38;#x0005;"> <!ENTITY ack "&#38;#x0006;"> <!ENTITY softReturn "&#38;#x0007;"> <!ENTITY bs "&#38;#x0008;"> <!ENTITY hTab "&#38;#x0009;"> <!ENTITY lineFeed "&#38;#x000A;"> <!ENTITY vTab "&#38;#x000B;"> <!ENTITY boxBreak "&#38;#x000C;"> <!ENTITY hardReturn "&#38;#x000D;"> <!ENTITY so "&#38;#x000E;"> <!ENTITY flexSpace "&#38;#x000F;"> <!ENTITY dle "&#38;#x0010;"> <!ENTITY dcOne "&#38;#x0011;"> <!ENTITY dcTwo "&#38;#x0012;"> <!ENTITY dcThree "&#38;#x0013;"> <!ENTITY dcFour "&#38;#x0014;"> <!ENTITY nak "&#38;#x0015;"> <!ENTITY syn "&#38;#x0016;"> <!ENTITY etb "&#38;#x0017;"> <!ENTITY can "&#38;#x0018;"> <!ENTITY em "&#38;#x0019;">	Entities that represent QuarkXPress special characters. **Note:** Some entities, such as softreturn, are different for QuarkXPress than they are in the Unicode(R) specification.	Entities that represent QuarkXPress special characters. **Note:** Some entities, such as softreturn, are different for QuarkXPress than they are in the Unicode specification.	Entities that represent QuarkXPress special characters. **Note:** Some entities, such as softreturn, are different for QuarkXPress than they are in the Unicode specification.

<!ENTITY sub "&#38;#x001A;">			
<!ENTITY esc "&#38;#x001B;">			
<!ENTITY fs "&#38;#x001C;">			
<!ENTITY discReturn "&#38;#x001D;">			
<!ENTITY indentHere "&#38;#x001E;">			
<!ENTITY discHyphen "&#38;#x001F;">			
<!ENTITY shy "&#38;#x00AD;">			
<!ENTITY ensp "&#38;#x2002;">			
<!ENTITY emsp "&#38;#x2003;">			
<!ENTITY threePerEmSpace "&#38;#x2004;">			
<!ENTITY fourPerEmSpace "&#38;#x2005;">			
<!ENTITY sixPerEmSpace "&#38;#x2006;">			
<!ENTITY figureSpace "&#38;#x2007;">			
<!ENTITY punctSpace "&#38;#x2008;">			
<!ENTITY thinsp "&#38;#x2009;">			
<!ENTITY hairSpace "&#38;#x200A;">			
<!ENTITY zeroWidthSpace "&#38;#x200B;">			
<!ENTITY hyphen "&#38;#x2010;">			
<!ENTITY ndash "&#38;#x2013;">			
<!ENTITY mdash "&#38;#x2014;">			
<!ENTITY wordJoiner "&#38;#x2060;">			
<!ELEMENT PROJECT (SAVEAS?, LAYOUT*)>	Describes the QuarkXPress project using one or more LAYOUT elements and allows you to save a copy of the project.	Identifies the QuarkXPress project being modified and allows you to save a copy of that project.	Identifies the QuarkXPress project being deconstructed.
<!ATTLIST PROJECT			
PROJECTNAME CDATA #IMPLIED	Specifies the name of the file to construct.	Not applicable.	Identifies the QuarkXPress project being deconstructed.

JOBJACKET CDATA #IMPLIED	The name and absolute path (on the server computer) of the Job Jackets(R) file to use during construct. If the Job Jackets file cannot be located, cannot be read, or contains invalid XML, an error is returned. **Note:** You cannot create or modify Job Jackets files using the construct namespace and the modify attribute. To create or modify Job Jackets files, use the **Job Jackets Manager** dialog box (**Utilities** menu) in QuarkXPress.	Not applicable.	The name and path of the Job Jackets file associated with the deconstructed project.
JOBTICKET CDATA #IMPLIED	The name of the Job Ticket that contains the resources for this project. **Note:** All resources in the Job Ticket will be added to the project.	Not applicable.	The name of the Job Ticket associated with the deconstructed project.
XMLVERSION CDATA #IMPLIED>	Not applicable.	Not applicable.	Identifies the version of QuarkXPress Server from which the XML is being returned. Ensures compatibility with future versions of the DTD. For example, the value 8.0 is returned for

			QuarkXPress Server 8.0.
<!ELEMENT SAVEAS EMPTY>	Lets you save a constructed QuarkXPress project to a specific location on the server computer. Roughly equivalent to choosing **File > Save As** in QuarkXPress.	Lets you save a modified QuarkXPress project to a specific location on the server computer. Roughly equivalent to choosing **File > Save As** in QuarkXPress.	Not applicable.
<!ATTLIST SAVEAS			
NEWNAME CDATA #IMPLIED	Specifies a name for the project being saved.	Specifies a name for the project being saved. Can be a relative path to the document pool.	Not applicable.
PATH CDATA #IMPLIED	The absolute path on the server computer for saving the project.	The absolute path on the server computer for saving the project.	Not applicable.
SAVETOPOOL (true | false) "true"	Specifies whether the project should be saved to the document pool, in addition to saving it in the location specified in the PATH attribute.	Specifies whether the project should be saved to the document pool, in addition to saving it in the location specified in the PATH attribute.	Not applicable.
REPLACE (true | false) "true">	Indicates whether the saved project should replace any existing file with the same name in the specified location. An index number gets appended to the file name if this value is set to *false* and a file with the supplied name exists at the specification	Indicates whether the saved project should replace any existing file with the same name in the specified location. An index number gets appended to the file name if this value is set to *false* and a file with the supplied name exists at the specification location.	Not applicable.

	location. For example, if NEWNAME = file.qxp and the REPLACE value is set to false, the file is saved as file1.qxp when a file with the same name exists at the specified location.	For example, if NEWNAME = file.qxp and the REPLACE value is set to false, the file is saved as file1.qxp when a file with the same name exists at the specified location.	
<!ELEMENT LAYOUT (ID, LAYOUTPROPERTY?, ARTICLE*, LAYER*, (SPREAD | BOX | TABLE)*)>	Describes a layout in a project.	Identifies a layout to be modified. Use the ID@NAME or ID@UID attribute to indicate the target layout. Layout numbers start with 1; layout=1 refers to the first layout in the project. Note: If you want to modify existing boxes, regardless of where the boxes appear in the project, boxes to modify can be specified as direct children of the LAYOUT element, rather than being enclosed within a specific SPREAD.	Specifies the layout number in the ID@UID element and the layout name in the ID@NAME element.
<!ATTLIST LAYOUT			
POINTSPERINCH CDATA #IMPLIED	Not applicable.	Not applicable.	Specifies how many points to use per inch for measurements.
<!ELEMENT ID EMPTY>	Lets you specify a name for a LAYOUT, LAYER, BOX, LINKEDBOX, TABLE, GROUP, or COMPOSITION	Identifies an object by its UID or NAME. **Note:** QuarkXPress Server evaluates the ID element for	Identifies an object by its unique ID and by its name (if any). If a NAME value exists, the NAME displays in the content of the ID

	ZONE. Lets you specify a unique ID for a SPREAD or PAGE.	a NAME value first and for a UID second. If a NAME is found, the UID is ignored.	element: <ID UID=456 NAME=Name of box>Name of box</ID> If a NAME value does not exist, the UID displays in the content of the ID element: <ID UID=457>457</ID> **Note**: If a NAME value does not exist for a box, the word Box and the box UID are concatenated and display in the XML.
<!ATTLIST ID NAME CDATA #IMPLIED	The name of the parent element. The NAME is assigned to QuarkXPress elements during document construction. For example, NAME="BOX1" would be assigned to a box after it has been constructed. Required for LAYOUT, LAYER, BOX, TABLE, GROUP, and COMPOSITION ZONE elements. QuarkXPress Server automatically assigns a UID to such elements. Ignored for	The name of the LAYOUT, LAYER, SPREAD, BOX, TABLE, GROUP, or element to be modified.	The name of the parent element.

	spreads and pages.		
UID CDATA #IMPLIED>	Required for PAGE and SPREAD elements. Ignored for all other element types.	The unique ID of the element to be modified.	Specifies the unique ID of an element in the QuarkXPress project.
<!ELEMENT LAYOUTPROPERTY (MARGINS, COLUMNGUIDES)>	Describes the layout specifications for a particular layout.	LAYOUTPROPERTY and its child elements and attributes are only valid for construct requests.	LAYOUTPROPERTY and its child elements and attributes are only valid for construct requests.
<!ATTLIST LAYOUTPROPERTY			
HEIGHT CDATA #REQUIRED	Height of layout to be constructed.	Not applicable.	Not applicable.
WIDTH CDATA #REQUIRED	Width of layout to be constructed.	Not applicable.	Not applicable.
FACINGPAGES (true | false) "false"	Whether layout should have facing pages.	Not applicable.	Not applicable.
AUTOMATICBOX (true | false) "false"	Whether layout should have automatic box.	Not applicable.	Not applicable.
STORYDIRECTION (HORIZONTAL | VERTICAL) #IMPLIED>	STORYDIRECTION layout should have facing pages.	Not applicable.	Not applicable.
<!ELEMENT MARGINS EMPTY>	Defines margin placement in a layout.	Not applicable.	Not applicable.
<!ATTLIST MARGINS			
TOP CDATA #REQUIRED	Top margin.	Not applicable.	Not applicable.
LEFT CDATA #REQUIRED	Left margin.	Not applicable.	Not applicable.
BOTTOM CDATA #REQUIRED	Bottom margin.	Not applicable.	Not applicable.
RIGHT CDATA #REQUIRED>	Right margin.	Not applicable.	Not applicable.
<!ELEMENT COLUMNGUIDES EMPTY>	Defines the position of column guides in a layout.	Not applicable.	Not applicable.
<!ATTLIST COLUMNGUIDES			
COLUMNCOUNT CDATA #REQUIRED	Number of columns in automatic text box.	Not applicable.	Not applicable.
GUTTERWIDTH CDATA #REQUIRED>	Width in points between columns.	Not applicable.	Not applicable.
<!ELEMENT ARTICLE (ID, RGBCOLOR?, COMPONENT+)>	Describes an Article (a series of one or more COMPONENT	Describes an Article (a series of one or more COMPONENT	Describes an Article (a series of one or more COMPONENT

	elements). Note: New articles should not be created in a QuarkXPress project in systems working directly with QPS. Instead, create an article only within a QuarkCopyDesk(R) file. To assign an article in QPS(R), use the QPS SDK.	elements).	elements).
<!ATTLIST ARTICLE			
OPERATION (CREATE | DELETE) #IMPLIED DOCFORMAT (LIGHTWEIGHT | FULLFEATURED) "LIGHTWEIGHT" EXPORTARTICLE (true | false) "false">	Describes the type of Article.	Not applicable.	Describes the type of Article. "LIGHTWEIGHT" and "FULLFEATURED" articles are forms of QuarkCopyDesk articles that can be constructed/modified through QuarkXPress Server.
<!ELEMENT COMPONENT EMPTY>	The component(s) that make up an article. Required for ARTICLE element.	The component(s) that make up an article. Required for ARTICLE element.	The component(s) that make up an article.
<!ATTLIST COMPONENT			
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the specified component from the ARTICLE.	Not applicable.
NAME CDATA #IMPLIED	The name given to a specific component in an ARTICLE. Required for COMPONENT.	The name given to a specific component in an ARTICLE. Required for COMPONENT.	Specifies the name of the component in the ARTICLE.
UID CDATA #IMPLIED	QuarkXPress Server	The unique ID of the	Specified the unique ID of the .

	automatically assigns a unique ID to components.	COMPONENT to be modified.	
BOXNAME CDATA #IMPLIED	Specifies the name of the user-assigned box to which the COMPONENT belongs.	Specifies the name of the user-assigned box to which the COMPONENT belongs.	Specifies the name of the user-assigned box to which the COMPONENT belongs.
BOXUID CDATA #IMPLIED	Not applicable.	Specifies the ID of the QuarkXPress Server-assigned box to which the COMPONENT belongs.	Specifies the ID of the QuarkXPress Server-assigned box to which the COMPONENT belongs.
COMPONENTCLASS (CT_TEXT | CT_PICT | CT_GROUP) "CT_TEXT"	Describes whether the component resides in a text box, picture box, or group.	Describes whether the component resides in a text box, picture box, or group.	Describes whether the component resides in a text box, picture box, or group.
ROWNUM CDATA #IMPLIED	If the component resides in a Table cell, the value will describe the row number.	If the component resides in a Table cell, the value will describe the row number.	If the component resides in a Table cell, the value will describe the row number.
COLNUM CDATA #IMPLIED>	If the component resides in a Table cell, the value will describe the column number.	If the component resides in a Table cell, the value will describe the column number.	If the component resides in a Table cell, the value will describe the column number.
<!ELEMENT SPREAD (ID, PAGE*, (BOX | TABLE | GROUP | COMPOSITIONZONE)*)>	Describes a spread (a series of one or more PAGE elements, divided by a SPINE)	Identifies the spread to be modified.	Describes a spread (a series of one or more PAGEs, divided by a SPINE).
<!ATTLIST SPREAD OPERATION (CREATE | DELETE) #IMPLIED>	Not applicable.	Specifies whether to create or delete the indicated spread.	Not applicable.
<!ELEMENT PAGE (ID)>	A page to be created.	The page to be created or deleted. **Note:** To locate a page, for example, for creating a box,	Indicates a page's absolute page number (in the ID@UID element) **Note:** Page

		you use the GEOMETRY@PAGE attribute in the BOX element.	names are not returned.
<!ATTLIST PAGE OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the indicated page.	Not applicable.
MASTER CDATA #IMPLIED	Identifies the master page from which to create a page. This value should be specified as a number, with 3 indicating the first master page. **Note:** Only the number of a master page is included in this attribute. The definition of the master page is stored in the project's Job Jackets file.	Identifies the master page from which to create a page. This value should be specified as a number, with 3 indicating the first master page.	Identifies the master page that is applied to a page. Specified as a number, with "1" indicating the first master page. **Note:** Only the number of a master page is included in this attribute. The definition of the master page is stored in the project's Job Jackets file.
POSITION (LEFTOFSPINE | RIGHTOFSPINE) "RIGHTOFSPINE">	Specifies whether a page should be on the left or right side of the spine.	Specifies whether a page should be on the left or right side of the spine.	Specifies whether a page is on the left or right of the spine.
<!ELEMENT BOX (ID, METADATA?, (TEXT | PICTURE | GEOMETRY | CONTENT | SHADOW | FRAME | PLACEHOLDER | CONTENTPH)*)>	Describes a text box or picture box. **Note:** On construct, you must provide a box name in the ID@NAME attribute; QuarkXPress Server assigns an ID@UID to each BOX you create. **Note:** When a box is created, its page number is inferred from the GEOMETRY@P	Identifies a text box or picture box to be modified. You can use either the ID@UID or ID@NAME value to identify the box. **Note:** Named boxes can be easily identified by an XPath search for //BOX[@NAME]).	Describes a text box or picture box. If a NAME value exists, the NAME displays in the content of the ID element: <ID UID=456 NAME=*Name of box* >Name of box</ID> If a NAME value does not exist, the UID displays in the content of the ID element: <ID

	AGE attribute.		UID=457>457</ID>
<!ATTLIST BOX			
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the indicated box.	Not applicable.

BOXTYPE (CT_NONE | CT_TEXT | CT_PICT) #IMPLIED	The box type:
	CT_NONE = No box type specified.
	CT_TEXT = Text box
	CT_PICT = Picture box
		The box type:
		CT_NONE = No box type specified.
		CT_TEXT = Text box
		CT_PICT = Picture box
			The box type:
			CT_NONE = No box type specified.
			CT_TEXT = Text box
			CT_PICT = Picture box

COLOR CDATA #IMPLIED	Identifies the background color of a box. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the project's Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the background color of a box. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the project's Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server. The color definition can also be based on an existing color created and saved in the project.	Identifies the background color of a box. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the project's Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server. The color definition can also be based on an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of a box's background color, specified as a float value from 0 to 100.	Specifies the shade of a box's background color, specified as an integer percentage from 0 to 100.	Specifies the shade of a box's background color, specified as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of a box's background color, specified as a float value from 0 to 100.	Specifies the opacity of a box's background color, specified as an integer percentage from 0 to 100.	Indicates the opacity of a box's background color, specified as an integer percentage from 0 to 100.
BLENDSTYLE (SOLID | LINEAR | MIDLINEAR | RECTANGULAR |	Specifies the type of blend applied	Specifies the type of blend applied	Specifies the type of blend applied

DIAMOND | CIRCULAR | FULLCIRCULAR | none) "none"	to this box (linear, circular, rectangular, etc.).	to this box (linear, circular, rectangular, etc.).	to this box (linear, circular, rectangular, etc.).
BLENDANGLE CDATA #IMPLIED	Specifies the angle of the blend.	Specifies the angle of the blend.	Specifies the angle of the blend.
BLENDCOLOR CDATA #IMPLIED	Specifies the second color of the blend. The first color of the blend is the color applied to the box, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the box, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the box, as in QuarkXPress.
BLENDSHADE CDATA #IMPLIED	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the box.	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the box.	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the box.
BLENDOPACITY CDATA #IMPLIED	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the box.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the box.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the box.
ANCHOREDIN CDATA #IMPLIED>	Not applicable.	Not applicable.	Indicates an anchored box and identifies its parent box.
<!ELEMENT METADATA (VALUE+)>	Specifies if the box will have metadata associated with it. Metadata takes the form of key/value pairs.	Specifies if the box will have metadata associated with it. Metadata takes the form of key/value pairs.	Describes the metadata associated with the box.
<!ELEMENT VALUE (#PCDATA)>	Specifies the VALUE of the key/value pair. The value can be given in CDATA form only, such as: <METADATA>	Specifies the VALUE of the key/value pair. The value can be given in CDATA form only, such as: <METADATA>	Specifies the VALUE of the key/value pair. The value can be given in CDATA form only, such as: <METADATA>

	<VALUE KEY="myKey"><![CDATA[METADATAVALUE]]></VALUE></METADATA>	<VALUE KEY="myKey"><![CDATA[METADATAVALUE]]></VALUE></METADATA>	<VALUE KEY="myKey"><![CDATA[METADATAVALUE]]></VALUE></METADATA>
<!ATTLIST VALUE			
KEY CDATA #REQUIRED>	Specifies the KEY attribute of the key/value pair.	Specifies the KEY attribute of the key/value pair. Metadata that contains a value for KEY but no value for VALUE will delete any metadata matching the value for KEY.	Specifies the KEY attribute of the key/value pair.
<!ELEMENT TEXT ((INSET)*, STORY)>	Container for an INSET and STORY element.	Container for an INSET and STORY element.	Container for an INSET and STORY element.
<!ATTLIST TEXT			
ANGLE CDATA #IMPLIED	Specifies a rotation angle for text as a floating-point value between -360 degrees and 360 degrees.	Specifies a rotation angle for text as a floating-point value between -360 degrees and 360 degrees.	Indicates a rotation angle for text as a floating-point value between -360 degrees and 360 degrees.
SKEW CDATA #IMPLIED	Specifies a skew angle for text as a floating-point value from -75 degrees to 75 degrees.	Specifies a skew angle for text as a floating-point value from -75 degrees to 75 degrees.	Indicates a skew angle for text as a floating-point value from -75 degrees to 75 degrees.
COLUMNS CDATA #IMPLIED	Specifies a number of columns in a text box.	Specifies a number of columns in a text box.	Indicates a number of columns in a text box.
GUTTERWIDTH CDATA #IMPLIED	Specifies the gutter width between columns in a text box.	Specifies the gutter width between columns in a text box.	Indicates the gutter width between columns in a text box.
FLIPVERTICAL (true | false | none) "none"	Flips the text vertically in a text box.	Flips the text vertically in a text box.	Indicates the text is flipped vertically in a text box.
FLIPHORIZONTAL (true | false | none) "none"	Flips the text horizontally in a text box.	Flips the text horizontally in a text box.	Indicates the text is flipped horizontally in a text box.

VERTICALALIGNMENT (TOP | CENTERED | BOTTOM | JUSTIFIED | none) "none"	Vertically aligns the text.	Vertically aligns the text.	Indicates the vertical alignment of text.
INTERPARAGRAPHMAX CDATA #IMPLIED	Specifies the space between two consecutive paragraphs	Specifies the space between two consecutive paragraphs	Specifies the space between two consecutive paragraphs
FIRSTBASELINEMIN (ASCENT | CAPHEIGHT | CAPACCENT | none) "none"	Specifies the minimum distance between the top edge of a text box and the baseline of the first line of text. ASCENT = Specifies the distance based on the space needed for the accent mark of the tallest character. CAPHEIGHT= Specifies the distance based on the cap height of the tallest character. CAPACCENT = Specifies the distance based on the cap height of the tallest character plus the space required for an accent mark over an uppercase character.	Specifies the minimum distance between the top edge of a text box and the baseline of the first line of text. ASCENT = Specifies the distance based on the space needed for the accent mark of the tallest character. CAPHEIGHT= Specifies the distance based on the cap height of the tallest character. CAPACCENT = Specifies the distance based on the cap height of the tallest character plus the space required for an accent mark over an uppercase character.	Indicates the minimum distance between the top edge of a text box and the baseline of the first line of text. ASCENT = Specifies the distance based on the space needed for the accent mark of the tallest character. CAPHEIGHT= Specifies the distance based on the cap height of the tallest character. CAPACCENT = Specifies the distance based on the cap height of the tallest character plus the space required for an accent mark over an uppercase character.
OFFSET CDATA #IMPLIED>	Specifies the distance between the first text baseline in the text box and the top inside edge of the text box.	Specifies the distance between the first text baseline in the text box and the top inside edge of the text box.	Indicates the distance between the first text baseline in the text box and the top inside edge of the text box.
RUNTEXTAROUNDALLSIDES (true | false | none) "none"	Indicates text runaround on all sides of an item.	Indicates text runaround on all sides of an item.	Indicates text runaround on all sides of an item.
TEXTORIENTATION (ROTATE | SKEW | ROTATEANDSKEW | NOROTATEANDSKEW | none)	Specifies how the text should be attached to a line.	Specifies how the text should be attached to a line.	Indicates how the text is attached to a line.

<table>
<tr><td>"none"</td><td></td><td></td><td></td></tr>
<tr><td>TEXTALIGN (ASCENT | CENTER | BASELINE | DESCENT | none) "none"</td><td>Specifies the part of a font to use for positioning characters on a line.</td><td>Specifies the part of a font to use for positioning characters on a line.</td><td>Indicates the part of a font being used for positioning characters on a line.</td></tr>
<tr><td>TEXTALIGNWITHLINE (TOP | CENTER | BOTTOM | none) "none"</td><td>Specifies how to align text to a line.</td><td>Specifies how to align text to a line.</td><td>Indicates text is aligned to a line.</td></tr>
<tr><td>FLIPTEXT (true | false | none) "none"></td><td>Flips the characters horizontally on a line.</td><td>Flips the characters horizontally on a line.</td><td>Indicates characters are horizontally flipped on a line.</td></tr>
<tr><td><!ELEMENT INSET EMPTY></td><td>Specifies the distance between the inside border of a text box and the text.</td><td>Specifies the distance between the inside border of a text box and the text.</td><td>Indicates the distance between the inside border of a text box and the text.</td></tr>
<tr><td><!ATTLIST INSET</td><td></td><td></td><td></td></tr>
<tr><td>MULTIPLEINSETS (true | false | none) "none"</td><td>Specifies multiple insets.</td><td>Specifies multiple insets.</td><td>Indicates multiple insets.</td></tr>
<tr><td>TOP CDATA #IMPLIED</td><td>Specifies the distance between the top inside border of a text box and the text.</td><td>Specifies the distance between the top inside border of a text box and the text.</td><td>Indicates the distance between the top inside border of a text box and the text.</td></tr>
<tr><td>BOTTOM CDATA #IMPLIED</td><td>Specifies the distance between the bottom inside border of a text box and the text.</td><td>Specifies the distance between the bottom inside border of a text box and the text.</td><td>Indicates the distance between the bottom inside border of a text box and the text.</td></tr>
<tr><td>RIGHT CDATA #IMPLIED</td><td>Specifies the distance between the right inside border of a text box and the text.</td><td>Specifies the distance between the right inside border of a text box and the text.</td><td>Indicates the distance between the right inside border of a text box and the text.</td></tr>
<tr><td>LEFT CDATA #IMPLIED</td><td>Specifies the distance between the right inside border of a text box and the text.</td><td>Specifies the distance between the right inside border of a text box and the text.</td><td>Indicates the distance between the right inside border of a text box and the text.</td></tr>
<tr><td>ALLEDGES CDATA #IMPLIED></td><td>Specifies the distance between the inside border of all sides of a text box and the text.</td><td>Specifies the distance between the inside border of all sides of a text box and the text.</td><td>Indicates the distance between the inside border of all sides of a text box and the text.</td></tr>
<tr><td><!ELEMENT STORY (COPYFIT?, (PARAGRAPH | RICHTEXT |</td><td>Describes a text story in a text box or a chain of text</td><td>Describes a text story in a text box or a chain of text</td><td>Describes a text story in a text box or a chain of text</td></tr>
</table>

ANCHOREDBOXREF | LINKEDBOX | TEXTNODEPH | TEXTPH | HIDDEN | LIST)*, OVERMATTER?)> <!ATTLIST STORY	boxes.	boxes.	boxes.
CLEAROLDTEXT (true | false) "false"	Not applicable.	Clears any existing text from the box.	Not applicable.
FITTEXTTOBOX (true | false) "false"	Increases or decreases the size of the text to fit into the text box or text chain. **Note:** Text size increases only if **Allow Text to Grow** is checked in **Text Modifier** preferences (**QuarkXPress Server/Edit > Preferences**) in QuarkXPress Server.	Increases or decreases the size of the text to fit into the text box or text chain. **Note:** Text size increases only if **Allow Text to Grow** is checked in **Text Modifier** preferences (**QuarkXPress Server/Edit > Preferences**) in QuarkXPress Server.	Not applicable.
FILE CDATA #IMPLIED	The absolute path (on the server computer) to import a text document from.	The absolute path (on the server computer) to import a text document from.	Not applicable.
CONVERTQUOTES (true | false) "true"	Converts straight quotation marks to typesetter's quotation marks and double hyphens to em dashes in an imported text file.	Converts straight quotation marks to typesetter's quotation marks and double hyphens to em dashes in an imported text file.	Not applicable.
INCLUDESTYLESHEETS (true | false) "true"	Adds any style sheets in an imported text file or document to the QuarkXPress project.	Adds any style sheets in an imported text file or document to the QuarkXPress project.	Not applicable.
STORYDIRECTION (HORIZONTAL | VERTICAL) #IMPLIED>	Specified direction of this story.	Specified direction of this story.	Specified direction of this story.
<!ELEMENT COPYFIT EMPTY>	Not applicable.	Not applicable.	Indicates whether the copy in this text box or chain fits the available space.

<!ATTLIST COPYFIT			
STATE (fit | overFit | underFit) "fit"	Not applicable.	Not applicable.	Indicates whether the text currently fits in the box (fit), is too long (overFit), or is too short (underFit).
FITAMOUNT CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates the vertical distance in points by which text in a text box is overFit or underFit. See the STATE element.
NUMBEROFCHARACTERS CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates how many characters are included in the story.
NUMBEROFWORDS CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates how many words are included in the story.
NUMBEROFLINES CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates how many lines are included in the story.
FITLINEAMOUNT CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates how many lines the text is overfit or underfit.
STORYDEPTHAMOUNT CDATA #IMPLIED>	Not applicable.	Not applicable.	Not applicable.
<!ELEMENT PARAGRAPH ((TABSPEC | RULE | FORMAT | RICHTEXT | ANCHOREDBOXREF | HIDDEN)*)>	Describes a paragraph.	Describes a paragraph.	Describes a paragraph.
<!ATTLIST PARAGRAPH			
PARASTYLE CDATA #IMPLIED	Applies a paragraph style sheet to text. **Note:** Only the name of a paragraph style sheet is included in this attribute. The definition of the style sheet is stored in the projects Job	Applies a paragraph style sheet to text. **Note:** Only the name of a paragraph style sheet is included in this attribute. The definition of the style sheet is stored in the projects Job	Identifies the paragraph style sheet applied to a paragraph. **Note:** Only the name of a paragraph style sheet is included in this attribute. The definition of the style sheet is stored in the

	Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.
PARACHAR (HARDRETURN | VTAB | BOXBREAK) "HARDRETURN"	Defines a breaking character for a paragraph.	Defines a breaking character for a paragraph.	Defines a breaking character for a paragraph.
MERGE (true | false) "false">	Specifies whether formatting from a previous PARAGRAPH or RICHTEXT element should be carried over to the next.	Specifies whether formatting from a previous PARAGRAPH or RICHTEXT element should be carried over to the next.	Indicates whether formatting from a previous PARAGRAPH or RICHTEXT element is carried over to the next.
<!ELEMENT TEXTNODEPH ((TEXTNODEPH | PARAGRAPH | RICHTEXT | OVERMATTER | TEXTPH)*, METADATA?)>	A text node placeholder allows metadata to be defined hierarchically on a region of text, and can contain further text node placeholders and text placeholders.	A text node placeholder allows metadata to be defined hierarchically on a region of text, and can contain further text node placeholders and text placeholders.	A text node placeholder allows metadata to be defined hierarchically on a region of text, and can contain further text node placeholders and text placeholders.
<!ATTLIST TEXTNODEPH			
NAME CDATA #REQUIRED	The name of the text node placeholder. A placeholder name may not be Unique within the Box or XML Hierarchy.	The name of the text node placeholder. A placeholder name may not be Unique within the Box or XML Hierarchy.	The name of the text node placeholder. A placeholder name may not be Unique within the Box or XML Hierarchy.
OWNER (1347639377) "1347639377">	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXT ID (1347639377). All placeholders created through Modifier should	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXT ID (1347639377). All placeholders created through Modifier should	The XTensions ID of the XTensions that created this placeholder.

	use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute.	use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute).	
<!ELEMENT TEXTPH ((PARAGRAPH | RICHTEXT | OVERMATTER)*, METADATA?)>	A text placeholder allows metadata to be defined on a region of text.	A text placeholder allows metadata to be defined on a region of text.	A text placeholder allows metadata to be defined on a region of text.
<!ATTLIST TEXTPH			
NAME CDATA #REQUIRED	The name of the text node placeholder.	The name of the text node placeholder.	The name of the text node placeholder.
OWNER (1347639377) "1347639377">	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXT ID (1347639377). All placeholders created through Modifier should use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute.	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXT ID (1347639377). All placeholders created through Modifier should use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute.	The XTensions ID of the XTensions that created this placeholder.
<!ELEMENT FORMAT (KEEPLINESTOGETHER?, DROPCAP?)>	Describes formatting for a PARAGRAPH element.	Describes formatting for a PARAGRAPH element.	Describes formatting for a PARAGRAPH element.
<!ATTLIST FORMAT			
SPACEBEFORE CDATA #IMPLIED	Describes the amount of space before a paragraph.	Describes the amount of space before a paragraph.	Describes the amount of space before a paragraph.
SPACEAFTER CDATA #IMPLIED	Describes the amount of space after a paragraph.	Describes the amount of space after a paragraph.	Describes the amount of space after a paragraph.
LEFTINDENT CDATA #IMPLIED	Describes the	Describes the	Describes the

	amount of space in a paragraphs left indent.	amount of space in a paragraphs left indent.	amount of space in a paragraphs left indent.
RIGHTINDENT CDATA #IMPLIED	Describes the amount of space in a paragraphs right indent.	Describes the amount of space in a paragraphs right indent.	Describes the amount of space in a paragraphs right indent.
FIRSTLINE CDATA #IMPLIED	Describes the amount of space in a paragraphs first-line indent.	Describes the amount of space in a paragraphs first-line indent.	Describes the amount of space in a paragraphs first-line indent.
LEADING CDATA #IMPLIED	Describes a paragraphs line spacing.	Describes a paragraphs line spacing.	Describes a paragraphs line spacing.
ALIGNMENT (LEFT | RIGHT | CENTERED | JUSTIFIED | FORCED) "LEFT"	Indicates whether a paragraph should be left-aligned, right-aligned, centered, justified, or force-justified. **Note:** JUSTIFIED aligns the text in a paragraph to the left and right indentations, except for the last line. FORCED justifies every line, including the last line.	Indicates whether a paragraph should be left-aligned, right-aligned, centered, justified, or force-justified. **Note:** JUSTIFIED aligns the text in a paragraph to the left and right indentations, except for the last line. FORCED justifies every line, including the last line.	Indicates whether a paragraph is left-aligned, right-aligned, centered, justified, or force-justified. **Note:** JUSTIFIED aligns the text in a paragraph to the left and right indentations, except for the last line. FORCED justifies every line, including the last line.
HANDJ CDATA #IMPLIED	Identifies a hyphenation and justification specification to be applied to a paragraph. **Note:** Only the name of an H&J specification is included in this attribute. The definition of the H&J specification is stored in the projects Job Jackets file or defined using the **Document**	Identifies a hyphenation and justification specification to be applied to a paragraph. **Note:** Only the name of an H&J specification is included in this attribute. The definition of the H&J specification is stored in the projects Job Jackets file or defined using the **Document**	Identifies the hyphenation and justification specification applied to a paragraph. **Note:** Only the name of an H&J specification is included in this attribute. The definition of the H&J specification is stored in the projects Job Jackets file or defined using the **Document**

	Controls submenu in QuarkXPress Server.	**Controls** submenu in QuarkXPress Server.	**Controls** submenu in QuarkXPress Server.
KEEPWITHNEXT (true | false | none) "none"	Specifies whether the last lines of a paragraph should always appear on the same page as the next paragraph.	Specifies whether the last lines of a paragraph should always appear on the same page as the next paragraph.	Specifies whether the last lines of a paragraph should always appear on the same page as the next paragraph.
HANGINGCHARACTERS CDATA #IMPLIED	Describes the hanging character set used by this paragraph.	Describes the hanging character set used by this paragraph.	Describes the hanging character set used by this paragraph.
CHARACTERALIGNMENT (ROMANBASELINE | EMBOXTOP | EMBOXCENTER | EMBOXBOTTOM | ICFBOXTOP | ICFBOXBOTTOM) "ROMANBASELINE">	Defines the character alignment used by this paragraph. For a story with horizontal direction, EMBOXTOP, EMBOXBOTTOM, ICFBOXTOP, ICFBOXBOTTOM are applicable. For a story with vertical direction, EMBOXRIGHT, EMBOXLEFT, ICFBOXRIGHT, ICFBOXLEFT are applicable.	Defines the character alignment used by this paragraph. For a story with horizontal direction, EMBOXTOP, EMBOXBOTTOM, ICFBOXTOP, ICFBOXBOTTOM are applicable. For a story with vertical direction, EMBOXRIGHT, EMBOXLEFT, ICFBOXRIGHT, ICFBOXLEFT are applicable.	Defines the character alignment used by this paragraph. For a story with horizontal direction, EMBOXTOP, EMBOXBOTTOM, ICFBOXTOP, ICFBOXBOTTOM are applicable. For a story with vertical direction, EMBOXRIGHT, EMBOXLEFT, ICFBOXRIGHT, ICFBOXLEFT are applicable.
<!ELEMENT KEEPLINESTOGETHER EMPTY>	The Keep Lines Together feature specifies whether lines in paragraphs flow together or are separated when they reach the bottoms of columns.	The Keep Lines Together feature specifies whether lines in paragraphs flow together or are separated when they reach the bottoms of columns.	The Keep Lines Together feature specifies whether lines in paragraphs flow together or are separated when they reach the bottoms of columns.
<!ATTLIST KEEPLINESTOGETHER			
ENABLED (true | false | none) "none"	Specifies whether or not this feature is enabled.	Specifies whether or not this feature is enabled.	Specifies whether or not this feature is enabled.
ALLLINESINPARA (true | false | none) "none"	Specifies whether this is for all lines	Specifies whether this is for all lines	Specifies whether this is for all lines

	in the paragraph or has a specific start and end.	in the paragraph or has a specific start and end.	in the paragraph or has a specific start and end.
STARTLINE CDATA #IMPLIED	Specifies the number of lines at the beginning of a paragraph before wrapping text to keep lines together.	Specifies the number of lines at the beginning of a paragraph before wrapping text to keep lines together.	Specifies the number of lines at the beginning of a paragraph before wrapping text to keep lines together.
ENDLINE CDATA #IMPLIED>	Specifies the number of lines at the end of a paragraph before wrapping text to keep lines together.	Specifies the number of lines at the end of a paragraph before wrapping text to keep lines together.	Specifies the number of lines at the end of a paragraph before wrapping text to keep lines together.
<!ELEMENT DROPCAP EMPTY>	Describes a drop-capital effect at the beginning of a paragraph, which is when initial characters display at a large size and hang two or more lines below the first line of a paragraph.	Describes a drop-capital effect at the beginning of a paragraph, which is when initial characters display at a large size and hang two or more lines below the first line of a paragraph.	Describes a drop-capital effect at the beginning of a paragraph, which is when initial characters display at a large size and hang two or more lines below the first line of a paragraph.
<!ATTLIST DROPCAP			
CHARCOUNT CDATA #REQUIRED	Specifies how many characters should be included in a drop-cap effect.	Specifies how many characters should be included in a drop-cap effect.	Specifies how many characters are included in a drop-cap effect.
LINECOUNT CDATA #REQUIRED>	Specifies the number of lines a drop-caps should hang in the paragraph.	Specifies the number of lines a drop-caps should hang in the paragraph.	Specifies the number of lines drop-caps hang in the paragraph.
<!ELEMENT LOCKTOGRID EMPTY>	Specifies whether this paragraph is locked to the baseline grid. You can choose to lock to the page grid or the text box grid.	Specifies whether this paragraph is locked to the baseline grid. You can choose to lock to the page grid or the text box grid.	Specifies whether this paragraph is locked to the baseline grid. You can choose to lock to the page grid or the text box grid.
<!ATTLIST LOCKTOGRID			
ENABLED (true | false | none) "none"	Specifies whether LOCKTOGRID	Specifies whether LOCKTOGRID	Specifies whether LOCKTOGRID

	is enabled.	is enabled.	is enabled.
GRIDLEVEL (PAGE | TEXTBOX) "PAGE"	Specifies whether GRID applies on page level or text box level.	Specifies whether GRID applies on page level or text box level.	Specifies whether GRID applies on page level or text box level.
GRIDTYPE (TOPLINE | BOTTOMLINE | LEFTLINE | RIGHTLINE | CENTERLINE | BASELINE) "BASELINE">	Specifies grid type applied on page level or text box level grid.	Specifies grid type applied on page level or text box level grid.	Specifies grid type applied on page level or text box level grid.
<!ELEMENT TABSPEC (TAB)+>	Describes a group of tab stops.	Describes a group of tab stops.	Describes a group of tab stops.
<!ELEMENT TAB EMPTY>	Describes a single tab stop.	Describes a single tab stop.	Describes a single tab stop.
<!ATTLIST TAB			
POSITION CDATA #REQUIRED	Specifies the position of a tab stop.	Specifies the position of a tab stop.	Specifies the position of a tab stop.
FILL CDATA #IMPLIED	Identifies one or two characters to repeat in order to fill the space between text and a tab stop.	Identifies one or two characters to repeat in order to fill the space between text and a tab stop.	Identifies one or two characters that repeat in order to fill the space between text and a tab stop.
ALIGNMENT (LEFT | RIGHT | CENTER | COMMA | DECIMAL | ALIGNON) "LEFT"	Indicates how a tab stop should be aligned. LEFT = Aligns text flush left on the tab stop. RIGHT = Aligns text flush right on the tab stop. CENTER = Aligns text centrally on the tab stop. DECIMAL = Aligns text on a decimal point (period). COMMA = Aligns text on a first comma. ALIGN ON = Aligns text on any character you specify in the ALIGNON attribute.	Indicates how a tab stop should be aligned. LEFT = Aligns text flush left on the tab stop. RIGHT = Aligns text flush right on the tab stop. CENTER = Aligns text centrally on the tab stop. DECIMAL = Aligns text on a decimal point (period). COMMA = Aligns text on a first comma. ALIGN ON = Aligns text on any character you specify in the ALIGNON attribute.	Indicates how a tab stop is aligned. LEFT = Aligns text flush left on the tab stop. RIGHT = Aligns text flush right on the tab stop. CENTER = Aligns text centrally on the tab stop. DECIMAL = Aligns text on a decimal point (period). COMMA = Aligns text on a first comma. ALIGN ON = Aligns text on any character you specify in the ALIGNON attribute.

ALIGNON CDATA #IMPLIED>	Specifies a specific character to align a tab stop on.	Specifies a specific character to align a tab stop on.	Specifies a specific character a tab stop is aligned on.
ENABLED (true | false) "true">			
<!ELEMENT RULE EMPTY>	Describes a rule above or below a paragraph.	Describes a rule above or below a paragraph.	Describes a rule above or below a paragraph.
<!ATTLIST RULE			
ENABLED (true | false | none) "none"	Specifies whether to add a rule to a paragraph or not.	Specifies whether to add a rule to a paragraph or not.	Specifies whether a rule is applied to a paragraph or not.
POSITION (ABOVE | BELOW) "BELOW"	Specifies whether a rule should be above or below a paragraph.	Specifies whether a rule should be above or below a paragraph.	Specifies whether a rule is above or below a paragraph.
LENGTH (TEXT | COLUMN | INDENTS) "INDENTS"	Specifies the length of a rule. TEXT = Rule is the same length as the first line of text in the paragraph (for rule above) or the last line of text in the paragraph (for rule below). COLUMN = Rule extends to edges of parent box or column. INDENTS = Rule extends from the paragraph's left indent to its right indent.	Specifies the length of a rule. TEXT = Rule is the same length as the first line of text in the paragraph (for rule above) or the last line of text in the paragraph (for rule below). COLUMN = Rule extends to edges of parent box or column. INDENTS = Rule extends from the paragraph's left indent to its right indent.	Specifies the length of a rule. TEXT = Rule is the same length as the first line of text in the paragraph (for rule above) or the last line of text in the paragraph (for rule below). COLUMN = Rule extends to edges of parent box or column. INDENTS = Rule extends from the paragraph's left indent to its right indent.
LEFT CDATA #IMPLIED	Specifies a distance to indent a rule farther from the left. A positive number moves the end-point to the right; a negative number moves the end-point to the left.	Specifies a distance to indent a rule farther from the left. A positive number moves the end-point to the right; a negative number moves the end-point to the left.	Specifies a distance a rule is indented farther from the left. A positive number moves the end-point to the right; a negative number moves the end-point to the left.
RIGHT CDATA #IMPLIED	Specifies a distance to indent a rule farther from	Specifies a distance to indent a rule farther from	Specifies a distance a rule is indented farther

	the right. A positive number moves the end-point to the left; a negative number moves the end-point to the right.	the right. A positive number moves the end-point to the left; a negative number moves the end-point to the right.	from the right. A positive number moves the end-point to the left; a negative number moves the end-point to the right.
OFFSET CDATA #IMPLIED	Specifies the amount of space between a rule and the paragraph to which it is attached.	Specifies the amount of space between a rule and the paragraph to which it is attached.	Specifies the amount of space between a rule and the paragraph to which it is attached.
WIDTH CDATA #IMPLIED	Specifies the thickness of a rule.	Specifies the thickness of a rule.	Specifies the thickness of a rule.
COLOR CDATA #IMPLIED	Identifies the color for a rule. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color for a rule. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color for a rule. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of a rules color, as an integer percentage from 0 to 100.	Specifies the shade of a rules color, as an integer percentage from 0 to 100.	Specifies the shade of a rules color, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of a rules color, specified as an integer percentage from 0 to 100.	Specifies the opacity of a rules color, specified as an integer percentage from 0 to 100.	Specifies the opacity of a rules color, specified as an integer percentage from 0 to 100.
STYLE CDATA #IMPLIED>	Identifies a Dashes & Stripes style (	Identifies a Dashes & Stripes style (	Identifies a Dashes & Stripes style (

	LINESTYLE) for a rule. **Note:** Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	LINESTYLE) for a rule. **Note:** Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	LINESTYLE) for a rule. **Note:** Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.
<!ELEMENT HIDDEN (RICHTEXT)*>	Given the OPCODE and OWNER, this will specify hidden text within the project.	Given the OPCODE and OWNER, this will specify hidden text within the project.	Given the OPCODE and OWNER, this will specify hidden text within the project.
<!ATTLIST HIDDEN			
DATALEN CDATA #IMPLIED	Not applicable.	Not applicable.	Number of characters the hidden text spans.
OPCODE CDATA #REQUIRED	Hidden text opcode is a four-byte field that contains ownerId, opcodeId,and hiddenTextType. The Hidden text opcode is usually the originating XTensions ID of the XTensions that owns this hidden text. Note that you MUST be certain that the handling XTensions will correctly understand the data being passed, and handle any errors.	Hidden text opcode is a four-byte field that contains ownerId, opcodeId,and hiddenTextType. The Hidden text opcode is usually the originating XTensions ID of the XTensions that owns this hidden text. Note that you MUST be certain that the handling XTensions will correctly understand the data being passed, and handle any errors. XTensions	Hidden text opcode is a four-byte field that contains ownerId, opcodeId,and hiddenTextType. The Hidden text opcode is usually the originating XTensions ID of the XTensions that owns this hidden text.

	XTensions that are not designed to handle inappropriate data may cause QuarkXPress Server to unexpectedly quit.	that are not designed to handle inappropriate data may cause QuarkXPress Server to unexpectedly quit.	
OWNER CDATA #IMPLIED	Represents the XTensions ID of the XTensions software that owns this hidden text.	Represents the XTensions ID of the XTensions software that owns this hidden text.	
TYPE (OPENPAREN | CLOSEPAREN | NONPAREN | CHARACTERTYPE) #IMPLIED>	The type of hidden text, as described in the XDK.	The type of hidden text, as described in the XDK.	The type of hidden text, as described in the XDK.
<!ELEMENT RICHTEXT (#PCDATA)>	Describes formatting for text. Use this element to apply additional formatting besides formatting applied with a paragraph or style sheet. **Note:** The RICHTEXT element replaces the TYPE element in QuarkXPress Server 7.2 and later.	Describes formatting for text. Use this element to apply additional formatting besides formatting applied with a paragraph or style sheet. **Note:** The RICHTEXT element replaces the TYPE element in QuarkXPress Server 7.2 and later.	Describes formatting for text, other than formatting applied with a paragraph or style sheet. **Note:** The RICHTEXT element replaces the TYPE element in QuarkXPress Server 7.2 and later.
<!ATTLIST RICHTEXT			
CHARSTYLE CDATA #IMPLIED	Identifies a character style sheet to be applied to text. **Note:** Only the name of an H&J specification is included in this attribute. The definition of the H&J specification is stored in the projects Job Jackets file or defined using the **Document**	Identifies a character style sheet to be applied to text.	Identifies the character style sheet applied to text. **Note:** Only the name of an H&J specification is included in this attribute. The definition of the H&J specification is stored in the projects Job Jackets file or defined using the **Document**

	Controls submenu in QuarkXPress Server.		**Controls** submenu in QuarkXPress Server.
PLAIN (true | false | none) "none"	Removes existing formatting and renders text as plain text.	Removes existing formatting and renders text as plain text.	Removes existing formatting and renders text as plain text.
MERGE (true | false) "false"	Specifies whether the formatting from the previous RICHTEXT tag should be carried into this RICHTEXT tag.	Specifies whether the formatting from the previous RICHTEXT tag should be carried into this RICHTEXT tag.	Specifies whether the formatting from the previous RICHTEXT tag should be carried into this RICHTEXT tag.
BOLD (true | false | none) "none"	Applies the bold type style to text.	Applies the bold type style to text.	Identifies the bold type style applied to text.
ITALIC (true | false | none) "none"	Applies the italic type style to text.	Applies the italic type style to text.	Identifies the italic type style applied to text.
FONT CDATA #IMPLIED	Identifies a font to be applied to text.	Identifies a font to be applied to text.	Identifies a font applied to text.
MISSINGFONT (true | false) "false"	If the font is missing on rendering, then this attribute is set to true. This allows you to identify when rendering a portion of text that the original font is missing on the machine where the rendering is taking place, and allows your application to substitute the font (overriding the inbuilt font mapping functionality in QuarkXPress Server). If the font specified in the XML is missing and if the	If the font is missing on rendering, then this attribute is set to true. This allows you to identify when rendering a portion of text that the original font is missing on the machine where the rendering is taking place, and allows your application to substitute the font (overriding the inbuilt font mapping functionality in QuarkXPress Server).	If the font is missing on rendering, then this attribute is set to true. This allows you to identify when rendering a portion of text that the original font is missing on the machine where the rendering is taking place, and allows your application to substitute the font (overriding the inbuilt font mapping functionality in QuarkXPress Server).

	MISSINGFONT attribute is present then this becomes the basis for applying font fallback on the particular text run if the FontFallBack preference is enabled. Otherwise this would cause an error because the required font is missing.		
PSFONTNAME CDATA #IMPLIED	Some fonts have different postscript and menu display names. The FONTNAME attribute describes the menu name of the font, and PSFONTNAME describes the internal postscript name of the font family.	Some fonts have different postscript and menu display names. The FONTNAME attribute describes the menu name of the font, and PSFONTNAME describes the internal postscript name of the font family.	Some fonts have different postscript and menu display names. The FONTNAME attribute describes the menu name of the font, and PSFONTNAME describes the internal postscript name of the font family.
SIZE CDATA #IMPLIED	Specifies a size for text, from 2 to 720 points.	Specifies a size for text, from 2 to 720 points.	Identifies the size of the text, from 2 to 720 points.
COLOR CDATA #IMPLIED	Identifies the color for text. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color for text. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color	Identifies the color for text. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color

		created and saved in the project.	created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of text color, as an integer percentage from 0 to 100.	Specifies the shade of text color, as an integer percentage from 0 to 100.	Identifies the shade of text color, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of text, specified as an integer percentage from 0 to 100.	Specifies the opacity of text, specified as an integer percentage from 0 to 100.	Identifies the opacity of text, specified as an integer percentage from 0 to 100.
NONBREAKING (true | false | none) "none"	Specifies if the text will be nonbreaking or not. Used for special character (e.g., for a hyphen: <RICHTEXT NONBREAKING="true">-</RICHTEXT>)	Specifies if the text will be nonbreaking or not. Used for special characters (e.g., for a thinspace : <RICHTEXT NONBREAKING="true">&thinsp;</RICHTEXT>)	Specifies if the text will be nonbreaking or not. Used for special characters (e.g., for a thinspace : <RICHTEXT NONBREAKING="true">&thinsp;</RICHTEXT>)
UNDERLINE (true | false | none) "none"	Applies the underline type style to text.	Applies the underline type style to text.	Identifies the underline type style applied to text.
WORDUNDERLINE (true | false | none) "none"	Applies the word underline type style to text.	Applies the word underline type style to text.	Identifies the word underline type style applied to text.
SMALLCAPS (true | false | none) "none"	Applies small caps to text.	Applies small caps to text.	Identifies small caps applied to text.
ALLCAPS (true | false | none) "none"	Applies all caps to text.	Applies all caps to text.	Identifies all caps applied to text.
SUPERSCRIPT (true | false | none) "none"	Applies the superscript type style to text.	Applies the superscript type style to text.	Identifies the superscript type style applied to text.
SUBSCRIPT (true | false | none) "none"	Applies the subscript type style to text.	Applies the subscript type style to text.	Identifies the subscript type style applied to text.
SUPERIOR (true | false | none) "none"	Applies the superior type style to text.	Applies the superior type style to text.	Identifies the superior type style applied to text.
OUTLINE (true | false | none) "none"	Applies the outline type style to text.	Applies the outline type style to text.	Identifies the outline type style applied to text.

SHADOW (true | false | none) "none"	Applies the shadow type style to text.	Applies the shadow type style to text.	Identifies the shadow type style applied to text.
STRIKETHRU (true | false | none) "none"	Applies the strikethru type style to text.	Applies the strikethru type style to text.	Identifies the strikethru type style applied to text.
EMPHASISMARK (NONE | DOT | BLACKCIRCLE | WHITECIRCLE | WHITESQUARE | FISHEYE | COMMA | BLACKSESAME | WHITESESAME | BLACKTRIANGLE) "NONE"	Allows an emphasis mark to be placed on this RICHTEXT.	Allows an emphasis mark to be placed on this RICHTEXT.	Allows an emphasis mark to be placed on this RICHTEXT.
BASELINESHIFT CDATA #IMPLIED	Shifts text up or down without affecting paragraph line spacing. A positive value raises text; a negative value lowers text.	Shifts text up or down without affecting paragraph line spacing. A positive value raises text; a negative value lowers text.	Identifies a shift of text up or down without affecting paragraph line spacing. A positive value raises text; a negative value lowers text.
HORIZONTALSCALE CDATA #IMPLIED	Applies a horizontal scale to text, which makes characters narrower or wider.	Applies a horizontal scale to text, which makes characters narrower or wider.	Identifies a horizontal scale applied to text, which makes characters narrower or wider.
VERTICALSCALE CDATA #IMPLIED	Applies a vertical scale to text, which makes characters taller or shorter. Specified as an integer percentage from 25 to 400.	Applies a vertical scale to text, which makes characters taller or shorter. Specified as an integer percentage from 25 to 400.	Identifies a vertical scale applied to text, which makes characters taller or shorter. Specified as an integer percentage from 25 to 400.
TRACKAMOUNT CDATA #IMPLIED	Adjusts the amount of space between characters and words.	Adjusts the amount of space between characters and words.	Identifies an amount of adjusted space applied between characters and words.
KERNAMOUNT CDATA #IMPLIED	Adjusts the amount of space between two characters.	Adjusts the amount of space between two characters.	Identifies an amount of adjusted space applied between two characters.

LIGATURES (true | false | none) "none"	Indicates whether standard ligatures should be applied.	Indicates whether standard ligatures should be applied.	Indicates whether standard ligatures are applied.
OT_STANDARD_LIGATURES (true | false | none) "none"	Applies the OpenType standard ligatures type style to text.	Applies the OpenType standard ligatures type style to text.	Identifies the OpenType standard ligatures type style applied to text.
OT_DISCRETIONARY_LIGATURES (true | false | none) "none"	Applies the OpenType discretionary type style to text.	Applies the OpenType discretionary type style to text.	Identifies the OpenType discretionary type style applied to text.
OT_ORDINALS (true | false | none) "none"	Applies the OpenType ordinals type style to text.	Applies the OpenType ordinals type style to text.	Identifies the OpenType ordinals type style applied to text.
OT_TITLING_ALTERNATES (true | false | none) "none"	Applies the OpenType titling alternates type style to text.	Applies the OpenType titling alternates type style to text.	Identifies the OpenType titling alternates type style applied to text.
OT_ALL_SMALL_CAPS (true | false | none) "none"	Applies the OpenType all small caps type style to text.	Applies the OpenType all small caps type style to text.	Identifies the OpenType all small caps type style applied to text.
OT_FRACTIONS (true | false | none) "none"	Applies the OpenType fractions type style to text.	Applies the OpenType fractions type style to text.	Identifies the OpenType fractions type style applied to text.
OT_SWASHES (true | false | none) "none"	Applies the OpenType swashes type style to text.	Applies the OpenType swashes type style to text.	Identifies the OpenType swashes type style applied o text.
OT_SMALL_CAPS (true | false | none) "none"	Applies the OpenType small caps type style to text.	Applies the OpenType small caps type style to text.	Identifies the OpenType small caps type style applied to text.
OT_CONTEXTUAL_ALTERNATIVES (true | false | none) "none"	Applies the OpenType contextual alternates type style to text.	Applies the OpenType contextual alternates type style to text.	Identifies the OpenType contextual alternates type style applied to text.
OT_TABULAR_FIGURES (true | false | none) "none"	Applies the OpenType tabular figures type style	Applies the OpenType tabular figures type style	Identifies the OpenType tabular figures type style

	to text.	to text.	applied to text.
OT_PROPORTIONAL_FIGURES (true | false | none) "none"	Applies the OpenType proportional figures type style to text.	Applies the OpenType proportional figures type style to text.	Identifies the OpenType proportional figures type style applied to text.
OT_LINING_FIGURES (true | false | none) "none"	Applies the OpenType lining figures type style to text.	Applies the OpenType lining figures type style to text.	Identifies the OpenType lining figures type style applied to text.
OT_NONE (true | false | none) "none"	Removes OpenType formatting from text.	Removes OpenType formatting from text.	Indicates the OpenType formatting has been removed from text.
OT_SUPERSCRIPT (true | false | none) "none"	Applies the OpenType superscript type style to text.	Applies the OpenType superscript type style to text.	Identifies the OpenType superscript type style applied to text.
OT_SUBSCRIPT (true | false | none) "none"	Applies the OpenType subscript type style to text.	Applies the OpenType subscript type style to text.	Identifies the OpenType subscript type style applied to text.
OT_NUMERATOR (true | false | none) "none"	Applies the OpenType numerator type style to text.	Applies the OpenType numerator type style to text.	Identifies the OpenType numerator type style applied to text.
OT_DENOMINATOR (true | false | none) "none"	Applies the OpenType denominator type style to text.	Applies the OpenType denominator type style to text.	Identifies the OpenType denominator type style applied to text.
OT_OLDSTYLE_FIGURES (true | false | none) "none"	Applies the OpenType old style figures type style to text.	Applies the OpenType old style figures type style to text.	Identifies the OpenType old style figures type style applied to text.
OT_ SCIENTIFIC_INFERIOR_FEATURE (true | false | none) "none"	Replaces lining or old style figures with inferior figures (smaller glyphs which sit lower than the standard baseline, primarily for chemical or mathematical	Replaces lining or old style figures with inferior figures (smaller glyphs which sit lower than the standard baseline, primarily for chemical or mathematical	Replaces lining or old style figures with inferior figures (smaller glyphs which sit lower than the standard baseline, primarily for chemical or mathematical

	notation). May also replace lowercase characters with alphabetic inferiors.	notation). May also replace lowercase characters with alphabetic inferiors.	notation). May also replace lowercase characters with alphabetic inferiors.
OT_ITALICS_FEATURE (true | false | none) "none"	Some fonts (such as Adobe(R) Pro Japanese fonts) have both Roman and Italic forms of some characters in a single font. This feature replaces the Roman glyphs with the corresponding Italic glyphs.	Some fonts (such as Adobe Pro Japanese fonts) have both Roman and Italic forms of some characters in a single font. This feature replaces the Roman glyphs with the corresponding Italic glyphs.	Some fonts (such as Adobe Pro Japanese fonts) have both Roman and Italic forms of some characters in a single font. This feature replaces the Roman glyphs with the corresponding Italic glyphs.
OT_HVKANA_ALTERNATES (true | false | none) "none"	Apply specially designed horizontal or vertical Kana forms that correspond with the story direction (vertical or horizontal).	Apply specially designed horizontal or vertical Kana forms that correspond with the story direction (vertical or horizontal).	Apply specially designed horizontal or vertical Kana forms that correspond with the story direction (vertical or horizontal).
OT_RUBINOTATION_FORMS (true | false | none) "none"	Japanese typesetting often uses smaller kana glyphs, generally in superscripted form, to clarify the meaning of kanji which may be unfamiliar to the reader. These are called ruby, from the old typesetting term for four-point-sized type. This feature identifies glyphs in the font which have been designed for this use, substituting them for the default designs.	Japanese typesetting often uses smaller kana glyphs, generally in superscripted form, to clarify the meaning of kanji which may be unfamiliar to the reader. These are called ruby, from the old typesetting term for four-point-sized type. This feature identifies glyphs in the font which have been designed for this use, substituting them for the default designs.	Japanese typesetting often uses smaller kana glyphs, generally in superscripted form, to clarify the meaning of kanji which may be unfamiliar to the reader. These are called ruby, from the old typesetting term for four-point-sized type. This feature identifies glyphs in the font which have been designed for this use, substituting them for the default designs.

OT_LOCALIZED_FORMS (true | false | none) "none"	Replace default forms of glyphs with localized forms.	Replace default forms of glyphs with localized forms.	Replace default forms of glyphs with localized forms.
OT_ALTERNATE_WIDTHS_NONE (true | false | none) "none"	Apply alternate widths for heights based on story direction (vertical or horizontal).	Apply alternate widths for heights based on story direction (vertical or horizontal).	Apply alternate widths for heights based on story direction (vertical or horizontal).
OT_FULL_WIDTHS (true | false | none) "none"	Replace glyphs set on other em widths with glyphs set on full-em widths.	Replace glyphs set on other em widths with glyphs set on full-em widths.	Replace glyphs set on other em widths with glyphs set on full-em widths.
OT_HALF_WIDTHS (true | false | none) "none"	Replace glyphs set on other em widths with half-em width glyphs.	Replace glyphs set on other em widths with half-em width glyphs.	Replace glyphs set on other em widths with half-em width glyphs.
OT_THIRD_WIDTHS (true | false | none) "none"	Replace glyphs set on other em widths with glyphs set on third-em widths.	Replace glyphs set on other em widths with glyphs set on third-em widths.	Replace glyphs set on other em widths with glyphs set on third-em widths.
OT_QUARTER_WIDTHS (true | false | none) "none"	Replace glyphs set on other em widths with glyphs set on quarter-em widths.	Replace glyphs set on other em widths with glyphs set on quarter-em widths.	Replace glyphs set on other em widths with glyphs set on quarter-em widths.
OT_PROPORTIONAL_WIDTHS (true | false | none) "none"	Fit glyphs to individual, proportional widths.	Fit glyphs to individual, proportional widths.	Fit glyphs to individual, proportional widths.
OT_ALTVERTMETRICS (true | false | none) "none"	Center glyphs inside a full-em height.	Center glyphs inside a full-em height.	Center glyphs inside a full-em height.
OT_ PROPORTIONAL_ALTVERTMETRICS (true | false | none) "none"	Fit glyphs to individual, proportional heights.	Fit glyphs to individual, proportional heights.	Fit glyphs to individual, proportional heights.
OT_ALTERNATE_HALF_METRICS (true | false | none) "none"	Fit full-em height glyphs to half-em heights.	Fit full-em height glyphs to half-em heights.	Fit full-em height glyphs to half-em heights.
OT_ALTERNATE_FORMS_NONE (true | false | none) "none"OT_JIS78FORMS (true | false | none) "none" OT_JIS83FORMS (true | false | none) "none" OT_JIS90FORMS (true | false |	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially

none) "none" OT_JIS04FORMS (true | false | none) "none" OT_SIMPLIFIED_FORMS (true | false | none) "none" OT_TRADITIONAL_FORMS (true | false | none) "none"	designed for some Japanese OpenType fonts.	designed for some Japanese OpenType fonts.	designed for some Japanese OpenType fonts.
LANGUAGE (SwissGerman | SwissGermanReformed | BrazilianPortuguese | Bulgarian | Croatian | Czech | Dutch | Danish | Finnish | French | German | ReformedGerman | Hungarian | Greek | Italian | BokmalNorwegian | Portuguese | Polish | Slovak | Russian | Romanian | Swedish | Turkish | Spanish | USEnglish | Catalan | Estonian | Lithuanian | Latvian | Icelandic | Slovenian | InternationalEnglish | SimplifiedChinese | TraditionalChinese | Japanese | Korean | Ukrainian | NynorskNorwegian | None | none) "none"	Specifies the dictionary preference used for hyphenation.	Specifies the dictionary preference used for hyphenation.	Identifies the dictionary preference used for hyphenation.
SENDING CDATA #IMPLIED	Sending is a character spacing attribute used particularly in East Asian typography, similar to kerning, but applicable as a fixed value over a range of text.	Sending is a character spacing attribute used particularly in East Asian typography, similar to kerning, but applicable as a fixed value over a range of text.	Sending is a character spacing attribute used particularly in East Asian typography, similar to kerning, but applicable as a fixed value over a range of text.
APPLYSENDINGTONONCJK (true | false | none) "none"	Describes whether sending should be applied to both Roman and Chinese/Japanese/Korean glyphs (true) or just to Chinese, Japanese, and Korean Glyphs (false).	Describes whether sending should be applied to both Roman and Chinese/Japanese/Korean glyphs (true) or just to Chinese, Japanese, and Korean Glyphs (false).	Describes whether sending should be applied to both Roman and Chinese/Japanese/Korean glyphs (true) or just to Chinese, Japanese, and Korean Glyphs (false).
UEGGLYPHID CDATA #IMPLIED	Some glyphs, especially in legacy Korean documents, are	Some glyphs, especially in legacy Korean documents, are	Some glyphs, especially in legacy Korean documents, are

	not covered by the Unicode specification. These are referred to as UEG or Unencoded Glyphs. This attribute represents the font glyph ID for such characters that cannot be represented. Note that this is an empty element, as the glyph cannot be represented as text.	not covered by the Unicode specification. These are referred to as UEG or Unencoded Glyphs. This attribute represents the font glyph ID for such characters that cannot be represented. Note that this is an empty element, as the glyph cannot be represented as text.	not covered by the Unicode specification. These are referred to as UEG or Unencoded Glyphs. This attribute represents the font glyph ID for such characters that cannot be represented. Note that this is an empty element, as the glyph cannot be represented as text.
OTVARIANT CDATA #IMPLIED	Specifies which variant to use from among the multiple match found (if any).	Specifies which variant to use from among the multiple match found (if any).	Specifies which variant to use from among the multiple match found (if any).
OTFEATURE CDATA #IMPLIED	Contains the value of the OpenType feature applied on text like AlternateFractions (afrc), AlternateAnnotations, etc.	Contains the value of the OpenType feature applied on text like AlternateFractions (afrc), AlternateAnnotations, etc.	Contains the value of the OpenType feature applied on text like AlternateFractions (afrc), AlternateAnnotations, etc.
SCRIPT (Hira | Hani | Hrkt | Hang | Yiii | Kana |Bopo | none) "none"	Represents the script system used by this RICHTEXT element's content.	Represents the script system used by this RICHTEXT element's content.	Represents the script system used by this RICHTEXT element's content.
HALFWIDTHUPRIGHT (true | false | none) "none">	Specifies whether this character should be presented upright in a vertical story. This is specifically applicable to Roman characters within a vertical story.	Specifies whether this character should be presented upright in a vertical story. This is specifically applicable to Roman characters within a vertical story.	Specifies whether this character should be presented upright in a vertical story. This is specifically applicable to Roman characters within a vertical story.
<!ELEMENT RUBI (RUBITEXT, (RICHTEXT | ANCHOREDBOXREF | HIDDEN)+)>	Specifies a region of base text and the rubi text to include with that	Specifies a region of base text and the rubi text to include with that	Specifies a region of base text and the rubi text to include with that

	text. Note the second and subsequent children of the RUBI element (RICHTEXT | ANCHOREDBOX | HIDDEN)+ declare the base text to which the rubi text is to be applied.	text. Note the second and subsequent children of the RUBI element (RICHTEXT | ANCHOREDBOX | HIDDEN)+ declare the base text to which the rubi text is to be applied.	text. Note the second and subsequent children of the RUBI element (RICHTEXT | ANCHOREDBOX | HIDDEN)+ declare the base text to which the rubi text is to be applied.
<!ELEMENT RUBITEXT (RICHTEXT)>	Specifies the rubi text to be applied to the specified base text. The RUBITEXT element is a container for a RICHTEXT element. All the usual character formatting attributes can be applied to the rubi text through this RICHTEXT element.	Specifies the rubi text to be applied to the specified base text. The RUBITEXT element is a container for a RICHTEXT element. All the usual character formatting attributes can be applied to the rubi text through this RICHTEXT element.	Specifies the rubi text to be applied to the specified base text. The RUBITEXT element is a container for a RICHTEXT element. All the usual character formatting attributes can be applied to the rubi text through this RICHTEXT element.
<!ATTLIST RUBI			
ALIGNMENT (LEFT | TOP | CENTERED | RIGHT | BOTTOM | JUSTIFIED | FORCED | ONETOONE | EQUALSPACE | ONERUBISPACE) "CENTERED"	Controls how non-overhanging rubi text aligns with the base text. For more information, see "Rubi alignment options" in the QuarkXPress documentation.	Controls how non-overhanging rubi text aligns with the base text. For more information, see "Rubi alignment options" in the QuarkXPress documentation.	Controls how non-overhanging rubi text aligns with the base text. For more information, see "Rubi alignment options" in the QuarkXPress documentation.
OVERHANGALIGNMENT (none | LEFT | TOP | CENTERED | RIGHT | BOTTOM | JUSTIFIED | FORCED | ONETOONE | EQUALSPACE) "none"	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."
PLACEMENT (ABOVE | BELOW	This attribute	This attribute	This attribute

| RIGHT | LEFT) "ABOVE"	specifies whether rubi text displays above or below the base text (in a horizontal story) or to the left of or right of the base text (in a vertical story).	specifies whether rubi text displays above or below the base text (in a horizontal story) or to the left of or right of the base text (in a vertical story).	specifies whether rubi text displays above or below the base text (in a horizontal story) or to the left of or right of the base text (in a vertical story).
RELATIVESIZE CDATA "50"	Defines the size of the rubi text compared to the base text.	Defines the size of the rubi text compared to the base text.	Defines the size of the rubi text compared to the base text.
OFFSET CDATA "0"	Use this attribute to control how far the rubi text is offset from the base text.	Use this attribute to control how far the rubi text is offset from the base text.	Use this attribute to control how far the rubi text is offset from the base text.
OVERHANG (none | UNRESTRICTED | HALFRUBI | FULLRUBI | HALFBASE | FULLBASE) "HALFRUBI"	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."	Defines how far the rubi text can overhang base text that is unrelated to the rubi text. For more information, see "Rubi overhang options."
AUTOALIGNATLINEEDGES (true | false) "true"	Automatically aligns rubi text with the border of a text box when the rubi text overhangs the base text and touches the edge of the text box.	Automatically aligns rubi text with the border of a text box when the rubi text overhangs the base text and touches the edge of the text box.	Automatically aligns rubi text with the border of a text box when the rubi text overhangs the base text and touches the edge of the text box.
ANNONATIONS (true | false) "true"	Applicable for OT fonts applied to rubi. If the font supports annotations, then that is applied on the rubi text.	Applicable for OT fonts applied to rubi. If the font supports annotations, then that is applied on the rubi text.	Applicable for OT fonts applied to rubi. If the font supports annotations, then that is applied on the rubi text.
<!ELEMENT ANCHOREDBOXREF (#PCDATA)>	Specifies id of anchored box that is part of the story.	Specifies id of anchored box that is part of the story.	Specifies id of anchored box that is part of the story.
<!ATTLIST ANCHOREDBOXREF			
ALIGNWITHTEXT (ASCENT |	Determines	Determines	Determines

BASELINE) "ASCENT"	whether the top of the anchored box will align with the top of the text (ascent) or the bottom of the text (baseline).	whether the top of the anchored box will align with the top of the text (ascent) or the bottom of the text (baseline).	whether the top of the anchored box will align with the top of the text (ascent) or the bottom of the text (baseline).
OFFSET CDATA #IMPLIED>	Determines the offset when ALIGNWITHTEXT is set to BASELINE. Default is 0.	Determines the offset when ALIGNWITHTEXT is set to BASELINE. Default is 0.	Determines the offset when ALIGNWITHTEXT is set to BASELINE. Default is 0.
<!ELEMENT LINKEDBOX (ID)>	Specifies a linked box and its parent box. To force text to run into the next box in a chain, insert the boxbreak character entity where you want the text to break.	Specifies a linked box and its parent box. To force text to run into the next box in a chain, insert the boxbreak character entity where you want the text to break.	Identifies the point where the text has overflowed the current box and identifies the box where the text continues. Example: <BOX> <ID NAME="Box1"/> <TEXT> <STORY><RICHTEXT>This text is in box 1</RICHTEXT> <RICHTEXT>This text is in box 2</RICHTEXT> <RICHTEXT>This text is in box 3</RICHTEXT> <LINKEDBOX STARTOFFSET="22", ENDOFFSET="42" ><ID NAME="box2"></ID></LINKEDBOX> <LINKEDBOX STARTOFFSET="43", ENDOFFSET="6

			3" ><ID NAME="box3"> </ID> </LINKEDBOX> > </STORY> </BOX>
<!ATTLIST LINKEDBOX			
STARTOFFSET CDATA #IMPLIED	Not applicable.	Not applicable.	Identifies the first character placed in the next box in a chain.
ENDOFFSET CDATA #IMPLIED	Not applicable.	Not applicable.	Specifies the last character placed in the next box in a chain.
<!ELEMENT OVERMATTER (PARAGRAPH | RICHTEXT | ANCHOREDBOXREF | GROUPCHARACTERS | HIDDEN | RUBI)*>	Not applicable.	Not applicable.	Identifies where the current box overflows when there is no subsequence box for text to flow into.
<!ELEMENT PICTURE EMPTY>	Describes the properties of a picture box.	Describes the properties of a picture box.	Describes the properties of a picture box.
<!ATTLIST PICTURE			
FIT (CENTERPICTURE | FITPICTURETOBOX | FITBOXTOPICTURE | FITPICTURETOBOXPRO | NONE) "NONE"	Specifies how a picture should fit within a picture box. CENTERPICTURE = Shifts a picture to the center of the picture box without changing the pictures scale. FITPICTURETOBOX = Scales a picture to fit in its box exactly. The picture cannot be reduced to a size smaller than 10% or increased to a size larger than 1000%, both horizontally and	Specifies how a picture should fit within a picture box. CENTERPICTURE = Shifts a picture to the center of the picture box without changing the pictures scale. FITPICTURETOBOX = Scales a picture to fit in its box exactly. The picture cannot be reduced to a size smaller than 10% or increased to a size larger than 1000%, both horizontally and	Not applicable.

	vertically. FITBOXTOPICTURE = Resizes a box to fit its picture. FITPICTURETOBOXPRO = Scales a picture in a picture box in such a way that the x scale and y scale of a picture remain the same. The picture cannot be reduced to a size smaller than 10% or increased to a size larger than 1000%, both horizontally and vertically.	vertically. FITBOXTOPICTURE = Resizes a box to fit its picture. FITPICTURETOBOXPRO = Scales a picture in a picture box in such a way that the x scale and y scale of a picture remain the same. The picture cannot be reduced to a size smaller than 10% or increased to a size larger than 1000%, both horizontally and vertically.	
SCALEACROSS CDATA #IMPLIED	Specifies the horizontal scale of a picture as an integer percentage from 10 to 1000.	Specifies the horizontal scale of a picture as an integer percentage from 10 to 1000.	Specifies the horizontal scale of a picture as an integer percentage from 10 to 1000.
SCALEDOWN CDATA #IMPLIED	Specifies the vertical scale of a picture as an integer percentage from 10 to 1000.	Specifies the vertical scale of a picture as an integer percentage from 10 to 1000.	Specifies the vertical scale of a picture as an integer percentage from 10 to 1000.
OFFSETACROSS CDATA #IMPLIED	Specifies a horizontal offset for the content of a picture box.	Specifies a horizontal offset for the content of a picture box.	Specifies a horizontal offset for the content of a picture box.
OFFSETDOWN CDATA #IMPLIED	Specifies a vertical offset for the content of a picture box.	Specifies a vertical offset for the content of a picture box.	Specifies a vertical offset for the content of a picture box.
ANGLE CDATA #IMPLIED	Specifies a rotation angle for a picture as a floating-point value between -360 degrees and 360 degrees.	Specifies a rotation angle for a picture as a floating-point value between -360 degrees and 360 degrees.	Specifies a rotation angle for a picture as a floating-point value between -360 degrees and 360 degrees.
SKEW CDATA #IMPLIED	Specifies a skew angle for a picture as a floating-point	Specifies a skew angle for a picture as a floating-point	Specifies a skew angle for a picture as a floating-point

	value from -75 degrees to 75 degrees.	value from -75 degrees to 75 degrees.	value from -75 degrees to 75 degrees.
PICCOLOR CDATA #IMPLIED	Identifies a color to be applied to a grayscale picture. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies a color to be applied to a grayscale picture. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies a color applied to a grayscale picture. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a grayscale picture, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of a picture, specified as an integer percentage from 0 to 100.	Specifies the opacity of a picture, specified as an integer percentage from 0 to 100.	Specifies the opacity of a picture, specified as an integer percentage from 0 to 100.
FLIPVERTICAL (true | false | none) "none"	Flips a picture vertically.	Flips a picture vertically. If a picture is already flipped vertically, then this flips the picture back.	Indicates whether a picture has been flipped vertically.
FLIPHORIZONTAL (true | false | none) "none"	Flips a picture horizontally.	Flips a picture horizontally. If a picture is already flipped horizontally, then this flips the picture back.	Indicates whether a picture has been flipped horizontally.
SUPRESSPICT (true | false) "false"	Prevents a picture	Prevents a picture	Prevents a picture

	from being included in output.	from being included in output.	from being included in output.
FULLRES (true | false | none) "none"	Causes imported pictures to display at full resolution in QuarkXPress if the picture files are available.	Causes imported pictures to display at full resolution in QuarkXPress if the picture files are available.	Causes imported pictures to display at full resolution in QuarkXPress if the picture files are available.
MASK CDATA #IMPLIED>	Identifies an alpha channel in the picture file to be used to mask the picture file.	Identifies an alpha channel in the picture file to be used to mask the picture file.	Identifies an alpha channel in the picture file that is being used to mask the picture file.
<!ELEMENT CLIPPING EMPTY>	Describes a clipping path.	Describes a clipping path.	Describes a clipping path.
<!ATTLIST CLIPPING			
TYPE (ITEM | EMBEDDEDPATH | ALPHACHANNEL | NONWHITEAREAS | PICTUREBOUNDS) "ITEM"	Specifies the type of clipping applied to a picture item: ITEM = Runs along the edges of the item. EMBEDDEDPATH = Runs along a path embedded in the picture file. ALPHACHANNEL = Runs along an alpha channel embedded in the picture file. NONWHITEAREAS = Runs along a path based on the dark and light areas of the picture file. See the THRESHOLD attribute. PICTUREBOUNDS = Runs along the rectangular canvas area of the picture, regardless of the size and shape of the picture box.	Specifies the type of clipping applied to a picture item: ITEM = Runs along the edges of the item. EMBEDDEDPATH = Runs along a path embedded in the picture file. ALPHACHANNEL = Runs along an alpha channel embedded in the picture file. NONWHITEAREAS = Runs along a path based on the dark and light areas of the picture file. See the THRESHOLD attribute. PICTUREBOUNDS = Runs along the rectangular canvas area of the picture, regardless of the size and shape of the picture box.	Specifies the type of clipping applied to a picture item: ITEM = Runs along the edges of the item. EMBEDDEDPATH = Runs along a path embedded in the picture file. ALPHACHANNEL = Runs along an alpha channel embedded in the picture file. NONWHITEAREAS = Runs along a path based on the dark and light areas of the picture file. See the THRESHOLD attribute. PICTUREBOUNDS = Runs along the rectangular canvas area of the picture, regardless of the size and shape of the picture box.
TOP CDATA #IMPLIED	Valid when	Valid when	Valid when

	CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the clipping path by the specified number of points (positive=up, negative=down).	CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the clipping path by the specified number of points (positive=up, negative=down).	CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the clipping path by the specified number of points (positive=up, negative=down).
RIGHT CDATA #IMPLIED	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the clipping path by the specified number of points (positive=right, negative=left).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the clipping path by the specified number of points (positive=right, negative=left).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the clipping path by the specified number of points (positive=right, negative=left).
LEFT CDATA #IMPLIED	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the clipping path by the specified number of points (positive=left, negative=right).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the clipping path by the specified number of points (positive=left, negative=right).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the clipping path by the specified number of points (positive=left, negative=right).
BOTTOM CDATA #IMPLIED	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the clipping path by the specified number of points (positive=down, negative=up).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the clipping path by the specified number of points (positive=down, negative=up).	Valid when CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the clipping path by the specified number of points (positive=down, negative=up).
PATHNAME CDATA #IMPLIED	Identifies a path embedded in a picture for use as the clipping path.	Identifies a path embedded in a picture for use as the clipping path.	Identifies a path embedded in a picture for use as the clipping path.
OUTSET CDATA #IMPLIED	Valid when CLIPPING@TYPE =	Valid when CLIPPING@TYPE =	Valid when CLIPPING@TYPE =

	EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.	EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.	EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.
NOISE CDATA #IMPLIED	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a clipping path.
THRESHOLD CDATA #IMPLIED	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a clipping path.
SMOOTHNESS CDATA #IMPLIED	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created clipping path.	Valid when CLIPPING@TYPE = ALPHACHANNEL or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created clipping path.
OUTSIDEONLY (true | false | none)	Valid when	Valid when	Valid when

"none"	CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer edges of the clipping path should be used.	CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer edges of the clipping path should be used.	CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer edges of the clipping path should be used.
RESTRICTTOBOX (true | false | none) "none"	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the clipping path is restricted to the inside of the box.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the clipping path is restricted to the inside of the box.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the clipping path is restricted to the inside of the box.
INVERT (true | false | none) "none"	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the clipping path.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the clipping path.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the clipping path.
EDITED (true | false | none) "none">	Not applicable.	Not applicable.	Indicates whether the clipping path has been manually edited in QuarkXPress.
<!ELEMENT SPLINESHAPE (CONTOURS)>	Specifies a complex spline shape in QuarkXPress (i.e., the curve of a Bezier box or Bezier text path).	Specifies a complex spline shape in QuarkXPress (i.e., the curve of a Bezier box or Bezier text path).	Specifies a complex spline shape in QuarkXPress (i.e., the curve of a Bezier box or Bezier text path).
<!ATTLIST SPLINESHAPE			
RECTSHAPE (true | false) "false"	Specifies whether	Specifies whether	Specifies whether

	the shape is a pure rectangle.	the shape is a pure rectangle.	the shape is a pure rectangle.
INVERTEDSHAPE (true | false) "false"	Specifies whether the shape encodes the inverse of its area ("inside out").	Specifies whether the shape encodes the inverse of its area ("inside out").	Specifies whether the shape encodes the inverse of its area ("inside out").
HASSPLINES (true | false) "false"	Specifies whether any of the contours in the shape contains a spline.	Specifies whether any of the contours in the shape contains a spline.	Specifies whether any of the contours in the shape contains a spline.
HASHOLES (true | false) "false"	Specifies whether any of the contours is inside another.	Specifies whether any of the contours is inside another.	Specifies whether any of the contours is inside another.
NEWFORMAT (true | false) "false"	Specifies whether incompatible with "old" (3.31 and below) shapes.	Specifies whether incompatible with "old" (3.31 and below) shapes.	Specifies whether incompatible with "old" (3.31 and below) shapes.
MORETHANONETOPLEVELCONTOUR (true | false) "false"	Specifies whether there is more than one top-level contour.	Specifies whether there is more than one top-level contour.	Specifies whether there is more than one top-level contour.
CLOSEDSHAPE (true | false) "false"	Specifies whether all its contours are closed. (Polylines might not be.)	Specifies whether all its contours are closed. (Polylines might not be.)	Specifies whether all its contours are closed. (Polylines might not be.)
WELLFORMED (true | false) "false"	Specifies whether the shape does not intersect itself other than at the vertex.	Specifies whether the shape does not intersect itself other than at the vertex.	Specifies whether the shape does not intersect itself other than at the vertex.
TAGSALLOCATED (true | false) "false"	Specifies whether the vertex tags are set correctly.	Specifies whether the vertex tags are set correctly.	Specifies whether the vertex tags are set correctly.
INCOMPLETE (true | false) "false"	Specifies whether shape is associated with UNFINISHED box.	Specifies whether shape is associated with UNFINISHED box.	Specifies whether shape is associated with UNFINISHED box.
VERTSELECTED (true | false) "false">	Specifies whether one or more verts are selected.	Specifies whether one or more verts are selected.	Specifies whether one or more verts are selected.
<!ELEMENT CONTOURS (CONTOUR+)>	A group of contours which, combined, make a spline shape.	A group of contours which, combined, make a spline shape.	A group of contours which, combined, make a spline shape.
<!ELEMENT CONTOUR (VERTICES)>	A single contour within a spline	A single contour within a spline	A single contour within a spline

	shape.	shape.	shape.
<!ATTLIST CONTOUR			
CURVEDEDGES (true | false) "false"	Specifies whether there are any curved edges in the contour.	Specifies whether there are any curved edges in the contour.	Specifies whether there are any curved edges in the contour.
RECTCONTOUR (true | false) "false"	Specifies whether this contour is rectangular.	Specifies whether this contour is rectangular.	Specifies whether this contour is rectangular.
INVERTEDCONTOUR (true | false) "false"	Specifies whether the points describe a hole instead of an outside contour.	Specifies whether the points describe a hole instead of an outside contour.	Specifies whether the points describe a hole instead of an outside contour.
TOPLEVEL (true | false) "false"	Specifies whether the contour has no containing contours.	Specifies whether the contour has no containing contours.	Specifies whether the contour has no containing contours.
SELFINTERSECTED (true | false) "false"	Specifies whether the contour intersects itself.	Specifies whether the contour intersects itself.	Specifies whether the contour intersects itself.
POLYCONTOUR (true | false) "false"	Specifies whether this is a polycontour (as opposed to a spline contour).	Specifies whether this is a polycontour (as opposed to a spline contour).	Specifies whether this is a polycontour (as opposed to a spline contour).
VERTEXTAGEXISTS (true | false) "false">	Specifies whether there are vertex tags associated with the contour.	Specifies whether there are vertex tags associated with the contour.	Specifies whether there are vertex tags associated with the contour.
<!ELEMENT VERTICES (VERTEX+)>	A collection of vertexes which, combined, make up a contour.	A collection of vertexes which, combined, make up a contour.	A collection of vertexes which, combined, make up a contour.
<!ELEMENT VERTEX (LEFTCONTROLPOINT?, VERTEXPOINT, RIGHTCONTROLPOINT?)>	A single vertext (i.e. Line segment) in a bezier curve.	A single vertext (i.e. Line segment) in a bezier curve.	A single vertext (i.e. Line segment) in a bezier curve.
<!ATTLIST VERTEX			
SMOOTHVERTEX (true | false) "false"	Specifies whether the given vertex is "straight" - i.e.. C1 continuous.	Specifies whether the given vertex is "straight" - i.e.. C1 continuous.	Specifies whether the given vertex is "straight" - i.e.. C1 continuous.
STRAIGHTEDGE (true | false) "false"	Specifies whether the following edge is "straight".	Specifies whether the following edge is "straight".	Specifies whether the following edge is "straight".
SYMMVERTEX (true | false) "false"	Specifies whether the given vertex is also symmetrical - i.e., C2	Specifies whether the given vertex is also symmetrical - i.e., C2	Specifies whether the given vertex is also symmetrical - i.e., C2

	continuous.	continuous.	continuous.
CUSPVERTEX (true | false) "false"	Specifies whether the vertex is not smooth or symmetric.	Specifies whether the vertex is not smooth or symmetric.	Specifies whether the vertex is not smooth or symmetric.
TWISTED (true | false) "false"	Specifies whether the following (splined) edge intersects itself.	Specifies whether the following (splined) edge intersects itself.	Specifies whether the following (splined) edge intersects itself.
VERTEXSELECTED (true | false) "false">	Specifies whether the given vertex is selected.	Specifies whether the given vertex is selected.	Specifies whether the given vertex is selected.
<!ELEMENT LEFTCONTROLPOINT EMPTY>	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.
<!ELEMENT VERTEXPOINT EMPTY>	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the

	shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.
<!ELEMENT RIGHTCONTROLPOINT EMPTY>	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.	Each point on a curve is described by three geometric positions: the x,y coordinate of the vertex point (this coordinate is relative to the bounding geometry of the shape, not the page), and the left and right control handles – as you would see onscreen in the QuarkXPress user environment. For more information on drawing and manipulating bezier curves, please see A Guide to QuarkXPress.
<!ATTLIST LEFTCONTROLPOINT			
X CDATA #IMPLIED	X coordinate of LEFTCONTROLPOINT.	X coordinate of LEFTCONTROLPOINT.	X coordinate of LEFTCONTROLPOINT.
Y CDATA #IMPLIED>	Y coordinate of LEFTCONTROLPOINT.	Y coordinate of LEFTCONTROLPOINT.	Y coordinate of LEFTCONTROLPOINT.

<!ATTLIST VERTEXPOINT			
X CDATA #IMPLIED	X coordinate of VERTEXPOINT.	X coordinate of VERTEXPOINT.	X coordinate of VERTEXPOINT.
Y CDATA #IMPLIED	Y coordinate of VERTEXPOINT.	Y coordinate of VERTEXPOINT.	Y coordinate of VERTEXPOINT.
TAG CDATA #IMPLIED>	Specifies vertex of spline box as one of the following: Symmetrical, Smooth, Corner, Straight segment, Curved segment.	Specifies vertex of spline box as one of the following: Symmetrical, Smooth, Corner, Straight segment, Curved segment.	Specifies vertex of spline box as one of the following: Symmetrical, Smooth, Corner, Straight segment, Curved segment.
<!ATTLIST RIGHTCONTROLPOINT			
X CDATA #IMPLIED	X coordinate of RIGHTCONTROLPOINT.	X coordinate of RIGHTCONTROLPOINT.	X coordinate of RIGHTCONTROLPOINT.
Y CDATA #IMPLIED>	Y coordinate of RIGHTCONTROLPOINT.	Y coordinate of RIGHTCONTROLPOINT.	Y coordinate of RIGHTCONTROLPOINT.
<!ELEMENT GEOMETRY ((POSITION | MOVEUP | MOVEDOWN | MOVELEFT | MOVERIGHT | GROWACROSS | GROWDOWN | SHRINKACROSS | SHRINKDOWN | ALLOWBOXONTOPASTEBOARD | ALLOWBOXOFFPAGE | STACKINGORDER | SUPPRESSOUTPUT | RUNAROUND | LINESTYLE | SPLINESHAPE)*)>	Describes the geometric characteristics of a box or line.	Describes the geometric characteristics of a box or line, and allows you to change its position and size.	Describes the geometric characteristics of a box or line.
<!ATTLIST GEOMETRY			
SHAPE (SH_RECT | SH_CONVEXRRECT | SH_CONCAVERRECT | SH_STRAIGHTRRECT | SH_OVAL | SH_LINE | SH_ORTHLINE | SH_SPLINEBOX | SH_NONE | SH_ORTHPOLYLINE | SH_SPLINELINE | SH_ORTHPOLYBOX | SH_USER) "SH_RECT"	Describes the shape of a box or line. SH_RECT = Rectangular box SH_CONVEXRRECT = Box with convex corners SH_CONCAVERRECT = Box with concave corners SH_STRAIGHTRRECT = Box with beveled corners SH_OVAL =	Describes the shape of a box or line. SH_RECT = Rectangular box SH_CONVEXRRECT = Box with convex corners SH_CONCAVERRECT = Box with concave corners SH_STRAIGHTRRECT = Box with beveled corners SH_OVAL =	Describes the shape of a box or line. SH_RECT = Rectangular box SH_CONVEXRRECT = Box with convex corners SH_CONCAVERRECT = Box with concave corners SH_STRAIGHTRRECT = Box with beveled corners SH_OVAL =

	Elliptical box SH_LINE = Line SH_ORTHLINE = Orthogonal line (restricted to 45-degree angles) SH_SPLINEBOX = Freehand shape SH_NONE = Available to define in XDK API SH_ORTHPOLYLINE = Can be defined in XDK SH_SPLINELINE = Freehand line SH_ORTHPOLYBOX = Available to define in XDK API **Note:** You cannot specify PICTURE content for a box if its SHAPE attribute is set to SH_LINE.	Elliptical box SH_LINE = Line SH_ORTHLINE = Orthogonal line (restricted to 45-degree angles) **S**H_SPLINEBOX = Freehand shape SH_NONE = Available to define in XDK API SH_ORTHPOLYLINE = Can be defined in XDK SH_SPLINELINE = Freehand line SH_ORTHPOLYBOX = Available to define in XDK API **Note:** You cannot specify PICTURE content for a box if its SHAPE attribute is set to SH_LINE.	Elliptical box SH_LINE = Line SH_ORTHLINE = Orthogonal line (restricted to 90-degree angles) SH_SPLINEBOX = Freehand shape SH_NONE = Available to define in XDK API SH_ORTHPOLYLINE = Can be defined in XDK SH_SPLINELINE = Freehand line SH_ORTHPOLYBOX = Available to define in XDK API SH_USER = Available to define in XDK API
PAGE CDATA #IMPLIED	Specifies the number of the page where the upper left corner of this box or line should be created. **Note:** This attribute determines where to create a box or line, regardless of which PAGE element the box or line occurs within.	Specifies the number of the page where the upper left corner of this box or line is located. **Note:** This attribute determines where a box or line is, regardless of which PAGE element the box or line occurs within.	Specifies the number of the page where the upper left corner of this box or line is located. **Note:** This attribute determines where a box or line is, regardless of which PAGE element the box or line occurs within.
ANGLE CDATA #IMPLIED	Specifies a rotation angle for a box or line as a floating-point	Specifies a rotation angle for a box or line as a floating-point	Specifies a rotation angle for a box or line as a floating-point

	value between -360 degrees and 360 degrees.	value between -360 degrees and 360 degrees.	value between -360 degrees and 360 degrees.
LAYER CDATA #IMPLIED>	Identifies the layer where a box or line should be created.	Identifies the layer where a box or line is located.	Identifies the layer that a box resides on. **Note:** Boxes on non-displayed layers are not included. This means you can use the LAYER URL parameter as a filter when a layout contains multiple layers.
<!ELEMENT POSITION (TOP, LEFT, BOTTOM, RIGHT)>	Specifies the absolute position of a box or line on the page, using coordinates measured in points from the upper-left corner of the page.	Specifies the absolute position of a box or line on the page, using coordinates measured in points from the upper-left corner of the page.	Specifies the absolute position of a box or line on the page, using coordinates measured in points from the upper-left corner of the page.
<!ELEMENT MOVEUP (#PCDATA)>	Not applicable.	Moves a box up by the specified number of points. **Note:** You can move a box or line onto another page.	Not applicable.
<!ELEMENT MOVEDOWN (#PCDATA)>	Not applicable.	Moves a box down by the specified number of points. **Note:** You can move a box or line onto another page.	Not applicable.
<!ELEMENT MOVELEFT (#PCDATA)>	Not applicable.	Moves a box to the left by the specified number of points. **Note:** You can move a box or line onto another page.	Not applicable.
<!ELEMENT MOVERIGHT (#PCDATA)>	Not applicable.	Moves a box to the right by the	Not applicable.

		specified number of points. **Note:** You can move a box or line onto another page.	
<!ELEMENT GROWACROSS (#PCDATA)>	Not applicable.	Expands a box horizontally to the right by the specified number of points. **Note:** A box can be expanded on the same page or on other spreads and pages.	Not applicable.
<!ELEMENT GROWDOWN (#PCDATA)>	Not applicable.	Expands a box vertically toward the bottom of the page by the specified number of points. **Note:** A box can be expanded on the same page or on other spreads and pages.	Not applicable.
<!ELEMENT SHRINKACROSS (#PCDATA)>	Not applicable.	Shrinks a box horizontally to the left by the specified number of points. **Note:** A box can shrink on the same page or on other spreads and pages.	Not applicable.
<!ELEMENT SHRINKDOWN (#PCDATA)>	Not applicable.	Shrinks a box vertically toward the top of the page by the specified number of points. **Note:** A box can shrink on the same page or on other spreads and pages.	Not applicable.
<!ELEMENT	Not applicable.	Specifies whether	Not applicable.

ALLOWBOXONTOPASTEBOARD (#PCDATA)>		a box is allowed to be moved partially off of a page and onto the pasteboard by, for example, a MOVERIGHT element. Only accepts true or false values; default value is true.	
<!ELEMENT ALLOWBOXOFFPAGE (#PCDATA)>	Not applicable.	Specifies whether a box is allowed to be moved completely off of a page and onto the pasteboard by, for example, a MOVERIGHT element. Only accepts true or false values; default value is true.	Not applicable.
<!ELEMENT STACKINGORDER (#PCDATA)>	Lets you control whether a box or line is in front of or behind other items on the page. Only accepts SENDBACKWARD, SENDTOBACK, BRINGFORWARD, BRINGTOFRONOT.	Lets you control whether a box or line is in front of or behind other items on the page. Only accepts SENDBACKWARD, SENDTOBACK, BRINGFORWARD, BRINGTOFRONOT.	Not applicable.
<!ELEMENT SUPPRESSOUTPUT (#PCDATA)>	Specifies whether a box is included in output. A *true* value does not include the box; a *false* value includes the box.	Specifies whether a box is included in output. A *true* value does not include the box; a *false* value includes the box.	Specifies whether a box is included in output. A *true* value does not include the box; a *false* value includes the box.
<!ELEMENT TOP (#PCDATA)>	The distance between the box or lines top edge and the top of the	The distance between the box or lines top edge and the top of the	The distance between the box or lines top edge and the top of the

	page, in points.	page, in points.	page, in points.
<!ELEMENT LEFT (#PCDATA)>	The distance between the box or lines left edge and the left edge of the page, in points.	The distance between the box or lines left edge and the left edge of the page, in points.	The distance between the box or lines left edge and the left edge of the page, in points.
<!ELEMENT BOTTOM (#PCDATA)>	The distance between the box or lines bottom edge and the bottom of the page, in points.	The distance between the box or lines bottom edge and the bottom of the page, in points.	The distance between the box or lines bottom edge and the bottom of the page, in points.
<!ELEMENT RIGHT (#PCDATA)>	The distance between the box or lines right edge and the right edge of the page, in points.	The distance between the box or lines right edge and the right edge of the page, in points.	The distance between the box or lines right edge and the right edge of the page, in points.
<!ELEMENT RUNAROUND EMPTY>	Describes a runaround applied to a box or line.	Describes a runaround applied to a box or line.	Describes a runaround applied to a box or line.
<!ATTLIST RUNAROUND			
TYPE (NONE | ITEM | EMBEDDEDPATH | ALPHACHANNEL | NONWHITEAREAS | PICTUREBOUNDS | SAMEASCLIPPING | AUTOIMAGE | MANUAL) "NONE"	Specifies the type of runaround applied to a box or line: NONE = Text runs behind the box or line. ITEM = Text runs around the edges of the box or line. EMBEDDEDPATH = Text runs around a path embedded in the picture file. ALPHACHANNEL = Text runs around an alpha channel embedded in the picture file. NONWHITEAREAS = Text runs around a path based on the dark and light areas of the picture file.	Specifies the type of runaround applied to a box or line: NONE = Text runs behind the box or line. ITEM = Text runs around the edges of the box or line. EMBEDDEDPATH = Text runs around a path embedded in the picture file. ALPHACHANNEL = Text runs around an alpha channel embedded in the picture file. NONWHITEAREAS = Text runs around a path based on the dark and light areas of the picture file.	Specifies the type of runaround applied to a box or line: NONE = Text runs behind the box or line. ITEM = Text runs around the edges of the box or line. EMBEDDEDPATH = Text runs around a path embedded in the picture file. ALPHACHANNEL = Text runs around an alpha channel embedded in the picture file. NONWHITEAREAS = Text runs around a path based on the dark and light areas of the picture file.

	See the THRESHOLD attribute. PICTUREBOUNDS = Text runs around the rectangular canvas area of the picture, regardless of the size and shape of the picture box. SAMEASCLIPPING = Text runs around the pictures clipping path, if any. AUTOIMAGE = Text runs around a clipping path created based on the dark and light areas in the picture file. See the THRESHOLD attribute.	See the THRESHOLD attribute. PICTUREBOUNDS = Text runs around the rectangular canvas area of the picture, regardless of the size and shape of the picture box. SAMEASCLIPPING = Text runs around the pictures clipping path, if any. AUTOIMAGE = Text runs around a clipping path created based on the dark and light areas in the picture file. See the THRESHOLD attribute.	See the THRESHOLD attribute. PICTUREBOUNDS = Text runs around the rectangular canvas area of the picture, regardless of the size and shape of the picture box. SAMEASCLIPPING = Text runs around the pictures clipping path, if any. AUTOIMAGE = Text runs around a clipping path created based on the dark and light areas in the picture file. See the THRESHOLD attribute.
TOP CDATA #IMPLIED	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the runaround by the specified number of points (positive=up, negative=down).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the runaround by the specified number of points (positive=up, negative=down).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the runaround by the specified number of points (positive=up, negative=down).
RIGHT CDATA #IMPLIED	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the runaround by the specified number of points (positive=right,	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the runaround by the specified number of points (positive=right,	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the runaround by the specified number of points (positive=right,

	negative=left).	negative=left).	negative=left).
LEFT CDATA #IMPLIED	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the runaround by the specified number of points (positive=left, negative=right).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the runaround by the specified number of points (positive=left, negative=right).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the left edge of the runaround by the specified number of points (positive=left, negative=right).
BOTTOM CDATA #IMPLIED	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the runaround by the specified number of points (positive=down, negative=up).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the runaround by the specified number of points (positive=down, negative=up).	Valid when RUNAROUND @TYPE = ITEM or PICTUREBOUNDS. Moves the bottom edge of the runaround by the specified number of points (positive=down, negative=up).
PATHNAME CDATA #IMPLIED	Identifies a clipping path embedded in a picture for use as the runaround path.	Identifies a clipping path embedded in a picture for use as the runaround path.	Identifies a clipping path embedded in a picture for use as the runaround path.
OUTSET CDATA #IMPLIED	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, NONWHITEAREAS, or SAMEASCLIPPING. Specifies a single outset or inset integer value in points to be used on all sides.	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, NONWHITEAREAS, or SAMEASCLIPPING. Specifies a single outset or inset integer value in points to be used on all sides.	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, NONWHITEAREAS, or SAMEASCLIPPING. Specifies a single outset or inset integer value in points to be used on all sides.
NOISE CDATA #IMPLIED	Valid when RUNAROUND @TYPE =	Valid when RUNAROUND @TYPE =	Valid when RUNAROUND @TYPE =

	AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a runaround path.	AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a runaround path.	AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this number of points should be ignored when creating a runaround path.
THRESHOLD CDATA #IMPLIED	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a runaround path.	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a runaround path.	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the maximum integer percentage of darkness that should be considered white when creating a runaround path.
SMOOTHNESS CDATA #IMPLIED	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created runaround path.	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created runaround path.	Valid when RUNAROUND @TYPE = AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies the smoothness, in points, of an automatically created runaround path.
OUTSIDEONLY (true | false | none) "none"	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates that only the outer

	edges of the runaround path should be used.	edges of the runaround path should be used.	edges of the runaround path is used.
RESTRICTTOBOX (true | false | none) "none"	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the runaround path is restricted to the inside of the box.	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the runaround path is restricted to the inside of the box.	Valid when RUNAROUND @TYPE = AUTOIMAGE, EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Indicates whether the runaround path is restricted to the inside of the box.
INVERT (true | false | none) "none"	Valid when RUNAROUND @TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the runaround path.	Valid when RUNAROUND @TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the runaround path.	Valid when RUNAROUND @TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Reverses the shape of the runaround path.
EDITED (true | false | none) "none"	Not applicable.	Not applicable.	Indicates whether the runaround path has been manually edited in QuarkXPress.
<!ELEMENT LAYER (ID, RGBCOLOR)>	Describes a layer.	Describes a layer.	Describes a layer.
<!ATTLIST LAYER			
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the indicated layer. Note that when you delete a layer, all items on the layer are deleted.	Not applicable.
VISIBLE (true | false | none) "none"	Specifies whether a layer is visible. **Note:** In QuarkXPress, this parameter overrides the	Specifies whether a layer is visible. **Note:** In QuarkXPress, this parameter overrides the	Specifies whether a layer should be visible. **Note:** In QuarkXPress, this parameter

	Visible setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	**Visible** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	overrides the **Visible** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).
KEEPRUNAROUND (true | false | none) "none"	Specifies whether text on visible layers runs around text on hidden layers. **Note:** In QuarkXPress, this parameter overrides the **Keep Runaround** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether text on visible layers runs around text on hidden layers. **Note:** In QuarkXPress, this parameter overrides the Keep Runaround setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether text on visible layers runs around text on hidden layers. **Note:** In QuarkXPress, this parameter overrides the **Keep Runaround** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).
LOCKED (true | false | none) "none"	Specifies whether a layer is locked. **Note:** In QuarkXPress, this parameter overrides the **Locked** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether a layer is locked. **Note:** In QuarkXPress, this parameter overrides the **Locked** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether a layer is locked. **Note:** In QuarkXPress, this parameter overrides the **Locked** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).
SUPPRESS (true | false | none) "none">	Specifies whether output of a layer is suppressed. **Note:** In QuarkXPress, this parameter overrides the **Suppress Output** setting in the **Layers** pane of the **Preferences** dialog box (	Specifies whether output of a layer is suppressed. **Note:** In QuarkXPress, this parameter overrides the **Suppress Output** setting in the **Layers** pane of the **Preferences** dialog box (	Specifies whether output of a layer is suppressed. **Note:** In QuarkXPress, this parameter overrides the **Suppress Output** setting in the **Layers** pane of the **Preferences** dialog box (

	QuarkXPress/Edit menu).	**QuarkXPress/Edit** menu).	**QuarkXPress/Edit** menu).
<!ELEMENT RGBCOLOR EMPTY>	Describes an RGB color that can be associated with a layer, as displayed in the **Layers** palette in QuarkXPress.	Describes an RGB color that can be associated with a layer, as displayed in the **Layers** palette in QuarkXPress.	Describes an RGB color that can be associated with a layer, as displayed in the **Layers** palette in QuarkXPress.
<!ATTLIST RGBCOLOR			
RED CDATA #IMPLIED	An integer from 0 to 255, indicating the red component of an RGB color.	An integer from 0 to 255, indicating the red component of an RGB color.	An integer from 0 to 255, indicating the red component of an RGB color.
GREEN CDATA #IMPLIED	An integer from 0 to 255, indicating the green component of an RGB color.	An integer from 0 to 255, indicating the green component of an RGB color.	An integer from 0 to 255, indicating the green component of an RGB color.
BLUE CDATA #IMPLIED>	An integer from 0 to 255, indicating the blue component of an RGB color.	An integer from 0 to 255, indicating the blue component of an RGB color.	An integer from 0 to 255, indicating the blue component of an RGB color.
<!ELEMENT LINESTYLE EMPTY>	Describes a Dashes & Stripes style that can be applied to lines or frames.	Describes a Dashes & Stripes style that can be applied to lines or frames.	Describes a Dashes & Stripes style that can be applied to lines or frames.
<!ATTLIST LINESTYLE			
ARROWHEADS (PLAINLINE | LEFTARROW | RIGHTARROW | LEFTFARROW | RIGHTFARROW | DOUBLEARROW) "PLAINLINE">	Specifies whether a line should have arrows on its ends: PLAINLINE = No arrows LEFTARROW = Arrow head on left end RIGHTARROW = Arrow head on right end LEFTFARROW = Arrow head on left end, arrow tail on right end RIGHTFARROW = Arrow head on right end,	Specifies whether a line should have arrows on its ends: PLAINLINE = No arrows LEFTARROW = Arrow head on left end RIGHTARROW = Arrow head on right end LEFTFARROW = Arrow head on left end, arrow tail on right end RIGHTFARROW = Arrow head on right end,	Specifies whether a line has arrows on its ends: PLAINLINE = No arrows LEFTARROW = Arrow head on left end RIGHTARROW = Arrow head on right end LEFTFARROW = Arrow head on left end, arrow tail on right end RIGHTFARROW = Arrow head on right end, arrow tail on left

	arrow tail on left end DOUBLEARROW = Arrow heads on both ends	arrow tail on left end DOUBLEARROW = Arrow heads on both ends	end DOUBLEARROW = Arrow heads on both ends
<!ELEMENT CONTENTPH ((CONTENT), METADATA?)>	Placeholder that will contain either text or picture data from a linked file.	Placeholder that will contain either text or picture data from a linked file.	Placeholder that will contain either text or picture data from a linked file.
<!ATTLIST CONTENTPH			
NAME CDATA #REQUIRED	The name of the content placeholder (CONTENTPH).	The name of the content placeholder (CONTENTPH).	The name of the content placeholder (CONTENTPH).
OWNER (1347639377) "1347639377">	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXTID (1347639377). All placeholders created through Modifier should use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute.	The XTensions ID of the XTensions that created this placeholder. The default XT ID is PlaceHolderSXTID (1347639377). All placeholders created through Modifier should use this ID. This ID is assigned by default by the DTD, so there is no need to specify this manually. DTD validation will add this attribute.	The XTensions ID of the XTensions that created this placeholder.
<!ELEMENT CONTENT (#PCDATA)>	Specifies the path of an image or text file that you want to associate with the parent box. The CONTENT element also supports relative paths for images or text files.	Specifies the path of an image or text file that you want to import into the parent box. **Note:** If you use the CONTENT element to import text, the imported text is appended to the end of any existing text in the box.	Specifies the path of the image or text file (if any) associated with the parent box.
<!ATTLIST CONTENT			
CONVERTQUOTES (true | false)	If true, straight	If true, straight	Not applicable.

"true"	quotation marks in an imported text file are converted to typesetter's quotation marks and double hyphens are converted to em dashes.	quotation marks in an imported text file are converted to typesetter's quotation marks and double hyphens are converted to em dashes.	
INCLUDESTYLESHEETS (true | false) "true"	If true, any style sheets in an imported text file are added to the QuarkXPress project.	If true, any style sheets in an imported text file or document are added to the QuarkXPress project.	Not applicable.
FONTNAME CDATA #IMPLIED>	Specifies a font to apply to imported text.	Specifies a font to apply to imported text.	Not applicable.
<!ELEMENT SHADOW (EMPTY)>	Describes an automatic drop shadow.	Describes an automatic drop shadow.	Describes an automatic drop shadow.
<!ATTLIST SHADOW			
COLOR CDATA #REQUIRED	Identifies the color of a drop shadow. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a drop shadow. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file, defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color of a drop shadow. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #REQUIRED	Specifies the shade of the color applied to a drop shadow, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a drop shadow, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a drop shadow, as an integer percentage from 0 to 100.
OPACITY CDATA #REQUIRED	Specifies the	Specifies the	Specifies the

	opacity of a drop shadow, specified as an integer percentage from 0 to 100.	opacity of a drop shadow, specified as an integer percentage from 0 to 100.	opacity of a drop shadow, specified as an integer percentage from 0 to 100.
ANGLE CDATA #REQUIRED	Specifies an angle in degrees for a drop shadow. Should be a floating point value between -180 and 180.	Specifies an angle in degrees for a drop shadow. Should be a floating point value between -180 and 180.	Specifies an angle in degrees for a drop shadow. Should be a floating point value between -180 and 180.
DISTANCE CDATA #REQUIRED	Specifies the distance in points from the edge of an item to the edge of the items drop shadow as a floating point value.	Specifies the distance in points from the edge of an item to the edge of the items drop shadow as a floating point value.	Specifies the distance in points from the edge of an item to the edge of the items drop shadow as a floating point value.
SKEW CDATA #REQUIRED	Specifies a skew angle for a drop shadow as a floating-point value from -75 degrees to 75 degrees	Specifies a skew angle for a drop shadow as a floating-point value from -75 degrees to 75 degrees	Specifies a skew angle for a drop shadow as a floating-point value from -75 degrees to 75 degrees
SCALE CDATA #REQUIRED	Specifies the size of an items drop shadow as an integer percentage of the size of the item. Valid values are from 10 to 1000 percent.	Specifies the size of an items drop shadow as an integer percentage of the size of the item. Valid values are from 10 to 1000 percent.	Specifies the size of an items drop shadow as an integer percentage of the size of the item. Valid values are from 10 to 1000 percent.
BLUR CDATA #REQUIRED	Specifies the blur distance for a drop shadow, from 0 to 144 points, with higher values creating blurrier edges.	Specifies the blur distance for a drop shadow, from 144, with higher values creating blurrier edges.	Specifies the blur distance for a drop shadow, from 144, with higher values creating blurrier edges.
KNOCKOUTSHADOW (true | false) "false"	Specifies whether a shadow displays through semi-opaque areas of its item.	Specifies whether a shadow displays through semi-opaque areas of its item.	Specifies whether a shadow displays through semi-opaque areas of its item.
SYNCHRONIZEANGLE (true | false) "false"	Specifies whether to synchronize the angle of a drop	Specifies whether to synchronize the angle of a drop	Specifies whether to synchronize the angle of a drop

	shadow with the angles of other drop shadows in the layout.	shadow with the angles of other drop shadows in the layout.	shadow with the angles of other drop shadows in the layout.
RUNAROUNDSHADOW (true | false) "false"	Specifies whether to include a drop shadow with the text runaround specified in the RUNAROUND element. **Note:** The OUTSET attribute of the RUNAROUND element is measured from the edges of the drop shadow. For example, if text is wrapping around a rectangular pull-out quote with a drop shadow, text will not overlap the drop shadow if RUNAROUNDSHADOW is set to true.	Specifies whether to include a drop shadow with the text runaround specified in the RUNAROUND element. **Note:** The OUTSET attribute of the RUNAROUND element is measured from the edges of the drop shadow. For example, if text is wrapping around a rectangular pull-out quote with a drop shadow, text will not overlap the drop shadow if RUNAROUNDSHADOW is set to true.	Specifies whether to include a drop shadow with the text runaround specified in the RUNAROUND element. **Note:** The OUTSET attribute of the RUNAROUND element is measured from the edges of the drop shadow. For example, if text is wrapping around a rectangular pull-out quote with a drop shadow, text will not overlap the drop shadow if RUNAROUNDSHADOW is set to true.
MULTIPLYSHADOW (true | false) "true"	Specifies how a drop shadow is combined with its background. When true, the shadow color is combined with the background color or colors using a "multiply" blending mode, producing a darker result (similar to an overprint). When false, the color of the background is combined with the color of the	Specifies how a drop shadow is combined with its background. When true, the shadow color is combined with the background color or colors using a "multiply" blending mode, producing a darker result (similar to an overprint). When false, the color of the background is combined with the color of the	Specifies how a drop shadow is combined with its background. When true, the shadow color is combined with the background color or colors using a "multiply" blending mode, producing a darker result (similar to an overprint). When false, the color of the background is combined with the color of the

	shadow to create the intermediate shades you see on screen. In general, set to true if the shadow is a lighter color, and set to false if the shadow is black.	shadow to create the intermediate shades you see on screen. In general, set to true if the shadow is a lighter color, and set to false if the shadow is black.	shadow to create the intermediate shades you see on screen.
INHERITOPACITY (true | false) "false">	Specifies whether the drop shadow reflects the opacity or opacities of the item, such as differences in opacity between the box background and frame.	Specifies whether the drop shadow reflects the opacity or opacities of the item, such as differences in opacity between the box background and frame.	Specifies whether the drop shadow reflects the opacity or opacities of the item, such as differences in opacity between the box background and frame.
<!ELEMENT FRAME EMPTY>	Describes a box frame.	Describes a box frame.	Describes a box frame.
<!ATTLIST FRAME			
STYLE CDATA #IMPLIED	Specifies a Dashes & Stripes style for a frame.	Specifies a Dashes & Stripes style for a frame.	Specifies a Dashes & Stripes style for a frame.
WIDTH CDATA #IMPLIED	Specifies the thickness of a frame in points as a floating point value.	Specifies the thickness of a frame in points as a floating point value.	Specifies the thickness of a frame in points as a floating point value.
COLOR CDATA #IMPLIED	Identifies the color of a frame. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a frame. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved	Identifies the color of a frame. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved

		in the project.	in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a frame, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a frame, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a frame, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of a frame, specified as an integer percentage from 0 to 100.	Specifies the opacity of a frame, specified as an integer percentage from 0 to 100.	Specifies the opacity of a frame, specified as an integer percentage from 0 to 100.
GAPCOLOR CDATA #IMPLIED	Identifies the color of a frame gap. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a frame gap. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color of a frame gap. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.
GAPSHADE CDATA #IMPLIED	Specifies the shade of the color applied to a frame gap, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a frame gap, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a frame gap, as an integer percentage from 0 to 100.
GAPOPACITY CDATA #IMPLIED>	Specifies the opacity of the gap color of a frame, specified as an integer percentage from 0 to 100.	Specifies the opacity of the gap color of a frame, specified as an integer percentage from 0 to 100.	Specifies the opacity of the gap color of a frame, specified as an integer percentage from 0 to 100.
<!ELEMENT PLACEHOLDER (#PCDATA)>	Not applicable.	Not applicable.	Describes a placeholder inserted in QuarkXPress for use with XML Import XTensions

			software. **Note:** To replace placeholders with XML content, use XML Import XTensions software with QuarkXPress, or refer to thexmldoc and paginate parameters in this WIG.
<!ATTLIST PLACEHOLDER			
OWNER CDATA #REQUIRED>	Not applicable.	Not applicable.	The name of the element in the XML or DTD that created the Placeholder.
<!ELEMENT TABLE (ID, (ADDCELLS | DELETECELLS | COLSPEC | ROW | FRAME | GEOMETRY | SHADOW)*)>	Describes a table. **Note:** The size and position of a table are defined using the GEOMETRY element.	Describes a table. **Note:** The size and position of a table are defined using the GEOMETRY element.	Describes a table. **Note:** The size and position of a table are defined using the GEOMETRY element.
<!ATTLIST TABLE			
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the indicated table.	Not applicable.
ROWS CDATA #IMPLIED	Specifies the number of rows in a table.	Specifies the number of rows in a table.	Specifies the number of rows in a table.
COLUMNS CDATA #IMPLIED	Specifies the number of columns in a table.	Specifies the number of columns in a table.	Specifies the number of columns in a table.
MAINTAINGEOMETRY (true | false | none) "none"	Controls whether inserted rows or columns affect the entire table's width and height. true = Table height and width remain the same. false = Table height and width change to accommodate new rows and	Controls whether inserted rows or columns affect the entire table's width and height. true = Table height and width remain the same. false = Table height and width change to accommodate new rows and	Controls whether inserted rows or columns affect the entire table's width and height. true = Table height and width remain the same. false = Table height and width change to accommodate new rows and

	columns.	columns.	columns.
COLOR CDATA #IMPLIED	Identifies the color of a table. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a table. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color of a table. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a table, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a table, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a table, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of the color applied to a table, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a table, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a table, specified as an integer percentage from 0 to 100.
BLENDSTYLE (SOLID | LINEAR | MIDLINEAR | RECTANGULAR | DIAMOND | CIRCULAR | FULLCIRCULAR | none) "none"	Specifies the type of blend applied to this table (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this table (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this table (linear, circular, rectangular, etc.).
BLENDANGLE CDATA #IMPLIED	Specifies the angle of the blend.	Specifies the angle of the blend.	Specifies the angle of the blend.
BLENDCOLOR CDATA #IMPLIED	Specifies the second color of the blend. The first color of the blend is the color applied to the table, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the table, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the table, as in QuarkXPress.
BLENDSHADE CDATA #IMPLIED	Specifies the shade applied to	Specifies the shade applied to	Specifies the shade applied to

	the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the table.	the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the table.	the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the table.
BLENDOPACITY CDATA #IMPLIED	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the table.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the table.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the table.
ANCHOREDIN CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates an anchored box and identifies its parent box.
AUTOFIT (rows | columns | all | none) "none"	Specifies whether the rows or columns will adjust size to fit the content.	Specifies whether the rows or columns will adjust size to fit the content.	Specifies whether the rows or columns will adjust size to fit the content.
AUTOFITMAXLIMIT CDATA #IMPLIED>	Max limit for AUTOFIT.	Max limit for AUTOFIT.	Max limit for AUTOFIT.
<!ELEMENT PARENTTABLE EMPTY>	Identifies the originating table when a table has been broken.	Identifies the originating table when a table has been broken.	Identifies the originating table when a table has been broken.
<!ATTLIST PARENTTABLE			
NAME CDATA #IMPLIED	Specifies the name of the parent table.	Specifies the name of the parent table.	Specifies the name of the parent table.
UID CDATA #IMPLIED>	Not applicable.	Specifies the ID of the parent table assigned from QuarkXPress Server.	Specifies the ID of the parent table assigned from QuarkXPress Server.
<!ELEMENT TABLEBREAK (CHILDID | HEADER | FOOTER)*>	Sets a table break for a HEADER or FOOTER or both.	Sets a table break for a HEADER or FOOTER or both.	Sets a table break for a HEADER or FOOTER or both.
<!ATTLIST TABLEBREAK			
BREAKHEIGHT CDATA #REQUIRED	Specifies the height at which a table is set to break.	Specifies the height at which a table is set to break.	Indicates the height at which a table is set to break.

MAINTAINLINK (true | false) "true">	Specifies whether a child table will maintain a link to its parent.	Specifies whether a child table will maintain a link to its parent.	Specifies whether a child table will maintain a link to its parent.
<!ELEMENT CHILDID EMPTY>	Specifies a child of a parent TABLE element.	Specifies a child of a parent TABLE element.	Specifies a child of a parent TABLE element.
<!ATTLIST CHILDID			
NAME CDATA #IMPLIED	Indicates the user-assigned name of the CHILD element of the parent table.	Not applicable.	Indicates the user-assigned name of the CHILD element of the parent table.
UID CDATA #IMPLIED>	Not applicable.	Indicates the ID of the CHILD element of the parent table assigned from QuarkXPress Server.	Indicates the ID of the CHILD element of the parent table assigned from QuarkXPress Server.
<!ELEMENT ADDCELLS EMPTY>	Not applicable.	Adds cells to an existing table. **Note:** If you add a column, you must also define every ROW and CELL element in that column.	Not applicable.
<!ATTLIST ADDCELLS			
TYPE (ROW | COLUMN | HEADER | FOOTER) #REQUIRED	Not applicable.	Specifies whether to add rows, columns, headers, or footers.	Not applicable.
BASEINDEX CDATA #REQUIRED	Not applicable.	Specifies the index number of the cell before or after which the new cells should be inserted. See the INSERTPOSITION attribute.	Not applicable.
INSERTCOUNT CDATA #REQUIRED	Not applicable.	Specifies how many cells to add.	Not applicable.
INSERTPOSITION (AFTER | BEFORE) "AFTER"	Not applicable.	Specifies whether to add the new cells before or after the cell indicated in the BASEINDEX	Not applicable.

		attribute.	
KEEPATTRIBUTE (true | false) "false">	Not applicable.	Specifies whether an inserted row or column should adopt the same attributes as the BASEINDEX cell.	Not applicable.
<!ELEMENT DELETECELLS EMPTY>	Not applicable.	Deletes cells from an existing table.	Not applicable.
<!ATTLIST DELETECELLS			
TYPE (ROW | COLUMN | HEADER | FOOTER) #REQUIRED	Not applicable.	Specifies whether to delete rows, columns, headers, or footers.	Not applicable.
BASEINDEX CDATA #REQUIRED	Not applicable.	Specifies the index number of the first cell to be deleted.	Not applicable.
DELETECOUNT CDATA #REQUIRED>	Not applicable.	Specifies how many cells to delete.	Not applicable.
<!ELEMENT COLSPEC (COLUMN+)>	Describes the columns in a table. **Note:** If the COLSPEC element is missing for a table, then the table is created using columns of equal width, based on the number of columns in the row with the most columns.	Describes the columns in a table. **Note:** If the COLSPEC element is missing for a new table, then the table is created using columns of equal width, based on the number of columns in the row with the most columns.	Describes the columns in a table.
<!ELEMENT COLUMN (LINE*)>	Describes a column in a table.	Describes a column in a table.	Describes a column in a table.
<!ATTLIST COLUMN			
COLUMNCOUNT CDATA #REQUIRED	Specifies the index position of a column beginning from the left. For example, COLUMNCOUNT = 1 indicates the first column from the left, and COLUMNCOUNT = 2 indicates	Specifies the index position of a column beginning from the left. For example, COLUMNCOUNT = 1 indicates the first column from the left, and COLUMNCOUNT = 2 indicates	Specifies the index position of a column beginning from the left. For example, COLUMNCOUNT = 1 indicates the first column from the left, and COLUMNCOUNT = 2 indicates

	the second column from the left.	the second column from the left.	the second column from the left.
COLUMNWIDTH CDATA #IMPLIED	Specifies the width of a column.	Specifies the width of a column.	Specifies the width of a column.
COLOR CDATA #IMPLIED	Identifies the color of a column. Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a column. Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color of a column. Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a column, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a column, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a column, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of the color applied to a column, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a column, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a column, specified as an integer percentage from 0 to 100.
MERGECOLSPAN CDATA #IMPLIED	Attribute used for merging cells, rows, and columns.	Attribute used for merging cells, rows, and columns.	If a table includes merged cells, then the MERGECOLSPAN value is shown in the xml output.
SPLIT (true | false) #IMPLIED	Not applicable.	Attribute used for splitting merged cells.	Not applicable.

AUTOFIT (true | false | none) "none"	Specifies whether the rows or columns will adjust size to fit the content.	Specifies whether the rows or columns will adjust size to fit the content.	Indicates whether the rows or columns will adjust size to fit the content.
AUTOFITMAXLIMIT CDATA #IMPLIED>	Max limit for autofit.	Max limit for autofit.	Max limit for autofit.
<!ELEMENT ROW ((CELL | LINE)*)>	Describes a row in a table.	Describes a row in a table.	Describes a row in a table.
<!ATTLIST ROW			
ROWCOUNT CDATA #REQUIRED	Specifies the index position of a row from top to bottom. For example, ROWCOUNT = 1 indicates the first row from the top, and ROWCOUNT = 2 indicates the second row from the top.	Specifies the index position of a row from top to bottom. For example, ROWCOUNT = 1 indicates the first column from the top, and ROWCOUNT = 2 indicates the second row from the top.	Specifies the index position of a row from top to bottom. For example, ROWCOUNT = 1 indicates the first column from the top, and ROWCOUNT = 2 indicates the second column from the top.
ROWHEIGHT CDATA #IMPLIED	Specifies the height of a row. **Note:** If this attribute is empty, the row is resized to fit its contents, unless RICHTEXT@MAINTAINGEOMETRY is set to true, in which case any row that does not have a ROWHEIGHT attribute will be sized equally using the amount of space remaining after all the specified ROWHEIGHT attributes have been subtracted from the total height of the box.	Specifies the height of a row. **Note:** If this attribute is empty, the row is resized to fit its contents, unless RICHTEXT@MAINTAINGEOMETRY is set to true, in which case any row that does not have a ROWHEIGHT attribute will be sized equally using the amount of space remaining after all the specified ROWHEIGHT attributes have been subtracted from the total height of the box.	Specifies the height a row.
COLOR CDATA #IMPLIED	Identifies the color of a row.	Identifies the color of a row.	Identifies the color of a row.

	Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Overrides the TABLE@COLOR attribute. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a row, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a row, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a row, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of the color applied to a row, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a row, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a row, specified as an integer percentage from 0 to 100.
MERGEROWSPAN CDATA #IMPLIED	Attribute used for merging cells and rows.	Attribute used for merging cells and rows.	If a table includes merged cells, then the MERGECOLSPAN value is shown in the xml output.
SPLIT (true | false) #IMPLIED	Not applicable.	Attribute used for splitting rows and columns.	Not applicable.
AUTOFIT (true | false | none) "none"	Specifies whether the rows or columns will adjust size to fit the content.	Specifies whether the rows or columns will adjust size to fit the content.	Specifies whether the rows or columns will adjust size to fit the content.
AUTOFITMAXLIMIT CDATA #IMPLIED>	Max limit for autofit.	Max limit for autofit.	Max limit for autofit.

<!ELEMENT HEADER (ROW*)>	Specifies if the row is to be a header row.	Specifies if the row is to be a header row.	Indicates if the row is to be a header row.
<!ATTLIST HEADER			
HEADERROWS CDATA #IMPLIED>	Specifies number of header row.	Specifies number of header row.	Specifies number of header row.
<!ELEMENT FOOTER (ROW*)>	Specifies if the row is to be a footer row.	Specifies if the row is to be a footer row.	Indicates if the row is to be a footer row.
<!ATTLIST FOOTER			
FOOTERROWS CDATA #IMPLIED>	Specifies number of footer row.	Specifies number of footer row.	Specifies number of footer row.
<!ELEMENT CELL ((CONTENT| CONTENTPH | TEXT | PICTURE | PLACEHOLDER)*)>	Describes a table cell.	Describes a table cell.	Describes a table cell.
<!ATTLIST CELL			
COLUMNCOUNT CDATA #REQUIRED	Specifies the column index position of a cell, with the first cell being cell 1.	Specifies the column index position of a cell, with the first cell being cell 1.	Specifies the column index position of a cell, with the first cell being cell 1.

BOXTYPE (CT_NONE | CT_TEXT | CT_PICT) #IMPLIED	Specifies a cells type:
	CT_NONE = No-content cell
	CT_TEXT = Text cell
	CT_PICT = Picture cell
		Specifies a cells type:
		CT_NONE = No-content cell
		CT_TEXT = Text cell
		CT_PICT = Picture cell
			Specifies a cells type:
			CT_NONE = No-content cell
			CT_TEXT = Text cell
			CT_PICT = Picture cell

COLOR CDATA #IMPLIED	Identifies the color of a cell. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	Identifies the color of a cell. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.	Identifies the color of a cell. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Specifies the shade of the color	Specifies the shade of the color	Specifies the shade of the color

	applied to a cell, as an integer percentage from 0 to 100.	applied to a cell, as an integer percentage from 0 to 100.	applied to a cell, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of the color applied to a cell, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a cell, specified as an integer percentage from 0 to 100.	Specifies the opacity of the color applied to a cell, specified as an integer percentage from 0 to 100.
BLENDSTYLE (SOLID | LINEAR | MIDLINEAR | RECTANGULAR | DIAMOND | CIRCULAR | FULLCIRCULAR | none) "none"	Specifies the type of blend applied to this cell (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this cell (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this cell (linear, circular, rectangular, etc.).
BLENDANGLE CDATA #IMPLIED	Specifies the angle of the blend.	Specifies the angle of the blend.	Specifies the angle of the blend.
BLENDCOLOR CDATA #IMPLIED	Specifies the second color of the blend. The first color of the blend is the color applied to the cell, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the cell, as in QuarkXPress.	Specifies the second color of the blend. The first color of the blend is the color applied to the cell, as in QuarkXPress.
BLENDSHADE CDATA #IMPLIED	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the cell.	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the cell.	Specifies the shade applied to the second color of the blend. The shade of the first color of the blend is the shade of the color applied to the cell.
BLENDOPACITY CDATA #IMPLIED	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the cell.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the cell.	Specifies the opacity applied to the second color of the blend. The opacity of the first color of the blend is the opacity of the color applied to the cell.
MERGEROWSPAN CDATA #IMPLIED	Attribute used for merging cells and rows.	Attribute used for merging cells and rows.	If a table includes merged cells, then the MERGECOLSPAN value is shown in the xml

			output.
MERGECOLSPAN CDATA #IMPLIED	Attribute used for merging cells and columns.	Attribute used for merging cells and columns.	Not applicable.
SPLIT (true | false) #IMPLIED>	Not applicable.	Attribute used for splitting rows and columns.	Not applicable.
<!ELEMENT GRID (GRIDLINE)>	Element used for specifying a grid in a table.	Element used for specifying a grid in a table.	Element used for specifying a grid in a table.
<!ATTLIST GRID			
TYPE (HGRID | VGRID | ALLGRID) #IMPLIED>	Not applicable.	Attribute used for selecting a horizontal or vertical grid (or both).	Not applicable.
<!ELEMENT GRIDLINE EMPTY>	Element used to define line attributes.	Element used to define line attributes.	Element used to define line attributes.
<!ATTLIST GRIDLINE			
STYLE CDATA #IMPLIED	Identifies a Dashes & Stripes style (LINESTYLE) for a rule. Note: Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the Document Controls submenu in QuarkXPress Server.	Identifies a Dashes & Stripes style (LINESTYLE) for a rule. Note: Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the Document Controls submenu in QuarkXPress Server.	Identifies a Dashes & Stripes style (LINESTYLE) for a rule. Note: Only the name of a Dashes & Stripes style is included in this attribute. The definition of the Dashes & Stripes style is stored in the projects Job Jackets file or defined using the Document Controls submenu in QuarkXPress Server.
WIDTH CDATA #IMPLIED	Specifies the thickness of a line as a floating point value (measured in points).	Specifies the thickness of a line as a floating point value (measured in points).	Specifies the thickness of a line as a floating point value (measured in points).
COLOR CDATA #IMPLIED	Identifies the color of a line.	Identifies the color of a line.	Identifies the color of a line.
SHADE CDATA #IMPLIED	Specifies the shade of the color applied to a line,	Specifies the shade of the color applied to a line,	Specifies the shade of the color applied to a line,

	as an integer percentage from 0 to 100.	as an integer percentage from 0 to 100.	as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Specifies the opacity of a line, specified as an integer percentage from 0 to 100.	Specifies the opacity of a line, specified as an integer percentage from 0 to 100.	Specifies the opacity of a line, specified as an integer percentage from 0 to 100.
GAPCOLOR CDATA #IMPLIED	Identifies the color of a line gap.	Identifies the color of a line gap.	Identifies the color of a line gap.
GAPSHADE CDATA #IMPLIED	Specifies the shade of the color applied to a line gap, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a line gap, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a line gap, as an integer percentage from 0 to 100.
GAPOPACITY CDATA #IMPLIED>	Specifies the opacity of the gap color of a line, specified as an integer percentage from 0 to 100.	Specifies the opacity of the gap color of a line, specified as an integer percentage from 0 to 100.	Specifies the opacity of the gap color of a line, specified as an integer percentage from 0 to 100.
<!ELEMENT GROUP (ID, (BOX | TABLE | COMPOSITIONZONE | GROUP)*)>	Not applicable.	Not applicable.	Describes a group of items.
<!ATTLIST GROUP			
ANCHOREDIN CDATA #IMPLIED>	Not applicable.	Not applicable.	Indicates an anchored box in a text box and identifies its parent box.
<!ELEMENT COMPOSITIONZONE (ID, (FRAME | GEOMETRY | SHADOW)*)>	Describes a Composition Zones item. (Applies only to the xml namespace.)		
<!ATTLIST COMPOSITIONZONE			
BOXTYPE (CT_USER) #IMPLIED	Not applicable.	Not applicable.	Indicates CT_USER as the box type for a Composition Zones item.
TYPE (INTERNAL | EXTERNAL) #IMPLIED	Not applicable.	Not applicable.	Indicates the Composition Zones items type. INTERNAL = A Composition Zones item that uses a layout

			within the same project. EXTERNAL = A Composition Zones item that uses a layout in a different project.
PATH CDATA #IMPLIED	Not applicable.	Not applicable.	Indicates the absolute path to an external composition layout.
COLOR CDATA #IMPLIED	Not applicable.	Not applicable.	Identifies a color applied to a Composition Zones item. **Note:** Only the name of a color is included in this attribute. The definition of the color is stored in the projects Job Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server, or an existing color created and saved in the project.
SHADE CDATA #IMPLIED	Not applicable.	Not applicable.	Specifies the shade of a color applied to a Composition Zones object, as an integer percentage from 0 to 100.
OPACITY CDATA #IMPLIED	Not applicable.	Not applicable.	Specifies the opacity of a color applied to a Composition Zones item, specified as an integer percentage from 0 to 100.
ANCHOREDIN CDATA	Not applicable.	Not applicable.	Indicates an

#IMPLIED>			anchored Composition Zones and identifies its parent Composition Zones.
<!ELEMENT LIST ((PARAGRAPH | RICHTEXT)*,OVERMATTER?)>	Specifies a List in a QuarkXPress project.	Specifies a List in a QuarkXPress project.	Identifies a List in a QuarkXPress project.
<!ATTLIST LIST			
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create a list or delete a list.	Not applicable.
LISTSTYLE CDATA #REQUIRED>	Name of the List as defined in QuarkXPress.	Name of the List as defined in QuarkXPress.	Name of the List as defined in QuarkXPress.

Sample applications: QuarkXPress Server
These sample applications are available in the QuarkXPress Server installation files.

Java client demo

Purpose

This sample client gives standard QuarkXPress Server post requests for various operations.

Technology

This demo uses Swing components to create the user interface. It gathers the user information, generates a Multi-part Post request, and sends that request to QuarkXPress Server. It saves the QuarkXPress Server response in the form of a file in the current folder.

Installation

The sample is an executable JAR file that was developed and tested for the Windows(R) platform. A user can copy it anywhere and launch it by double-clicking, assuming that JVM TM is installed on the system.
"QXPSClientSample_Java.jar"

C# client demo

Purpose

This sample client gives standard QuarkXPress Server post requests for various operations. It works with the Modifier XTensions software (with XML on the client machine as well as on the server machine):

-
- Layers
-
- XML Import
-
- Vista
-
- Addfile
-
- Rendering of QuarkXPress project in different file formats

Technology

C#, .Net

Installation

The sample is an executable file tested for the Windows platform. After copying it to your computer, double-click it to launch.

"QXPSClientExec_C#.zip"

PHP-MySQL demo

This demo shows how QuarkXPress Server can be used with PHP and MySQL® to enable dynamic updating of the contents of a Web site.

Use the links below for detailed documentation (a PDF file) and the necessary binary files for Mac OS(R) or Windows.

"SampleAppGuide_PHP.pdf"
"qxpsdemo_PHP.zip"
"ScalingXT_QXPS7_PHP.zip"
Legal notice

Sample applications: QuarkXPress Server Manager
These sample applications are available in the QuarkXPress Server installation files.

JSP samples

Purpose

Contain a series of Web pages demonstrating different ways the object model can be used to post QuarkXPress Server requests for various operations.

Installation

The samples are JSP™ pages contained in ClientSDKSamples_JSP.zip file. They need to be extracted and placed under tomcat/webapps. Also if QuarkXPress Server Manager is installed, then these samples are already available as a Web site in Tomcat at tomcaturl/ClientSDKsamples.

Implementation

Set the endpoint address for the web service calls in file web.xml:
web-app-->CLIENT_SDK_URL-->Value
"ClientSDKSamples_JSP.zip"

ASP.NET samples

Purpose

Contains series of Web pages demonstrating different ways the object model can be used to post QuarkXPress Server requests for various operations.

Technology

ASP.Net

Installation

Follow these steps to install the Web demo.

1.
1. Create a virtual directory - for example, ClientSDKSamplesSite - in IIS.
2.
2. Extract the samples from ClientSDKSamples_ASPDOTNET.zip and set the home path of the Web demo to the virtual directory.
3.
3. Set the endpoint address for web services calls in the web.config file:
configuration-->appSettings-->add key="ClientSDKSamples.sdk.QManagerSDKSvcService" value= "End Point Address"
4.
4. Restart IIS.
5.
5. Go to http://<IIS Server Name>:<Port>/ClientSDKSamplesSite/Index.htm

"ClientSDKSamples_ASPDOTNET.zip"
Legal notice

Contact Quark
Quark, Inc.
1800 Grant St
Denver, CO 80203
Phone: (303) 894-8888
Developer Desk Fax: (303) 894-3782

Submit technical questions about QuarkXPress Server and QuarkXPress Server Manager to Quark or e-mail QuarkXPress Server support:

QuarkXPress Server Support

Visit Quark's Web site:

Quark Web Site

Log on to the Developer Resource Site*:

Developer Resource Site

*The Developer Resource Site is an online support center maintained specifically for XTensions software developers. Accessed through the Quark(R) Web site, the Developer Resource Site contains developer kit downloads, online documentation, additional sample files, online catalog entries, and an XTensions software knowledge base. It also has information about new developer programs, marketing opportunities, and more. Note that a username and password are required to access the site. To obtain your username and password, you must join the QuarkAlliance™ Developer program.

LEGAL NOTICE

©2008 Quark, Inc. as to the content and arrangement of this material. All rights reserved.
©1986-2008 Quark, Inc. and its licensors as to the technology. All rights reserved.

Protected by one or more of U.S. Patent Nos. 5,541,991, 5,907,704, 6,005,560, 6,052,514, 6,081,262, 6,947,959 B1, 6,940,518 B2, 7,116,843 and other patents pending.

Quark Products and materials are subject to the copyright and other intellectual property protection of the United States and foreign countries. Unauthorized use or reproduction without Quark's written consent is prohibited.

QUARK IS NOT THE MANUFACTURER OF THIRD PARTY SOFTWARE OR OTHER THIRD PARTY HARDWARE (HEREINAFTER "THIRD PARTY PRODUCTS") AND SUCH THIRD PARTY PRODUCTS HAVE NOT BEEN CREATED, REVIEWED, OR TESTED BY QUARK, THE QUARK AFFILIATED COMPANIES OR THEIR LICENSORS. (QUARK AFFILIATED COMPANIES SHALL MEAN ANY PERSON, BRANCH, OR ENTITY CONTROLLING, CONTROLLED BY OR UNDER COMMON CONTROL WITH QUARK OR ITS PARENT OR A MAJORITY OF THE QUARK SHAREHOLDERS, WHETHER NOW EXISTING OR FORMED IN THE FUTURE, TOGETHER WITH ANY PERSON, BRANCH, OR ENTITY WHICH MAY ACQUIRE SUCH STATUS IN THE FUTURE.)QUARK, THE QUARK AFFILIATED COMPANIES AND/OR THEIR LICENSORS MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE QUARK PRODUCTS/SERVICES AND/OR THIRD PARTY PRODUCTS/SERVICES, THEIR MERCHANTABILITY, OR THEIR FITNESS FOR A PARTICULAR PURPOSE. QUARK, THE QUARK AFFILIATED COMPANIES AND THEIR LICENSORS DISCLAIM ALL WARRANTIES RELATING TO THE QUARK PRODUCTS/SERVICES AND ANY THIRD PARTY PRODUCTS/SERVICES. ALL OTHER WARRANTIES AND CONDITIONS, WHETHER EXPRESS, IMPLIED OR COLLATERAL, AND WHETHER OR NOT, MADE BY DISTRIBUTORS, RETAILERS, XTENSIONS DEVELOPERS OR OTHER THIRD PARTIES ARE DISCLAIMED BY QUARK, THE QUARK AFFILIATED COMPANIES AND THEIR LICENSORS, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF NON-INFRINGEMENT, COMPATIBILITY, OR THAT THE SOFTWARE IS ERROR-FREE OR THAT ERRORS CAN OR WILL BE CORRECTED. THIRD PARTIES MAY PROVIDE LIMITED WARRANTIES AS TO THEIR OWN PRODUCTS AND/OR SERVICES, AND USERS MUST LOOK TO SAID THIRD PARTIES FOR SUCH WARRANTIES, IF ANY. SOME JURISDICTIONS, STATES OR PROVINCES DO NOT ALLOW LIMITATIONS ON IMPLIED WARRANTIES, SO THE ABOVE LIMITATION MAY NOT APPLY TO PARTICULAR USERS.IN NO EVENT SHALL QUARK, THE QUARK AFFILIATED COMPANIES, AND/OR THEIR LICENSORS BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, BUT NOT LIMITED TO, ANY LOST PROFITS, LOST TIME, LOST SAVINGS, LOST DATA, LOST FEES, OR EXPENSES OF ANY KIND ARISING FROM INSTALLATION OR USE OF THE QUARK PRODUCTS/SERVICES, IN ANY MANNER, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. IF, NOTWITHSTANDING THE FOREGOING, QUARK, THE QUARK AFFILIATED COMPANIES AND/OR THEIR LICENSORS ARE FOUND TO HAVE LIABILITY RELATING TO THE QUARK PRODUCTS/SERVICES OR THIRD PARTY PRODUCTS/SERVICES, SUCH LIABILITY SHALL BE LIMITED TO THE AMOUNT PAID BY THE USER TO QUARK FOR THE SOFTWARE/SERVICES AT ISSUE (EXCLUDING THIRD PARTY PRODUCTS/SERVICES), IF ANY, OR THE LOWEST AMOUNT UNDER

APPLICABLE LAW, WHICHEVER IS LESS. THESE LIMITATIONS WILL APPLY EVEN IF QUARK, THE QUARK AFFILIATED COMPANIES, THEIR LICENSORS AND/OR THEIR AGENTS HAVE BEEN ADVISED OF SUCH POSSIBLE DAMAGES. SOME JURISDICTIONS, STATES OR PROVINCES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION OR EXCLUSION MAY NOT APPLY. ALL OTHER LIMITATIONS PROVIDED UNDER APPLICABLE LAW, INCLUDING STATUTES OF LIMITATION, SHALL CONTINUE TO APPLY.IN THE EVENT ANY OF THESE PROVISIONS ARE OR BECOME UNENFORCEABLE UNDER APPLICABLE LAW, SUCH PROVISION SHALL BE MODIFIED OR LIMITED IN ITS EFFECT TO THE EXTENT NECESSARY TO CAUSE IT TO BE ENFORCEABLE.USE OF THE QUARK PRODUCTS IS SUBJECT TO THE TERMS OF THE END USER LICENSE AGREEMENT OR OTHER APPLICABLE AGREEMENTS FOR SUCH PRODUCT/SERVICE. IN THE EVENT OF A CONFLICT BETWEEN SUCH AGREEMENTS AND THESE PROVISIONS THE RELEVANT AGREEMENTS SHALL CONTROL.

Quark, the Quark logo, QuarkXPress, XTensions, QuarkCopyDesk, Job Jackets and Composition Zones, QuarkAlliance and QPS are trademarks or registered trademarks of Quark, Inc. and its affiliates in the U.S. and/or other countries.

OpenType, Visual C#, Visual Studio, Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Mac OS is a trademark of Apple, Inc. registered in the U.S. and other countries.

Adobe, PostScript and Acrobat are registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Unicode is a trademark of Unicode, Inc.

MySQL is a registered trademark of MySQL AB.

PANTONE® Colors displayed in the software application or in the user documentation may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE® and other Pantone, Inc. trademarks are the property of Pantone, Inc. ©Pantone, Inc., 2008.

Color Data is produced under license from Dainippon Ink and Chemicals, Inc.

FOCOLTONE and FOCOLTONE Colour System are registered trademarks of FOCOLTONE. The concept, structure, and form of FOCOLTONE material and intellectual property are protected by patent and copyright law. Any reproduction in any form, in whole or in part, for private use or for sale, is strictly forbidden. Contact FOCOLTONE, Ltd. for specific patent information.

Toyo Ink Mfg. Co., Ltd. is the copyright owner of TOYO INK COLOR FINDER™ SYSTEM AND SOFTWARE which is licensed to Quark, Inc. to distribute for use only in connection with QuarkXPress. TOYO INK COLOR FINDER™ SYSTEM AND SOFTWARE shall not be copied onto another diskette or into memory unless as part of the execution of QuarkXPress. TOYO INK COLOR FINDER™ SYSTEM AND SOFTWARE © TOYO INK MFG. CO., LTD., 1991. COLOR FINDER is in the process of registration as the registered trademark of Toyo Ink Mfg. Co., Ltd. COLOR FINDER™ computer video simulation used in the product may not match the COLOR FINDER™ book, and additionally some printer color used in the product may also not match. Please use the COLOR FINDER™ book to obtain the accurate color."

TRUMATCH, TRUMATCH Swatching System, and TRUMATCH System are trademarks of TRUMATCH, Inc.

As to tt2pt1 technology, Copyright (c) 1997-2003 by the AUTHORS: Andrew Weeks <ccsaw@bath.ac.uk> Frank M. Siegert <fms@this.net> Mark Heath <mheath@netspace.net.au> Thomas Henlich <thenlich@rcs.urz.tu-dresden.de> Sergey Babkin <babkin@users.sourceforge.net>, <sab123@hotmail.com> Turgut Uyar <uyar@cs.itu.edu.tr> Rihardas Hepas <rch@WriteMe.Com> Szalay Tamas <tomek@elender.hu> Johan Vromans <jvromans@squirrel.nl> Petr Titera

<P.Titera@sh.cvut.cz> Lei Wang <lwang@amath8.amt.ac.cn> Chen Xiangyang <chenxy@sun.ihep.ac.cn> Zvezdan Petkovic <z.petkovic@computer.org> Rigel <rigel863@yahoo.com> All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the TTF2PT1 Project and its contributors. THIS SOFTWARE IS PROVIDED BY THE AUTHORS 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 AUTHORS 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. For the approximate list of the AUTHORS' responsibilities see the project history. Other contributions to the project are: Turgut Uyar <uyar@cs.itu.edu.tr> The Unicode translation table for the Turkish language. Rihardas Hepas <rch@WriteMe.Com> The Unicode translation table for the Baltic languages. Szalay Tamas <tomek@elender.hu> The Unicode translation table for the Central European languages. Johan Vromans <jvromans@squirrel.nl> The RPM file. Petr Titera <P.Titera@sh.cvut.cz> The Unicode map format with names, the forced Unicode option. Frank M. Siegert <frank@this.net> Port to Windows Lei Wang <lwang@amath8.amt.ac.cn> Chen Xiangyang <chenxy@sun.ihep.ac.cn> Translation maps for Chinese fonts. Zvezdan Petkovic <z.petkovic@computer.org> The Unicode translation tables for the Cyrillic alphabet. Rigel <rigel863@yahoo.com> Generation of the dvips encoding files, modification to the Chinese maps. I. Lee Hetherington <ilh@lcs.mit.edu> The Type1 assembler (from the package 't1utils'), its full copyright notice: Copyright (c) 1992 by I. Lee Hetherington, all rights reserved. Permission is hereby granted to use, modify, and distribute this program for any purpose provided this copyright notice and the one below remain intact.

As to Apache technology, copyright (c) 1999-2008 The Apache Software Foundation. All rights reserved. Any Apache software which is distributed with this software is developed by the Apache Software Foundation (http://www.apache.org/). Licensed under the Apache License, Version 2.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS?IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

As to MoreFiles software, ©1992-2002 by Apple Computer, Inc., all rights reserved.

Portions of this product include technology used under license from Global Graphics.

As to ICU4J technology, ICU4J license - ICU4J 1.3.1 and later,

COPYRIGHT AND PERMISSION NOTICE , Copyright (c) 1995-2001 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. 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 OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

This software is based in part on the work of the Independent JPEG Group.

As to Microsoft technology, ©1988-2008 Microsoft Corporation. All rights reserved.

As to Nodeka software, ©1999-2002 Justin Gottschlich. All rights reserved.

As to STLport technology, Copyright 1999,2000 Boris Fomitchev. This material is provided “as is”, with absolutely no warranty expressed or implied. Any use is at your own risk. Permission to use or copy this sofware for any purpose is hereby granted without fee, provided the above notices are retained on all cpies. Permission to modify the code and to distribute modified code is granted, provided the above notices are retained, and a notice that the code was modified is included with the above copyright notice. The Licensee may distribute binaries compiled with STLport (whether original or modified) without any royalties or restrictions. The Licensee may distribute original or modified STLport sources, provided that: The conditions indicated in the above permission notice are met; The following copyright notices are retained when present, and conditions provided in accompanying permission notices are met: Copyright 1994 Hewlett-Packard Company. Copyright 1996,97 Silicon Graphics Computer Systems, Inc. Copyright 1997 Moscow Center for SPARC Technology.

Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Hewlett-Packard Company makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Silicon Graphics makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Moscow Center for SPARC Technology makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

As to Dr. Brian Gladman software, Copyright (c) 2001, Dr. Brian Gladman <brg@gladman.uk.net>, Worcester, UK. All rights reserved. LICENSE TERMS The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that: 1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer; 2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials; 3. the copyright holder’s name is not used to endorse products built using this software without specific written permission. DISCLAIMER This software is provided ‘as is’ with no explicit or implied warranties in respect of any properties, including, but not limited to, correctness and fitness for purpose.

As to cascading menus based on menu.js. by Gary Smith, July 1997, Copyright (c) 1997-1999 Netscape Communication Corp. Netscape grants you a royalty free license to use or modify the

cascading menus software provided that this copyright notice appears on all copies. This software is provided "AS IS," without a warranty of any kind.

Portions of this software is based on the work of Jean-loup Gailly and Mark Adler and is (C) 1995-1998 Jean-loup Gailly and Mark Adler [ZIP library]

As to Sun technology, Copyright 2003-2006, Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

As to Apple technology, ©2002-2004 Apple Computer, Inc. All rights reserved. This Apple software is supplied to you by Apple Computer, Inc. ("Apple") in consideration of your agreement to the following terms, and your use, installation, modification or redistribution of this Apple software constitutes acceptance of these terms. If you do not agree with these terms, please do not use, install, modify or redistribute this Apple software. In consideration of your agreement to abide by the following terms, and subject to these terms, Apple grants you a personal, non-exclusive license, under Apple's copyrights in this original Apple software (the "Apple Software"), to use, reproduce, modify and redistribute the Apple Software, with or without modifications, in source and/or binary forms; provided that if you redistribute the Apple Software in its entirety and without modifications, you must retain this notice and the following text and disclaimers in all such redistributions of the Apple Software. Neither the name, trademarks, service marks or logos of Apple Computer, Inc. may be used to endorse or promote products derived from the Apple Software without specific prior written permission from Apple. Except as expressly stated in this notice, no other rights or licenses, express or implied, are granted by Apple herein, including but not limited to any patent rights that may be infringed by your derivative works or by other works in which the Apple Software may be incorporated. The Apple Software is provided by Apple on an "AS IS" basis. APPLE MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE APPLE SOFTWARE OR ITS USE AND OPERATION ALONE OR IN COMBINATION WITH YOUR PRODUCTS. IN NO EVENT SHALL APPLE BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) ARISING IN ANY WAY OUT OF THE USE, REPRODUCTION, MODIFICATION AND/OR DISTRIBUTION OF THE APPLE SOFTWARE, HOWEVER CAUSED AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF APPLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

As to HTML Parsing code technology, Copyright (c) 1998 World Wide Web Consortium (Massachusetts
Institute of Technology, Institut National de Recherche en
Informatique et en Automatique, Keio University). All Rights
Reserved.

Contributing Author(s):
Dave Raggett <dsr@w3.org>

The contributing author(s) would like to thank all those who
helped with testing, bug fixes, and patience. This wouldn't
have been possible without all of you.

COPYRIGHT NOTICE:

This software and documentation is provided "as is," and
the copyright holders and contributing author(s) make no
representations or warranties, express or implied, including
but not limited to, warranties of merchantability or fitness
for any particular purpose or that the use of the software or
documentation will not infringe any third party patents,
copyrights, trademarks or other rights.

The copyright holders and contributing author(s) will not be
liable for any direct, indirect, special or consequential damages
arising out of any use of the software or documentation, even if
advised of the possibility of such damage.
Permission is hereby granted to use, copy, modify, and distribute
this source code, or portions hereof, documentation and executables,
for any purpose, without fee, subject to the following restrictions:
1. The origin of this source code must not be misrepresented.
2. Altered versions must be plainly marked as such and must
not be misrepresented as being the original source.
3. This Copyright notice may not be removed or altered from any
source or altered source distribution.
The copyright holders and contributing author(s) specifically
permit, without fee, and encourage the use of this source code
as a component for supporting the Hypertext Markup Language in
commercial products. If you use this source code in a product,
acknowledgment is not required but would be appreciated.

As to DOM4J software, Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.

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

The name "DOM4J" must not be used to endorse or promote products derived from this Software without prior written permission of MetaStuff, Ltd. For written permission, please contact dom4j-info@metastuff.com.

Products derived from this Software may not be called "DOM4J" nor may "DOM4J" appear in their names without prior written permission of MetaStuff, Ltd. DOM4J is a registered trademark of MetaStuff, Ltd.

Due credit should be given to the DOM4J Project - http://www.dom4j.org

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

Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.

As to Jaxen technology: Copyright 2003-2006 The Werken Company. All Rights Reserved.

License Text

$Id: LICENSE.txt,v 1.5 2006/02/05 21:49:04 elharo Exp $

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

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

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

3. Neither the name of the 'Jaxen' nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
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.
All other marks are the property of their respective owners.