QXP Server 8.5 Web Integration Guide

Contents

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 interfaces available in QuarkXPress Server:

- **HTTP**: Lets you interact with the server using URLs that contain calls or point to XML files that contain calls. You can write client applications in any language that supports HTTP requests. For more information, see "*Getting started: HTTP*".
- **Web services**: Lets you interact with the server via Web services using the QuarkXPress Server Manager object model. You can write client applications in Java, .NET, or any other programming language that can consume SOAP-based Web services. For more information, see "*Getting started: Web services*".

➡ To develop a custom load balancer or a custom application in Java, you must have version 1.5 of the JDK.

➡ To use Web Objects in ASP.NET / Visual C#, you must have the .NET 1.1/2.0 framework and development environment (Visual Studio).

The Dynamic Publishing Process (DPP)

The Dynamic Publishing Process (DPP) has several stages. You may not need to use all of these stages every time, but this the order in which they occur:

- *Pre-Processing Stage*: During this stage, QuarkXPress Server performs any necessary initial steps, such as creating style sheets, colors, and H&J rules for a new QuarkXPress project.
- *Content Loading Stage*: During this stage, QuarkXPress Server loads dynamic content into boxes in the project.

- *Layout Modification Stage*: During this stage, QuarkXPress Server modifies the layout of the project.
- *Post-Processing Stage*: During this stage, QuarkXPress Server examines the project and performs maintenance tasks.

The WIG and the XTensions Developer Kit (XDK)

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

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

Changes in this version

Changes in QuarkXPress Server 8.5 include the following:

- The `FITTEXT` element type lets you control the way in which stories are fit into text boxes on a story-by-story basis, rather than requiring you to always use the application preferences. For more information, see "*FITTEXT (Modifier DTD)*." Note that the `FITTEXTTOBOX` still works the same way it did in the previous version, using application preferences.
- The Clang and Cplatform administrative request handlers have been removed.

Getting started

The topics below describe how to create requests for the QuarkXPress Server Web interface. For information about the options available in such requests, see "*Using the Web interface*."

Getting started: HTTP

You can submit HTTP requests to QuarkXPress Server as URLs manually from a browser or automatically from an HTTP client application. QuarkXPress Server processes such requests and returns rendered content in the HTTP responses. Depending on the type of request, QuarkXPress Server preferences, and the type of content returned, rendered content may be downloaded by the end user, displayed in the end-user's browser, or saved to a file system location accessible to QuarkXPress Server.

You can write a QuarkXPress Server client application in almost any language that can generate HTTP GET/POST requests. A QuarkXPress Server HTTP-based solution typically consists 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 end users' input into HTTP requests and sends the requests to QuarkXPress Server or QuarkXPress Server Manager, which processes the requests and returns rendered content.

Dissecting a QXP Server URL

To interact with QuarkXPress Server from a Web browser, use a URL like the following:

```
http://[server]:[port]/[namespace]/[directory]/[DocumentName]?[parameter]=Value
```

- `[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, relative to the QuarkXPress Server document pool. To access the root level, no directory path is necessary.

- `[DocumentName]`: 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. Pass parameters in the form `attribute=value`, with parameters separated by the "&" character.

For QuarkXPress Server Manager, use a URL like the following:

```
http://[server]:[port]/quark/servlet/qxpsm/[namespace]/[directory]/[DocumentName]?[parameter]=Value
```

➡ 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]/` .

➡ Versions of QuarkXPress Server Manager prior to 7.22 required absolute paths. You can now use both absolute and relative paths when you modify a project with SDK objects or classes. Relative paths are almost always relative to the document pool. If you use multiple QuarkXPress Server instances, you should use a common document pool.

Interpreting the QXP Server Manager response

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 supplied additional parameters to QuarkXPress Manager. For more information, see "Working with QuarkXPress Server Manager" in *A Guide to QuarkXPress Server*.

If an error occurs, QuarkXPress Server Manager retries the request, either on the same QuarkXPress server instance or a different one (depending on the error and global settings established in the QuarkXPress Server Manager client). If QuarkXPress Server Manager cannot process the request, it returns an XML response describing the error, plys any header error codes returned by QuarkXPress Server. For example:

```
<?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>
```

HTTP GET and POST Requests

The topics below describe how you can use HTML to interact with QuarkXPress Server.

➡ QuarkXPress Server supports both the GET and POST HTML methods. 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 the message body. Use the GET method only when the form processing is idempotent. In short: GET is for retrieving data, whereas POST can involve storing or updating data, ordering a product, or sending an e-mail.

Using HTTP GET with QXP Server

Use HTML like the following to specify a server and port where you want to send a request. You can specify the name of the target project, the output type, and a scaling value. You can specify the name of a box and the path of a text or picture files to import into that box, as long as the file's path is on the server's file system. You can also use HTML like the following to specify the page number and layout number of the project.

The form section of the HTML should begin with the following line of code:

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

For both GET and POST, the browser constructs a form data set and encodes it according to the `ENCTYPE` attribute (you can use `multipart/form-data` for POST and `application/x-www-form-urlencoded` (the default) for both POST and GET).

To create fields that let the user specify the server IP address, the port, and the project name, use HTML like the following:

```
<TABLE cellSpacing=1 cellPadding=1 border=1 id=TABLE1 >
  <TBODY>
  <TR>
    <TD>
    <INPUT id=ServerTxt name=ServerTxt value="Server ID"
      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>
```

To create a drop-down menu that lets the end user specify a render format, use HTML like the following:

```
<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/>
```

To create a drop-down menu that lets the end user specify a rendering scale, use HTML like the following:

```
<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/>
```

To create input fields that let the end user specify a box name and the name of a file to be imported into that box, use HTML like the following:

```
<TD>
<INPUT id=box1Txt value="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>
```

To create fields that let the end user enter a page number a layout number, use HTML like the following:

```
<TABLE cellSpacing=1 cellPadding=1 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>
```

To create a button that lets the end user submit the request, use HTML like the following:

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

The above HTML calls a function named `Submit_onclick()`. You can add such a function to the `<HEAD>` section of the HTML. For example:

```
<head>
<TITLE>Quark Stream</TITLE>
<script ID="clientEventHandlersJS" LANGUAGE="javascript">
  function Submit_onclick() {
  var prefix;
  var renderer;
  var file;
  var url;
  var box1Name;
  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 = "";
    }
  document.getElementById("Page").name = "Page";
  document.getElementById("Layout").name = "Layout";
  url = prefix + port + renderer + file;
  document.getElementById("form1").action = url;
}
</script>
</head>
```

The `Submit_onclick()` function reads the values from the formand builds a request URL using the server, port, and render type.

➡ If the end user specifies a file name in the "File on Server" text box, he or she must add `file:` to the beginning of the file path (for example, `file:C:\data.txt`).

➡ The code above adds `@dataimport` to the end of the box name to accommodate Quark Data Import XTensions software, which is necessary to import text and picture files.

The action of the form is defined by this line:

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

This form's method is GET. The user agent gets the value (the URL) of the action, appends a `?` to it, adds the form data set, and submits the URL.

➡ In this scenario, form data must be in ASCII.

Using HTTP GET with QXP Server Manager

HTTP GET with QuarkXPress Server Manager works the same way as HTTP GET with QuarkXPress Server (see "*Using HTTP GET with QXP Server*"), except that Quark does not recommend using GET if you are working with non-ASCII characters. The behavior of GET requests with characters is highly browser-dependent, and there is no standard that all browsers follow. Instead, use POST.

Using HTTP POST with QXP Server

Use HTML like the HTML in "*Using HTTP GET with QXP Server*" to specify a server and port where you want to send a request. You can specify the name of the target project, the output type, and a scaling value. You can specify the name of a box and the path of a text or picture files to import into that box, as long as the file's path is on the server's file system. You can also use HTML like the HTML in "*Using HTTP GET with QXP Server*" to specify the page number and layout number of the project. Differences between the GET method and the POST method are described below.

The form section of the HTML should begin with the following line of code:

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

The following HTML creates a input fields that let the end user specify the name of a file to be imported into a box:

```
<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>
```

The action of the form is defined by this line:

```
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.

Using HTTP POST with QXP Server Manager

HTTP POST with QuarkXPress Server Manager works the same way as HTTP POST with QuarkXPress Server (see "*Using HTTP POST with QXP Server*"), except that with QuarkXPress Server Manager, you must use UTF-8.

Getting started: Web services

The Web services interface is a collection of request classes. You can download the SDK WSDL class definitions from `http://<server>:<port>/quark/services/qxpsmsdk?wsdl` (replace `<server>` with the QuarkXPress Server Manager computer's IP address and `<port>` with the QuarkXPress Server Manager port number.

These classes can be chained together to form compound QuarkXPress Server requests. The sample applications (see "*Sample applications*") show how to use these classes to invoke a QuarkXPress Server command and manipulate the response.

For more information, see "*Using the Web interface*." 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 response. For more information, see the sample applications and "*Using the Web interface*."

- `QRequestContext` is the argument you pass to `QManagerSDKSvc`. This object contains settings which must be set once per request. Set all chained requests 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 a box's name and/or ID as the parameter name and the box's 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 QuarkXPress Server Manager. This object is returned by the `getErrorObject()` method.
- `QManagerScriptingSvc` is the Web services scripting interface.

You can extend the WIG to include your own XTensions software applications by simply modifying an XML file and redeploying the WIG Web service.

➡ To exclude empty tags in the request HTML, set the value of the appropriate variable to `null`.

➡ For Javadocs, WSDL schemas, and JSP samples, see the Welcome page that displays when you launch QuarkXPress Server Manager.

The following topics describe the general Web services classes.

QRequestContext

Description	An argument passed to `QManagerSDKSvc`. Contains settings that must be set once per request. All chained requests must be set inside the request context.		
Type	Web service data object		
Members	**Name**	**Types**	**Description**
	`documentName`	String	File or object name on which the command will be rendered.
	`serverName`	String	Server name. Default is NULL. Load balancer searches for the host itself in this case.
	`serverPort`	Integer	Port at which the desired server is listening.
	`userName`	String	Server admin username.
	`userPassword`	String	Server admin password.

	`maxRetries`	Integer	Max number of times to try executing the command before returning failure.
	`requestTimeout`	Integer	Max time out in milliseconds.
	`useCache`	Boolean	Indicates whether the cache should be checked for an existing result or if the command should be executed again.
	`responseAsURL`	Boolean	This value indicates whether the server should send the response as-is (text or binary) or store the response on the server and return its location as a URL. Because the object model works on SOAP, which can be slow when transferring large binary files, you might choose to set this value to "true" if you suspect that the response is going to be several megabytes or larger.
	`bypassFileInfo`	Boolean	Indicates whether file info should be fetched before executing the command.
	`context`	String	Context in which the command is being executed.
	`request`	`QRequest`	QuarkXPress Server request is instances of request objects 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. `QManagerSDKSvc` has a generic method named `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 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 a negative value is passed as timeout, the session never expires.
	`closeAlldocs`	Closes all open documents in the session without saving them. If the session does not exist, an error is returned. If an error occurs while closing the document, it is logged and the document is marked closed in the internal cache. No error is returned.		
		Parameter	**Type**	**Description**
		`sessionId`	String	Session whose documents are to be closed.
	`closeDoc`	Closes the specified document without saving it. If the session does not exist, an error is returned. If the document is not open, and error is returned. If the document is open in another session, an error is returned. If an error occurs while closing the document, it is logged and the document is marked closed in the internal cache. No error is returned.		
		Parameter	**Type**	**Description**
		`docName`	String	Document to be closed.
		`sessionId`	String	Session in which document was opened.
	`closeSession`	Closes the specified session. If the session does not exist, an error is returned. If any documents are still open in the session, an error is returned.		
		Parameter	**Type**	**Description**
		`sessionId`	String	Session to be closed.
	`getErrorObject`	Gets the internal error object. If you receive an exception from Web services, and that exception was caused by QuarkXPress Server or Manager (rather than a runtime exception such as a null pointer exception), you can 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, and so forth.		
	`getOpenDocs`	Gets all the open documents in the session. If the session does not exist, an error is returned.		
		Parameter	**Type**	**Description**
		`sessionId`	String	Session whose open documents are sought.
	`getOpenSessions`	Gets all open sessions.		
	`getPreferences`	Gets QuarkXPress Server preferences.		
	`setPreferences`	Sets QuarkXPress Server preferences.		
	`getXPressDOM`	Creates a DOM for the specified document.		

<table>
<tr><td><code>newDoc</code></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, an error is returned. If the session does not exist, an error is returned.
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td><code>docName</code></td><td>String</td><td>Document to be opened for modification. Provide the name only. You can proivde a relative path when you save the document.</td></tr>
<tr><td><code>jobJacketName</code></td><td>String</td><td>Name of the Job Jackets file to be used. The Job Jackets file is assumed to be already available on the QuarkXPress server computer.</td></tr>
<tr><td><code>jobTicketName</code></td><td>String</td><td>Name of the Job Ticket to be used.</td></tr>
<tr><td><code>host</code></td><td>String</td><td>The QuarkXPress Server instance that should be used for this document modification. If null, this value is supplied by the load balancer. If the indicated server is not an active registered server, an error is thrown.</td></tr>
<tr><td><code>port</code></td><td>Integer</td><td>The port for the server specified in the <code>host</code> parameter.</td></tr>
<tr><td><code>sessionId</code></td><td>String</td><td>Session in which the document should be opened.</td></tr>
</table></td></tr>
<tr><td><code>openDoc</code></td><td>Opens the specified document and keeps it open until further notice. If the document is already open, an error is returned. If the session does not exist, an error is returned.
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td><code>docName</code></td><td>String</td><td>Document (along with relative path if required) to be opened for modification.</td></tr>
<tr><td><code>host</code></td><td>String</td><td>QuarkXPress Server instance which should be used for this document modification. If null, this value is supplied by the load balancer. If the indicated server is not an active registered server, an error is thrown.</td></tr>
<tr><td><code>port</code></td><td>Integer</td><td>The port for the server specified in the <code>host</code> parameter.</td></tr>
<tr><td><code>sessionId</code></td><td>String</td><td>Session in which the document should be opened.</td></tr>
</table></td></tr>
<tr><td><code>processRequestEx</code></td><td>Executes the request context. If a session ID is specified, the document is kept open after the request is executed. If no session ID is specified, the request is</td></tr>
</table>

executed normally without keeping the document open. If the document is open in another session, an error is returned. If the document is marked dirty, 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
`reqContextObj`	`QRequestContext`	Request to be executed.
`sessionId`	String	Session in which the request should be executed. This value may be null. If a session ID is provided, the document is kept open. If no session ID is provided, the request is executed normally, as if processRequest had been called.

`saveAllDocs`

Saves all open documents in the session. The documents are saved one by one. If error occurs while saving a document, an error is returned immediately and the rest of the documents remain unsaved. If a document is marked dirty, 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 value is provided, copies of open documents with changes made so far are saved in the new location. The open documents are not saved but have all of the changes made so far.
`sessionId`	String	Session in which the document exists.

`saveDoc`

Saves the open document. If a document is marked dirty, 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 the same name that was used when opening or creating the document.
`newName`	String	New name of the document. If null, the document is saved with the 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. The open document is not saved but has all of 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 request objects (such as `PDFRenderRequest`). All request objects share some common data members, which are described below.		
Type	Web service data object		
Members	**Name**	**Types**	**Description**
	`request`	`QRequest`	QuarkXPress Server request that includes instances of request objects chained together.

RequestParameters

Description	Generic class for executing any request and for adding dynamic properties to a request.		
Type	Web service data object		
Members	**Name**	**Type**	**Description**
	`namespace`	String	Namespace of the request (for example, jpeg).
	`params`	`NameValueParam[]`	Parameter array for the specified request (for example, `jpegquality`).
Additional comments	You can use this class 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. That means the namespace provided here can be null. The parameters of this class can be used to parameterize a 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 a request. This class is specifically for requests that take a 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 the name/ID of the box.
	`textValue`	String	Text value of the box. (You can set either `textValue` or `streamValue`.)
	`streamValue`	byte[]	Stream value of the box. (You can set either `textValue` or `streamValue`.)
	`contentType`	String	The MIME content type of the parameter.

QContentData

Description	A response to a Web Services call to QuarkXPress Server.		
Type	Web service data object		
Members	**Name**	**Types**	**Description**
	`contentType`	String	The type of the response. For example, "text/xml" or "text/plain."
	`textData`	String	If the response type is text, this contains the text. Otherwise, this value is null.
	`responseURL`	String	If the `responseAsURL` parameter was set to "true" in the request, this contains the URL of the response. Otherwise, this value is null.
	`streamValue`	binary	If the response type is binary, this contains the byte array. Otherwise, this value is null.
	`encodingType`	String	If the response type is text, this value indicates the encoding of the text (for example, 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 a set of headers, this array contains the header response.
	`multipartResponse`	String	If the response returned by the server is multipart, this array contains the multipart response parts.

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 QuarkXPress Manager. This class is returned by the `getErrorObject` method.		
Type	`Exception`		
Members	**Name**	**Types**	**Description**
	`httpResponseCode`	String	HTTP response code.
	`managerErrorCode`	String	QuarkXPress Server Manager error code.
	`managerErrorMessage`	String	QuarkXPress Server Manager localized error message.
	`serverErrorCode`	String	QuarkXPress Server error code.
	`serverErrorMessage`	String	QuarkXPress Server response message.
	`serverExtendedMessage`	String	QuarkXPress Server extended error 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`
	`  //QuarkXPress Manager threw an exception and it 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 a 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, with passed arguments.		
		Parameter	**Type**	**Description**
		`id`	String	String ID.
		`function`	String	Function to execute.
		`arguments`	String[]	Arguments to pass to function.
	`executeScriptWithVars`	Executes a script with variables for the script to use.		
		Parameter	**Type**	**Description**
		`id`	String	Script ID.
		`variables`	`QScriptVar[]`	Variables to be used by script.
	`getAllScripts`	Gets all scripts saved with the system.		
	`getErrorObject`	Creates an error object from a error string.		
		Parameter	**Type**	**Description**
		`errorString`	String	Error string to use.
	`getScript`	Gets the script with the specified ID.		
		Parameter	**Type**	**Description**
		`id`	String	Script ID.
	`getScriptExecutionDetails`	Gets the runtime details of a script.		

		Parameter	**Type**	**Description**
		`scriptId`	String	Script ID.
	`getSupportedLanguages`	Gets the supported scripting languages.		
	`isLanguageSupported`	Checks whether a particular scripting language is supported.		
		Parameter	**Type**	**Description**
		`language`	String	Language to check.
	`updateScript`	Updates a script. If the script does not exist, this function adds it.		
		Parameter	**Type**	**Description**
		`script`	`QScript`	Script to update or add.

QXP Server Manager

The following topics are for people 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 was developed using interface-based programming and uses the Spring Framework to instantiate pluggable objects. When QuarkXPress Server Manager starts up, it reads the contents of a Spring context definition file named "ManagerContainerConfig.xml" and instantiates all of the beans listed in the file. QuarkXPress Server Manager then initializes by reading various configuration options from a file named "ManagerConfig.xml."

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 "cache.xml" file, which is located in Tomcat's `conf/Catalina/localhost` folder.

If QuarkXPress Server Manager needs to be hosted in another web server, you can write your own adapter or use `QDefaultContainerAdapterImpl` (which is 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 can be set using the Spring configuration file or the `setCacheFolderRelativePath` method.

Embedding QXP Server Manager

You can embed QuarkXPress Server Manager in applications (stand-alone or otherwise). 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, like so:

```
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 them. 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`. You can then insert this new handler class anywhere in the chain of responsibility pattern, starting with `QDocProviderImpl` and ending with `QHostRequestHandler`.

➡ Try not to change end points. In your handler implementation, handle your special flags, then either return a response after handling or pass the control to the successor for further handling.

Implementing a custom load balancer

To implement a custom load balancer, first implement the `com.quark.manager.lb.QLoadBalancer` interface. To use this interface, add a reference to "managerengine.jar" to your project.

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 so it can be displayed in the QuarkXPress Server Manager client.
Returns	Description of the load balancer.
`useFileInfo`	
Signature	`public Boolean useFileInfo();`
Description	Gets a flag that indicates whether the load balancer uses file information to decide on which host to use.
Returns	True if the `fileinfo` command should be fired before rendering, otherwise false.
`getAvailableHost`	
Signature	`public QHostProxy getAvailableHost(QHostProxy[] hosts, QCommand command);`
Description	Gets an 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.

Next:

1. Make a jar for the load balancer.
2. Deploy the jar to the following folder: `{Apache-Tomcat Home}\webapps\axis\WEB-INF\lib`
3. Configure "ManagerContainerConfig.xml" for bean mapping:
    1. Navigate to `{Apache-Tomcat Home}\webapps\axis\WEB-INF\classes` .

2 Open the "ManagerContainerConfig.xml" file and look for the XML tag bean whose `id` has the value `ConfigurationManager`.
3 Within that tag find the property name `availableLoadBalancers`.
4 In the `<list>` tag, add the following: `<ref bean={your newbeanID}/>`
5 Above this `ConfigureManager` tag, define the bean ID as your new bean ID: `<bean id={your newbeanID} class={yourLoadBalancerClass}/>`
6 Restart the Tomcat server.
7 Log on with the QuarkXPress Server Manager client and choose **Global Setting > Load Balancer Method > Choose Load Balancer**.
8 Locate your new load balancer method, then click **Save**.

Generating a custom client SDK class

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

Adding new SDK classes

To add new SDK classes:

1 Modify "ManagerSDK.xml" (located in `Server/utilities/`) to reflect changes in "Modifier.dtd."

2 Using a different folder, create backups of "managersdkro.jar," "managerdomgenerator.jar," and "managerrequestserializer.jar." All of these files are located in `<Installation Folder>/Server/apache-tomcat-5.5.16/webapps/quark/WEB-INF/lib`.

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, then execute `ClientSDKRequestObjectGenerator` again. When the process completes successfully, "managersdkro.jar", "managerdomgenerator.jar", and "managerrequestserializer.jar" are regenerated in the `webapps/quark/WEB-INF/lib` folder. Check the timestamps to verify that the files are new.

4 Launch QuarkXPress Server Manager.

5 Modify `Server/utilities/deploy_sdk.wsdd` to add the bean mapping for the newly generated class. To make change tracking easier, position the mapping within the classes to match the position of any changes made to the DTD or XML.

6 Edit `Server/utilities/deploy.sh` (Mac OS) or `Server/utilities/deploy.bat` (Windows) to change the port number where QuarkXPress Server Manager is running (if it is different from 8090).

7 Execute `Server/utilities/deploy.sh` (Mac OS) or `Server/utilities/deploy.bat` (Windows) and check for errors.

8 Open a Web browser and enter the following URL: `http://localhost:8090/quark/services/qxpsmsdk?wsdl`. Verify that the class you just added is visible in WSDL.

Generating new SDK stubs

To generate new stubs:

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, then execute `Server/utilities/stub.sh` or `Server/utilities/stub.bat` again. If the process succeeds, "managerwebservicestubs.jar" is generated in the `Server/utilities` directory. You can use these Java stubs in Java applications that communicate with QuarkXPress Server Manager.

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, then either refresh the Web service reference or remove the Web service reference and then add it again.

Understanding ManagerSDK.xml

"ManagerSDK.xml" is used to generate client SDK classes for QuarkXPress Server requests. Each element in "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 the 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, as follows.

- `<Class>`: One element for each SDK class generated. The class generated is derived from `QRequest`. Attributes are:
  - `name:` The name of the generated class.
  - `namespace:` The namespace recognized by QuarkXPress Server when this request class is translated into a QuarkXPress Server request.
  - `description:` A description of the class. Unless this value is null, the 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:` Determines how the class is serialized. The valid values are:
    - `nameValue` indicates 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 `JPEGRenderRequest` and `ModifierStreamRequest`.)

    - `xml` indicates that the class should be serialized as XML with the class name or alias as the element value. All of 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 must be an array of `QRequest`-derived classes.
    - `mixed` indicates that the class should be serialized as XML with the class name or alias as the element. All the primitive fields of the class are serialized as attributes. If the field is a subclass of `QRequest`, it is serialized as a child element and processed recursively. If the field is an array, it must be an array of `QRequest`-derived classes.
    - `attribute` indicates that the class should be serialized as XML with the class name or alias as the element. The class must be primitive. All such fields must be serialized as attributes of the element. Also, "value" fields must be serialized as values of the element. Valid only if the parent class has a `serializeAs` value of "xml" or "mixed."

- `<Attribute>`: One element for each class field.
    - `name:` The name of the generated class variable.
    - `accessor:` The name of the accessor that gets the property. If this value is null, the default accessor name is used. The default name is "get" + CamelCase(name) (for example, if the name of the property is "quality," the default accessor method is `getQuality`).
    - `mutator:` The name of the accessor that sets the property. If this value is null, the default mutator name is used. The default name is "set" + CamelCase(name) (for example, if the name of the property is "quality," the default mutator method is `setQuality`).
    - `description:` A description of the attribute. Unless this value is null, the description is included in variable headers and accessor and mutator headers and is included in generated API docs.
    - `type:` The type of the class variable. If this value is null, the default type (`string`) is used. If this is not a primitive data type, it should be defined as a separate `Class` element. If this attribute 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 this attribute has a null value, during serialization the value of the field should be set in the reference class provided. Note that the reference class should be declared using "type=reference" as explained above.
    - `readonly:` If this value is `true`, this field is for read-only purposes and should be ignored during serialization.
    - `hidden:` If this value is `true`, this field should be generated as a private variable. As such, it would not be included in WSDL.

- `deprecated:` If this value is `true`, this field has been deprecated, should not be used, is not supported, and will be removed in a future version of QuarkXPress Server.
- `cdata:` If this value is `true`, the value of this field is to be wrapped in a `cdata` section before being sent 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>` is 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.

Although you can write scripts with almost any script-editing application, you might want to use the QuarkXPress Server Manager Scripting Environment, which ships with QuarkXPress Server Manager. You can use this application to write scripts, run scripts manually, and schedule scripts to start and end at specific times or until specific conditions are met.

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

When writing scripts, you can directly access the functions named `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 are used.

By default, the scripting environment 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." Scripts you save in this way can also be executed remotely using Web services; for more information, see "*QManagerScriptingSvc*."

Keep document open (sessions)

In early 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, 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. This script opens a session, opens a QuarkXPress project, and then modifies a text box until the text in it fits.

Using the Web interface

The topics below describe the features available via the QuarkXPress Server Web interface. The topics covered here include the following:

- *Render types* are namespaces you can use to return a QuarkXPress project in a specified file format.
- *Render modifiers* let you control which parts of a project are rendered and set the scale of the returned renderings.
- *Content modifiers* let you alter the content and formatting of boxes in layouts without using the XML modify parameter.
- *XML modify* lets you modify QuarkXPress projects using XML.
- The `xml` namespace deconstructs a project according to the Modifier DTD. The `construct` namespace lets you turn an XML representation of a QuarkXPress project back into a QuarkXPress project.
- *Administrative request handlers* let you change the behavior of QuarkXPress Server.

➡ QuarkXPress Server uses case-sensitive XML.

Understanding rendering

Rendering is the process in which QuarkXPress Server opens a QuarkXPress project, transforms it into a different format (the *render type*), and then sends a response to the requestor. Depending on the type of rendering operation, the response may be a message or a rendered file.

For information on how to submit a render request, see "*Getting started: HTTP*."

Alerts	Cannot open this document type. Please select a QuarkXPress document or template.	HTTP Error #500 This alert displays if you try to render a file that is not a QuarkXPress project that exists in the document pool.

	The file system document pool is not enabled.	HTTP Error #404 This alert displays if the file system document pool is disabled. *What to do*: In QuarkXPress Server, choose **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box, then check **Enable File System Document Pool**.
	File not found	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you try to render a project that does not exist.
	I/O error trying to read or write to disk.	HTTP Error #500 QuarkXPress Server Error #–36 This alert displays if QuarkXPress Server is running on Windows and a shared network folder was selected as the document pool, but the folder is no longer shared. *What to do*: In QuarkXPress Server, choose **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box, then choose a shared folder.
	Cannot find required volume or folder.	HTTP Error #404 QuarkXPress Server Error #–35 This alert displays if QuarkXPress Server is running on Mac OS and a shared network volume was selected as the document pool, but the volume is no longer shared. *What to do*: In QuarkXPress Server, choose the menu option **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box, then choose a shared volume.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`	
Notes	There are two ways to specify a render format: 1. Enter the render type directly in the browser address field:`http://localhost:8080/pdf/project.qxp`. 2. In QuarkXPress Server, choose **QuarkXPress Server > Server Configuration** to display the **Server Configuration** dialog box, then choose the default render type from the **Default Render Type** drop-down menu in the **Server** tab.	

Understanding render types

Render types are namespaces you can use to return a QuarkXPress project in a specified file format. The topics covered here include the following:

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`
swf	Returns a SWF file.	`SWFRenderRequest`

➡ The default render type is JPEG.

➡ Developers can implement additional rendering formats through server XTensions software.

eps

The `eps` render type returns an EPS rendering of a page or spread.

Namespace	`EPS`		

Parameters	`outputstyle`	stylename	Lets you specify an output style. To use a named output style, use the name of that output style. For example:
			`http://localhost:8080/pdf/sample.qxp?outputstyle=stylename`
			To use settings that have been captured with the **Capture Settings** in the QuarkXPress **Print** dialog box, use `document`. For example:
			`http://localhost:8080/pdf/sample.qxp?outputstyle=document`

	`epsformat`	color | dcs2	Lets you specify an EPS format. The default value is `color`.
	`epspreview`	tiff | none	Lets you include or omit a TIFF preview. The default value is `tiff`.

	epsdata	ascii | binary | clean8bit	Lets you specify a data type for the EPS file. The default value is clean8bit.
	epstransparent	1 | 0 | true | false | yes | no	Lets you specify whether the EPS can include transparent areas.
Render modifier parameters	page	Integer	Lets you specify a page.
	produceblankpages	1 | 0 | true | false | yes | no	Lets you specify whether to render blank pages.
	scale	Float .1 to 6.92 for Windows .1 to 8 on Mac OS	Lets you specify a scaling percentage. The valid values are from .1 (10%) to 8 (800%) on Mac OS or 6.92 (692%) on Windows.
	spread	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	layout	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
	downloadlayoutFonts	1 | 0 | true | false | yes | no	Lets you specify whether to download all fonts used in the layout and all system fonts.
	downloadImportedPdfEpsFonts	1 | 0 | true | false | yes | no	Lets you specify whether to download all fonts required by imported PDF and EPS files.
Response	An EPS file.		
Alerts	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you submit a render request with the pages or box parameter.	
	This Output Style does not exist.	This alert displays if you specify a nonexistent output style.	
	This Output Style cannot be used with this render type.	This alert displays if you specify an output style that is incompatible with this render type.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`

Example, object model	Request object name: `EPSRenderRequest`
	`//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);`

Notes	You can specify an output style and set additional local parameters of that output style. For example, if no bleed setting is specified in the output style named "mystylename", you can specify a bleed setting with a URL like the following: `http://localhost:8080/eps/sample.qxp?outputstyle=mystylename?bleed=symmetric` You can override settings in an output style. For example, if an asymmetric bleed is specified in the output style named "mystylename," you could override it with the same URL. If you do not specify an EPS output style, the default EPS output style is used.

jpeg

The `jpeg` render type returns a JPEG rendering of a page or spread.

Namespace	`JPEG`		
Parameters	`jpegquality`	1 | 2 | 3 | 4	Lets you specify the image quality of a rendered JPEG image. The valid values are: 1 (highest quality), 2 (high quality), 3 (medium quality), and 4 (lowest quality). The default value is 1.
	`upadateimage`	true | false	Lets you specify whether to return modified pictures in the response or not. If set to `false`, modified pictures are not returned. If set to `true`, modified pictures are returned. The default value is `true`.

	`pasteboard`	true | false	Lets you specify whether to display pasteboard items. Works only with `spread` parameter. The default value is `true`. For example: `http://localhost:8080/jpeg/document.qxp?spread=1&pasteboard=true`
Render modifier parameters	`boxes`	String	Lets you request multiple boxes.
	`page`	Integer	Lets you request a single page.
	`scale`	Float .1 to 6.92 for Windows .1 to 8 on Mac OS)	Lets you specify a scaling percentage. The valid values are from .1 (10%) to 8 (800%) on Mac OS or 6.92 (692%) on Windows.
	`box`	String	Lets you request a single box.
	`spread`	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
Response	A JPEG file.		
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`
	`//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);`		

literal

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

Namespace	`literal`	
Response	The requested file returned in the HTTP response.	
Alerts	Incorrect administration realm username and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in theQuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/10/2005 10:04:52 — literal/Test1.doc — Type: application/vnd.Quark.QuarkXPress — Size: 800768 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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);`	

pdf

The `pdf` render type returns a PDF rendering of a project.

Namespace	`PDF`		

Parameters	`outputstyle`	stylename, document	Lets you specify an output style. To use a named output style, use the name of that output style. For example:
			`http://localhost:8080/pdf/sample.qxp?outputstyle=stylename`
			To use settings that have been captured with the **Capture Settings** in the QuarkXPress **Print** dialog box, use `document`. For example:
			`http://localhost:8080/pdf/sample.qxp?outputstyle=document`

`title`	String	Lets you specify the title of the PDF file.
`subject`	String	Lets you specify the subject of the PDF file.
`author`	String	Lets you specify the author of the PDF file.
`keywords`	String	Lets you specify keywords for the PDF file.
`includehyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to include hyperlinks in the PDF file.
`exportlistsashyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to export lists as hyperlinks. To use this parameter, you must set `includehyperlinks` to true.
`exportindexesashyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to export the index as hyperlinks. To use this parameter, you must set `includehyperlinks` to true.
`exportlistsasbookmarks`	1 | 0 | true | false | yes | no	Lets you specify whether to export lists as bookmarks. To use this parameter, you must set `includehyperlinks` to true.
`mode`	composite or separations	Lets you specify whether the PDF file is a composite or includes separations.
`printcolors`	cmyk, rgb, grayscale, cmykandspot, asis	Lets you specify the color space of the PDF file. This option is available only when `mode` is set to `composite.`
`plates`	converttoprocess, processandspot, inripseps	Lets you specify a separation method. This option is available only when `mode` is set to `separations`.
`produceblankpages`	1 | 0 | true | false | yes | no	Lets you specify whether to include blank pages. This option is available only when `mode` is set to `composite.`
`useopi`	1 | 0 | true | false | yes | no	Lets you specify whether to use OPI.
`images`	includeimages, omittiff, omittiffandeps	Lets you specify whether to include TIFF and EPS images from an OPI server.
`registration`	off, centered, offcenter	Lets you include, omit, and configure registration marks.
`offset`	0–30 (in points)	Lets you specify the offset of registration marks.
`bleed`	pageitemsonly, symmetric	Lets you specify a bleed type.
`offsetbleed`	0–6 (in inches)	Lets you specify a bleed offset to use. This option is available only when `bleed` is set to `symmetric`.

spreads	1 | 0 | true | false | yes | no	Lets you specify whether to output spreads.
lowresolution	1 | 0 | true | false | yes | no	Lets you request a low-resolution (36 dpi) PDF.
colorimagedownsample	9–2400	Lets you specify the resolution of color images.
grayscaleimagedownsample	9–2400	Lets you specify the resolution of grayscale images.
monochromeimagedownsample	9–2400	Lets you specify the resolution of monochrome images.
colorcompression	true | false	Lets you specify whether medium-quality manual JPEG compression should be applied to color images.
grayscalecompression	true | false	Lets you specify whether medium-quality manual JPEG compression should be applied to grayscale images.
monochromecompression	true | false	Lets you specify whether ZIP compression should be applied to monochrome images.
pdffile	String	Lets you specify the PDF name. This option is available only when **PDF to Folder** is selected in QuarkXPress Server PDF preferences.
psfile	String	Lets you specify the PostScript file name. This option is available only when **PostScript for later Distilling** is selected in QuarkXPress Server PDF preferences.
thumbnail	bw | color	Lets you embed a thumbnail in the PDF file.
mode	composite | separations	Lets you specify the PDF file's color mode.
fontdownload	yes | no	Lets you turn font download on or off. You cannot specify which fonts are downloaded.
layers	String	Lets you specify which layers should be included, as a comma-separated list.
transparencyres	Integer value from 36 to 3600	Lets you specify the resolution for flattened content.
verification	pdfx1a | pdfx3	Lets you use PDF/X–1a or PDF/X–3 verification.
separate	yes | no	Lets you specify whether to output each page as a separate file.
produceblankplates	yes | no	Lets you specify whether to include blank plates.
download	Boolean 1 | 0 | true | false	When download is true, the browser always displays a dialog box that lets the end user save the returned file, even if the browser can display it.

			When `download` is false, the browser attempts to display the returned file. If the browser cannot display the file, it lets the end user save the returned file. The default value is false.
	`layoutstart`	Integer	Lets you specify the number of the first layout to render when you render multiple layouts as separate PDF files. PDF files are saved at the location specified in QuarkXPress Server preferences (**Server/QuarkXPress Server > Preferences > PDF** pane > **PDF to Folder**). The first layout in a project is layout 0. For example: `http://localhost:8080/pdf/multilayout.qxp?layoutstart=0&layoutend=3`
	`layoutend`	Integer	Lets you specify the number of the last layout to render when you render multiple layouts as separate PDF files. PDF files are saved at the location specified in QuarkXPress Server preferences (**Server/QuarkXPress Server > Preferences > PDF** pane > **PDF to Folder**). The first layout in a project is layout 0. For example: `http://localhost:8080/pdf/multilayout.qxp?layoutstart=0&layoutend=3`
Render modifier parameters	`page`	Integer	Lets you specify a single page.
	`pages`	String (page range)	Lets you specify a range of pages.
	`spread`	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
	`spreads`	Boolean 1 | 0 | true | false | yes | no	Lets you specify that the output use spreads.
Response	A PDF file.		
Alerts	This page range is invalid	HTTP Error #500 QuarkXPress Server Error #147 This alert displays if you try to render an invalid page range.	
	No file produced. The project requested contains only blank pages.	HTTP Error #500 This alert displays if you try to render a a project that contains only blank pages.	
	This Output Style does not exist.	This alert displays if you specify a nonexistent output style.	
	This Output Style cannot be used with this render type.	This alert displays if you specify an output style that is incompatible with this render type.	

Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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 "sample.qxp" as a PDF with a symmetric bleed:
	`http://localhost:8080/pdf/sample.qxp?bleed=symmetric&offsetbleed=2`
	This URL renders a PDF in which color images are downsampled to a resolution of 300 dpi and manual medium-quality JPEG compression is applied:
	`http://localhost:8080/pdf/sample.qxp?colorimagedownsample=300&colorcompression=true`

Example, object model	Request object name: `PDFRenderRequest`
	`//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.`
	`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);`
	For more information about the object model, see the samples.

Notes	There are three ways to generate PDF files with QuarkXPress Server. You can generate a PDF file in QuarkXPress Server and return it to the end user, generate the PDF in QuarkXPress server and save it to a folder on the server computer, or generate a PostScript file for later distilling and save it to a folder on the server computer. To choose one of these output methods in QuarkXPress Server, choose **QuarkXPress Server > Preferences**, click **PDF** in the list on the left, and then click **PDF Direct**, **PDF to Folder**, or **PostScript for Later Distilling**. If you choose either of the last two options, click **Browse** and navigate to the target folder, then choose an option from the **Default Name** drop-down menu.

	You can specify an output style and set additional local parameters of that output style. For example, if no bleed setting is specified in the output style named "mystylename", you can specify a bleed setting with a URL like the following: `http://localhost:8080/pdf/sample.qxp?outputstyle=mystylename&bleed=symmetric` You can override settings in an output style. For example, if an asymmetric bleed is specified in the output style named "mystylename," you could override it with the same URL. If you do not specify a PDF output style, the default PDF output style is used.

png

The `png` render type returns a PNG rendering of a page or spread.

Namespace	`PNG`		
Parameters	`pngcompression`	1 | 2 | 3 | 4	Lets you specify the compression of a PNG response. The valid values are: 1 (lowest compression), 2 (medium compression), 3 (high compression), and 4 (highest compression). The default value is 1.
	`upadateimage`	true | false	Lets you specify whether to return modified pictures in the response or not. If set to `false`, modified pictures are not returned. If set to `true`, modified pictures are returned. The default value is `true`.
Render modifier parameters	`boxes`	String	Lets you request multiple boxes.
	`page`	Integer	Lets you specify a single page.
	`scale`	Float .1 to 6.92 for Windows .1 to 8 on Mac OS	Lets you specify a scaling percentage. The valid values are from .1 (10%) to 8 (800%) on Mac OS or 6.92 (692%) on Windows.
	`box`	String	Lets you request a single box.
	`spread`	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
Response	A PNG file.		
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`
	`//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);`

postscript

The `postscript` render type returns a PostScript rendering of a project.

Namespace	`PostScript`		

Parameters	`prntbleed`	Page | asym, clip<Boolean>, top<float>, bottom<float>, left<float>, right<float> | sym, clip<Boolean>, amount<float>	Lets you specify bleed values for a page.
			To specify an asymmetric bleed, use the following format:
			`prntbleed=asym,clip,top,bottom, left, right`
			The `clip` value is Boolean (yes/no). The `top`, `bottom`, `left`, and `right` values are float values. For example:
			`http://localhost:8080/postscript/Sample.qxp?prntbleed=asym,true,1,2,2,1`
			The above example results in an asymmetric bleed of 1 on the top, 2 on the bottom, 2 on the left, and 1 on the right.
			To specify a symmetric bleed, use the following format:
			`prntbleed=sym,clip,amount`
			The `clip` value is Boolean (yes/no). The `amount` value is a float value. For example:
			`http://localhost:8080/postscript/Sample.qxp?prntbleed=sym,true,1`
			The above example results in a symmetric bleed of 1 on all sides. Default: `prntbleed=sym,yes,0`

	`outputstyle`	stylename, document	Lets you specify an output style. To use a named output style, use the name of that output style. For example:
			`http://localhost:8080/postscript/sample.qxp?outputstyle=stylename`
			To use settings that have been captured with the **Capture Settings** in the QuarkXPress **Print** dialog box, use `document`. For example:
			`http://localhost:8080/postscript/sample.qxp?outputstyle=document`

Render modifier parameters	`page`	Integer	Lets you specify a single page.
	`pages`	String (page range)	Lets you specify a range of pages.
	`spread`	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
Response	A PostScript file.		
Alerts	This page range is invalid.	HTTP Error #500 QuarkXPress Server Error #147 This alert displays if you try to render an invalid page range.	
	No file produced. The document requested contains only blank pages.	HTTP Error #500 This alert displays if you try to render a a project that contains only blank pages.	
	PostScript printer mapped to file not found	HTTP Error #500 This alert displays if the PostScript printer or driver is not set to **Print to File**.	
	This Output Style does not exist.	This alert displays if you specify a nonexistent output style.	
	This Output Style cannot be used with this render type.	This alert displays if you specify an output style that is incompatible with this render type.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`
	`//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);`

Notes	To create a PostScript file, you must have a PostScript driver on the server computer. You can specify an output style and set additional local parameters of that output style. For example, if no bleed setting is specified in the output style named "mystylename", you can specify a bleed setting with a URL like the following: `http://localhost:8080/eps/sample.qxp?outputstyle=mystylename&bleed=symmetric` You can override settings in an output style. For example, if an asymmetric bleed is specified in the output style named "mystylename," you could override it with the same URL. If you do not specify a PostScript-compatible output style, the default PostScript-compatible output style is used.

ppml

The `ppml` render type returns a PPML rendering of a page or spread.

Namespace	`PPML`		

Parameters	`outputstyle`	`stylename`	Lets you specify an output style. To use a named output style, use the name of that output style. For example:
			`http://localhost:8080/ppml/sample.qxp?outputstyle=stylename`
			To use settings that have been captured with the **Capture Settings** in the QuarkXPress **Print** dialog box, use `document`. For example:
			`http://localhost:8080/ppml/sample.qxp?outputstyle=document`

	`path`	String	Lets you specify a location for PPML output. For example: `path=C:\output`

<table>
<tr><td rowspan="3">Render modifier parameters</td><td>thexmldoc</td><td>XML</td><td>Lets you supply XML that matches the placeholders in the project.</td></tr>
<tr><td>paginate</td><td>XML</td><td>Lets you supply XML that matches the placeholders in the project. This parameter creates a new layout containing the imported and formatted XML.</td></tr>
<tr><td>layout</td><td>String</td><td>Lets you specify a layout by name or ID. The first layout is Layout 1.</td></tr>
<tr><td>Response</td><td colspan="3">PPML output.</td></tr>
<tr><td rowspan="4">Alerts</td><td>The renderer for this image type has no way of rendering the desired objects.</td><td colspan="2">HTTP Error #406
This alert displays if you submit a render request with the pages or box parameter.</td></tr>
<tr><td>This Output Style does not exist.</td><td colspan="2">This alert displays if you specify a nonexistent output style.</td></tr>
<tr><td>This Output Style cannot be used with this render type.</td><td colspan="2">This alert displays if you specify an output style that is incompatible with this render type.</td></tr>
<tr><td>The file path is invalid.</td><td colspan="2">HTTP Error #500
This alert displays if you specify an invalid path parameter.</td></tr>
<tr><td>Logs</td><td colspan="3">If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example:
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.</td></tr>
<tr><td>Example, GET URL</td><td colspan="3">http://localhost:8080/ppml/sample.qxp?
paginate=file:MacintoshHD:file.xml
&path=C:\abc&includefont=true</td></tr>
<tr><td>Example, object model</td><td colspan="3">Request object name: PPMLRenderRequest

//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);</td></tr>
</table>

	`//STEP3: Create the WIG service and`
	`//call the processRequest() API`
	`QManagerSDKSvcServiceLocator serviceLocator =`
	`  new QManagerSDKSvcServiceLocator();`
	`QManagerSDKSvc service = serviceLocator.getqxpsmsdk();`
	`sdk.QContentData data =`
	`  service.processRequest(requestCtx);`

Notes	You can specify an output style and set additional local parameters of that output style. For example, if no bleed setting is specified in the output style named "mystylename", you can specify a bleed setting with a URL like the following: `http://localhost:8080/ppml/sample.qxp?outputstyle=mystylename&bleed=symmetric` You can override settings in an output style. For example, if an asymmetric bleed is specified in the output style named "mystylename," you could override it with the same URL. If you do not specify a PPML output style, the default PPML output style is used.

qcddoc

The `qcddoc` render type returns a QuarkCopyDesk article.

Namespace	`qcddoc`		
Parameters	`article`	String	Lets you specify which article in a project to render. For example: `http://localhost:8080/qcddoc/abc.qxp?article=article1`
	`component`	String	Lets you specify which component in an article to render. For example: `http://localhost:8080/copydesk/abc.qcd?component=comp1`
	`format`	lightweight | fullfeatured	Lets you render an article in lightweight or full-featured format. For example: `http://localhost:8080/qcddoc/abc.qxp?article=article1&format=fullfeatured`

	`saveastemplate`	true | false	Lets you save a copy of an article that was created in QuarkCopyDesk as a template. The default value is `true`. For example:
			`http://QXPServer8:8080/saveas/qcddoc/article.qcd?saveastemplate=true`
			You can also use this parameter to save a copy of a template as an article. For example:
			`http://QXPServer8:8080/saveas/qcddoc/template.qct?saveastemplate=false`

	`includepagepicture`	true | false | 1 | 0	Lets you include a page picture when you export an article from a QuarkXPress layout. Valid options are: `picformat` (`embedded` or `separate`) `quality` (`blackandwhite` or `color`) `picdpi` (`72`, `144`, or `200`) `spreadrange` (`all` or `first`)

			For example:
			`http://localhost:8080/saveas/qcddoc/4.qxp?includepagepicture=1&quality=blackandwhite&picdpi=144&spreadrange=first`
			`http://localhost:8080/saveas/qcddoc/PagePicture.qxp?includepagepicture=true`

Render modifier parameters	`modify`	`XML`	Lets you modify the article with XML. For more information, see "*Using XML modify*."
Response	A QuarkCopyDesk article.		
Alerts	There is no box with the specified identifier.	This alert displays if the box corresoponding to a referenced component does not exist.	
	The number of characters in the article name can't be greater than max limit.	This alert displays if an article name is longer than 32 characters.	
	The article/component name is not unique.	This alert displays if you create or change the name of an article or component so that it is the same as the name of an existing article or component.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`		

qxpdoc

The `qxpdoc` render type returns a QuarkXPress project.

Namespace	`qxpdoc`		
Parameters	`qxpdocver`	7 | 8 | korean6 | japanese6	Indicates the QuarkXPress version format to use. For example: `http://localhost:8080/qxpdoc/construct/project1.qxp?qxpdocver=7`
	`upadateimage`	true | false	Lets you specify whether to return modified pictures in the response or not. If set to `false`, modified pictures are not returned. If set to `true`, modified pictures are returned. The default value is `true`.
	`saveastemplate`	true | false	Lets you save a copy of a project as a template. The default value is `true`. For example: `http://localhost:8080/saveas/qxpdoc/project.qxp?saveastemplate=true`

			You can also use this parameter to save a copy of a template as a project. For example: `http://localhost:8080/saveas/qxpdoc/template.qpt?saveastemplate=false`
Render modifier parameters	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
Response	A QuarkXPress project.		
Alerts	QuarkXPress document return is disabled.	HTTP Error #500 This alert displays if **Disable QuarkXPress Document Return** is checked in the **Server** tab of the QuarkXPress Server **Server Configuration** dialog box (**QuarkXPress Server > Server Configuration**).	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you submit a `qxpdoc` render request with the `page`, `pages`, `box`, or `spread` parameter.	
	Cannot save a QuarkXPress Project down to an earlier version.	HTTP Error #500 This alert displays if you attempt to save a QuarkXPress 6.x project to an earlier version of QuarkXPress with the `qxpdocver` parameter.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`
	`//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);
```

screenpdf

The `screenpdf` render type returns a low-resolution PDF rendering of a project. This render type overrides the setting in the **PDF Workflow** area of the **PDF** pane in the QuarkXPress Server **Preferences** dialog box (**QuarkXPress Server > Preferences**) and always sends the PDF file to the browser.

Namespace	`Screenpdf`		

Parameters	`outputstyle`	`stylename`	Lets you specify an output style. To use a named output style, use the name of that output style. For example:
			`http://localhost:8080/screenpdf/sample.qxp?outputstyle=stylename`
			To use settings that have been captured with the **Capture Settings** in the QuarkXPress **Print** dialog box, use `document`. For example:
			`http://localhost:8080/screenpdf/sample.qxp?outputstyle=document`

	`title`	String	Lets you specify the title of the PDF file.
	`subject`	String	Lets you specify the subject of the PDF file.
	`author`	String	Lets you specify the author of the PDF file.
	`keywords`	String	Lets you specify keywords of the PDF file.
	`includehyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to include hyperlinks in the PDF file.
	`exportlistsashyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to export lists as hyperlinks. To use this parameter, you must set `includehyperlinks` to true.
	`exportindexesashyperlinks`	1 | 0 | true | false | yes | no	Lets you specify whether to export the index as hyperlinks. To use this parameter, you must set `includehyperlinks` to true.
	`exportlistsasbookmarks`	1 | 0 | true | false | yes | no	Lets you specify whether to export lists as bookmarks. To use this parameter, you must set `includehyperlinks` to true.
	`mode`	composite or separations	Lets you specify whether the PDF file is a composite or includes separations.
	`printcolors`	cmyk, rgb, grayscale, cmykandspot, asis	Lets you specify the color space of the PDF file. This option is available only when `mode` is set to `composite.`
	`plates`	converttoprocess, processandspot, inripseps	Lets you specify a separation method. This option is available only when `mode` is set to `separations`.

produceblankpages	1 | 0 | true | false | yes | no	Lets you specify whether to include blank pages. This option is available only when mode is set to composite.
useopi	1 | 0 | true | false | yes | no	Lets you specify whether to use OPI.
images	includeimages, omittiff, omittiffandeps	Lets you specify whether to include TIFF and EPS images from an OPI server.
registration	off, centered, offcenter	Lets you include, omit, and configure registration marks.
offset	0–30 (in points)	Lets you specify the offset of registration marks.
bleed	pageitemsonly, symmetric	Lets you specify a bleed type.
offsetbleed	0–6 (in inches)	Lets you specify a bleed offset to use. This option is available only when bleed is set to symmetric.
spreads	1 | 0 | true | false | yes | no	Lets you specify whether to output spreads.
lowresolution	1 | 0 | true | false | yes | no	Lets you request a low-resolution (36 dpi) PDF.
colorimagedownsample	9–2400	Lets you specify the resolution of color images.
grayscaleimagedownsample	9–2400	Lets you specify the resolution of grayscale images.
monochromeimagedownsample	9–2400	Lets you specify the resolution of monochrome images.
colorcompression	true | false	Lets you specify whether medium-quality manual JPEG compression should be applied to color images.
grayscalecompression	true | false	Lets you specify whether medium-quality manual JPEG compression should be applied to grayscale images.
monochromecompression	true | false	Lets you specify whether ZIP compression should be applied to monochrome images.
pdffile	String	Lets you specify the PDF name. This option is available only when **PDF to Folder** is selected in QuarkXPress Server PDF preferences.
psfile	String	Lets you specify the PostScript file name. This option is available only when **PostScript for later Distilling** is selected in QuarkXPress Server PDF preferences.
thumbnail	bw | color	Lets you embed a thumbnail in the PDF file.
mode	composite | separations	Lets you specify the PDF file's color mode.

	`fontdownload`	yes | no	Lets you turn font download on or off. You cannot specify which fonts are downloaded.
	`layers`	String	Lets you specify which layers should be included, as a comma-separated list.
	`transparencyres`	Integer value from 36 to 3600	Lets you specify the resolution for flattened content.
	`verification`	pdfx1a | pdfx3	Lets you use PDF/X–1a or PDF/X–3 verification.
	`separate`	yes | no	Lets you specify whether to output each page as a separate file.
	`produceblankplates`	yes | no	Lets you specify whether to include blank plates.
Render modifier parameters	`page`	Integer	Lets you specify a single page.
	`pages`	String (page range)	Lets you specify a range of pages.
	`spread`	Integer	Lets you specify a spread. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
	`layout`	String	Lets you specify a layout by name or ID. The first layout is Layout 1.
	`spreads`	Boolean (1 | 0 | true | false | yes | no)	Lets you specify that the output use spreads.
Response	A screen-resolution PDF file		
Alerts	This page range is invalid.	HTTP Error #500 QuarkXPress Server Error #147 This alert displays if you try to render an invalid page range.	
	No file produced. The document requested contains only blank pages.	HTTP Error #500 This alert displays if you try to render a a project that contains only blank pages.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`		

Example, object model	Request object name: `ScreenPDFRenderRequest`
	`//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);
```

swf

The `swf` render type returns a SWF (Flash) rendering of a Print layout or an Interactive layout.

Namespace	SWF		
Parameters	`version`	swf6 | swf7	Lets you specify the minimum compatible version of Flash Player.
	`layout`	string	Lets you specify a layout by name or ID. The first layout is Layout 1.
	`page`	string	Lets you specify a single page.
	`pages`	string	Lets you specify a range of pages
	`fullscreen`	true | false	Lets you specify whether the SWF file should run in full-screen mode by default.
	`embedallfonts`	true | false	Lets you indicate whether to include any fonts that are necessary to correctly render text in Text Box objects within the exported SWF file
	`compressswf`	true | false	Lets you specify whether to compress the exported file.
	`compressaudio`	true | false	Lets you specify whether to compress audio in the exported file.
	`jpegquality`	1–100	Lets you specify the quality of JPEG images in the exported file, with 100 being highest quality.
	`download`	true | false	When `download` is true, the browser always displays a dialog box that lets the end user save

			the returned file, even if the browser can display it. When `download` is false, the browser attempts to display the returned file. If the browser cannot display the file, it lets the end user save the returned file. The default value is false.
	`spreads`	Boolean 1 | 0 | true | false | yes | no	Lets you specify that the output use spreads. Applicable only to Print layouts.
Response	A single SWF file or a multipart reply containing an SWF file and a set of support files. A multipart reply can result if you render an Interactive layout that uses external assets -- for example, if the layout includes a Video object containing a video file that has been specified by choosing **Choose** from a drop-down menu. In this case, the reply includes a "RelativePath" parameter for each file, and each file is saved in the directory indicated by this "RelativePath" parameter You can use the `saveas` request handler to save the files that are included in a multipart reply in a particular directory, using the relative folder hierarchy that was specified in the multipart reply.		
Alerts	The requested page does not exist.	This alert displays if you try to render a nonexistent page.	
	This page range is invalid.	This alert displays if you try to render an invalid page range.	
	The requested layout does not exist.	This alert displays if you try to render a nonexistent leyout.	
	Unknown swf player version.	This alert displays if you specify an invalid `version` parameter.	
	Value of jpegquality is outside range, valid values are 1 - 100.	This alert displays if you specify an invalid `jpegquality` parameter.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/2/2005 17:17:17 — swf/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 Interactive layout named "Presentation" in the "sample.qxp" file as a free-standing SWF file set to display in full-screen mode: `http://localhost:8080/swf/sample.qxp?layout=Presentation&fullscreen=true`		

Example, object model	Request object name: `SWFRenderRequest`
	`//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 SWF renderer request`
	`//and embed it in the request context.`
	`SWFRenderRequest swfreq = new SWFRenderRequest();`
	`swfreq.setVersion(request.getParameter("Version"));`		

```
swfreq.setLayout(request.getParameter("Layout"));
swfreq.setPage(request.getParameter("Page"));
swfreq.setFullscreen(request.getParameter("Fullscreen"));
swfreq.setEmbedallfonts(request.getParameter(
  "Embedallfonts"));
swfreq.setCompressswf(request.getParameter("Compressswf"));
swfreq.setCompressaudio(request.getParameter(
  "Compressaudio"));
swfreq.setJpegquality(request.getParameter("Jpegquality"));
swfreq.setDownload(request.getParameter("Download"));
requestCtx.setRequest(swfreq);

//STEP3: Create the WIG service and call the processRequest(
//) API
QManagerSDKSvcServiceLocator serviceLocator =
    new QManagerSDKSvcServiceLocator();
QManagerSDKSvc service = serviceLocator.getqxpsmsdk();
sdk.QContentData data = service.processRequest(requestCtx);
```

Notes	@@@

Understanding render modifiers

Render modifiers let you control which parts of a project are rendered and set the scale of the returned renderings. The topics covered here include the following:

Property	Description
Box	The `box` render modifier lets you render a single box.
Boxes	The `boxes` render modifier lets you render multiple boxes.
Layer	The `layer` render modifier lets you show and hide layers prior to rendering. This render modifier also lets you add and remove layers from a project on the server.
Layout	The `layout` render modifier lets you render a particular layout.
Movepages	The `movepages` render modifier lets you move pages prior to rendering.
Page	The `page` render modifier lets you render a single page.
Pages	The `pages` render modifier lets you render multiple pages.
Scale	The `scale` render modifier lets you specify the scale at which content is rendered.
Spread	The `spread` render modifier lets you render a single spread.
Spreads	The `spreads` render modifier lets you render multiple spreads.

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

➡ In the QuarkXPress Server Manager API, render modifiers are properties of render request classes.

➡ Render modifier names are not case-sensitive.

Box

The box render modifier lets you render a single box.

Parameters	box	String	Lets you specify which box to render.
	overlap	String	Lets you specify whether to show the area overlapped by the specified box.
Compatible with	jpeg, png, raw		
Alerts	There is no box with the specified identifier.	HTTP Error #500 This alert displays if you request a box that does not exist.	
	Cannot render box. The box must be within the page boundaries.	HTTP Error #500 This alert displays if you request a box that is outside the page boundary.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you try to use the box parameter with the eps, pdf, or qxpdoc render types.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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 in a particular layout, use a URL like the following: http://localhost:8080/png/sample.qxp? layout=2&page=3&box=textbox When you render using the box parameter, the box ID has a higher priority than the box name.		

Boxes

The boxes render modifier lets you render multiple boxes.

Parameters	boxes	String	Lets you specify which boxes to render.
	overlap	String	Lets you specify whether to show the area overlapped by the specified boxes.
Compatible with	jpeg, png, raw		

Alerts	There is no box with the specified identifier.	HTTP Error #500 This alert displays if you request a box that does not exist.
	Cannot render box. The box must be within the page boundaries.	HTTP Error #500 This alert displays if you request a box that is outside the page boundary. .
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you try to use the `boxes` parameter with the `eps`, `pdf`, or `qxpdoc` render types.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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 boxes in a particular layout, use a URL like the following: `http://localhost:8080/png/sample.qxp?layout=2&page=3&box=textbox` When you render using the `box` parameter, the box ID has a higher priority than the box name.	

Compositionzone

The `compositionzone` parameter lets you return an XML representation of one or more Composition Zones items.

Parameters	`compositionzone`	String	Lets you specify which Composition Zones item to return. For example: `http://localhost:8080/xml/sample.qxp?compositionzone=czbox`
	`compositionzones`	String	Lets you specify which Composition Zones items to return. For example: `http://localhost:8080/xml/sample.qxp?compoxitionzones=czbox1, czbox2`
Compatible with	`xml`		
Alerts	Invalid box given in Box Param.	Error #10401 This alert displays if you request a box that is not a Composition Zones item.	

Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 12/1/2005 10:41:14 — xml/sample.qxp — Type: text/xml — Size: 63940 — 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: 12/1/2005 10:39:45 — Error — Error Code: 10401 — Invalid box given in Box Param.

Layer

The `layer` render modifier lets you show and hide layers prior to rendering. This render modifier also lets you add and remove layers from a project on the server.

Parameters	`layer`	String	Lets you specify which layer to render. You can specify multiple layer names in one request.
	`addlayer`	String	Lets you add a new layer. You can add one layer per request.
	`deletelayer`	String	Lets you delete a layer and the items on that layer. You can delete one layer per request.
	`alllayers`	Boolean (1 | 0 | true | false | yes | no)	Lets you render every layer in the project, including hidden and suppressed layers.
	`layerattribute`	String	Lets you modify the attributes of a layer. You can modify one layer per request.
	`name`	String	Lets you specify a new name for a layer. You must use this parameter in conjunction with the `layerattribute` parameter.
	`visible`	Boolean (1 | 0 | true | false | yes | no)	Lets you make a layer visible or invisible. You can use this parameter in conjunction with the `addlayer` and `layerattribute` parameters. This parameter overrides QuarkXPress layer visibility preferences.
	`suppressoutput`	Boolean (1 | 0 | true | false | yes | no)	Lets you suppress or allow the output of a layer. You can use this parameter in conjunction with the `addlayer` and `layerattribute` parameters. This parameter overrides QuarkXPress suppress output preferences.
	`locked`	Boolean (1 | 0 | true | false | yes | no)	Lets you lock or unlock a layer. You can use this parameter in conjunction with the `addlayer` and `layerattribute` parameters. This parameter overrides QuarkXPress layer locking preferences.
	`keeprunaround`	Boolean (1 | 0 | true | false | yes | no)	Lets you set or change a layer's Keep Runaround setting. You can use this parameter in conjunction with the `addlayer` and `layerattribute` parameters. This parameter overrides QuarkXPress Keep Runaround preferences.
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `ppml`, `qcddoc`, `qxpdoc`, `raw`, `pdf`, `screenpdf`, `swf`, `xml`		

Alerts	This layer does not exist. Please verify the layer name.	HTTP Error #500 This alert displays if you specify an invalid layer name with the `layer`, `layerattribute`, or `deletelayer` parameter.
	Specify a layer name.	HTTP Error #500 This alert displays if you do not specify a layer name with the `layer`, `layerattribute`, `addlayer`, or `deletelayer` parameter.
	A layer with the same name already exists.	HTTP Error #500 This alert displays if you try to add a layer that already exists or change the name of a layer to a name is already used in the project.
	Cannot change the name of the default layer.	HTTP Error #500 This alert displays if you try to change the name of the default layer.
	Cannot delete the default layer.	HTTP Error #500 This alert displays if you try to delete the default layer.
	Invalid parameter value.	HTTP Error #500 This alert displays if you do not specify additional attributes or specify attributes with invalid values in an `addlayer` or `layerattribute` request.
	This layer has been locked and cannot be modified.	HTTP Error #500 This alert displays if you try to add or modify an item on a locked layer.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	To render a single layer, use a URL like the following:
	`http://localhost:8080/doc.qxp?layer=layer1`
	To add a layer, use a URL like the following:
	`http://localhost:8080/qxpdoc/doc.qxp?addlayer=NewLayer&visible=yes&suppressoutput=yes&locked=no`
	To delete a layer, use a URL like the following:
	`http://localhost:8080/qxpdoc/doc.qxp?deletelayer=Layer1`
	To render all layers in a project, use a URL like the following:
	`http://localhost:8080/qxpdoc/doc.qxp?alllayers=true`
	To set layer attributes, use a URL like the following:
	`http://localhost:8080/qxpdoc/doc.qxp?layerattribute=Layer1&name=Layer2&visible=true&keeprunaround=true`	

Example, object model	To add a new layer to a project, use code like the following:
	`Layer layer = new Layer();`
	`layer.name = "New Layer";`	

	`layer.operation = "CREATE";`
	`RGBColor rgbcolor = new RGBColor();`
	`layer.RGBColor = rgbcolor;`
	`layout.layer = new Layer[]{layer};`
	To edit the properties of an existing layer, use the following object hierarchy:
	`ModifierRequest < Project < Layout < Layer`
	To delete a layer, set its `operation` attribute to "DELETE".

Notes	You cannot add, modify, or delete multiple layers in a single request. You cannot print layers whose `visible` and `suppressoutput` properties are set to `false`. You can render a hidden or suppressed layer by referencing it with the `layer` parameter. Suppressed layers are rendered for the `jpeg`, `png`, and `qxpdoc` render types, but not for the `pdf`, `postscript`, and `eps` render types. You can use the `deconstruct` and `getdocinfo` request handlers to view information about the layers in a project. When you add a layer using `addlayer`, any unspecified attributes use the settings in the QuarkXPress Server layer preferences (**QuarkXPress Server > Preferences > Default Print Layout > Layers**). If the `visible` property is set to `false`, the `suppressoutput` property is automatically set to `true`.

Layout

The `layout` render modifier lets you render a specific layout.

Parameters	`layout`	String	Lets you specify which layout to render. The first layout is layout 1.
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `ppml`, `raw`, `pdf`, `screenpdf`, `swf`		
Alerts	The requested layout does not exist.	HTTP Error #500 This alert displays if you supply an invalid layout value.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	To render a layout by its layer ID, use a URL like the following:
	`http://localhost:8080/png/sample.qxp?layout=2`
	To render a layout by its name, use a URL like the following:
	`http://localhost:8080/png/sample.qxp?layout=Layout 2`		

Movepages

The `movepages` render modifier lets you move pages prior to rendering.

<table>
<tr><td rowspan="2">Parameters</td><td>movepages</td><td>String</td><td>Lets you specify which pages to move. You can use a single page number (for example, 2) or a range of pages with the starting and ending page numbers separated by a hyphen (for example, 2–5).</td></tr>
<tr><td>afterpage</td><td>String</td><td>Lets you specify the page after which the page or pages should be moved. To move pages to the beginning of a layout, use afterpage=start. To move pages to the end of a layout, use afterpage=end.</td></tr>
<tr><td>Compatible with</td><td colspan="3">eps, jpeg, png, postscript, ppml, qcddoc, qxpdoc, raw, pdf, screenpdf, swf, xml</td></tr>
<tr><td rowspan="5">Alerts</td><td colspan="2">This page does not exist.</td><td>QuarkXPress Server Error #61</td></tr>
<tr><td colspan="2">Invalid page range.</td><td>QuarkXPress Server Error #62</td></tr>
<tr><td colspan="2">The specified page range cannot be moved there.</td><td>QuarkXPress Server Error #51</td></tr>
<tr><td colspan="2">This page range is invalid.</td><td>QuarkXPress Server Error #146</td></tr>
<tr><td colspan="2">Invalid parameter value.</td><td>QuarkXPress Server Error #10108</td></tr>
<tr><td>Logs</td><td colspan="3">If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example:
3/24/2009 10:42:01 — pdf/5pagedoc.qxp?movepages=AB-AD&afterpage=end — Type: application/pdf — Size: 12074 — 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:
3/24/2009 10:45:45 — Error — Error Code: 61 — This page does not exist. RequestURL: pdf/5pagedoc.qxp?movepages=AB-AD&afterpage=AF.</td></tr>
<tr><td>Example GET URL</td><td colspan="3">To move pages 2–3 to after page 5, use a URL like the following:
http://localhost:8080/abc.qxp?movepages=2-3&afterpage=5
To move page 7 to the beginning of a layout, use a URL like the following:
http://localhost:8080/abc.qxp?movepages=7&afterpage=
start</td></tr>
<tr><td>Example, object model</td><td colspan="3">To move pages before rendering a layout, use code like the following:
//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.setMovePages("2-4");
pdfreq.setAfterPage("7");
requestCtx.setRequest(pdfreq);</td></tr>
</table>

	`//STEP3: Create the WIG service and call the`
	`//processRequest() API`
	`QManagerSDKSvcServiceLocator serviceLocator =`
	`  new QManagerSDKSvcServiceLocator();`
	`QManagerSDKSvc service = serviceLocator.getqxpsmsdk();`
	`sdk.QContentData data = service.processRequest(`
	`  requestCtx);`

Notes	The `movepages` operation executes only after all other modifications are complete. For example, if you use `movepages` in a modify request, the pages are moved only after the modify request is complete.

Page

The `page` render modifier lets you render a single page.

Parameters	`page`	Integer	Lets you specify which page to render.
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `qcddoc`, `raw`, `pdf`, `screenpdf`, `swf`		
Alerts	The requested page does not exist.	HTTP Error #500 This alert displays if you attempt to render a page that does not exist.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you use a `page` parameter with the `qxpdoc` render type.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	To add a new page to an existing spread in a project, use code like the following:
	`Spread spread = new Spread();`
	`Page page = new Page();`
	`page.UID = "5";`
	`page.operation = "CREATE";`
	`spread.page = new Page[]{page};`
	To edit the properties of an existing page, use the following object hierarchy:
	`ModifierRequest < Project < Layout < Spread < Page`
	To delete a page, set its `operation` attribute to "DELETE".		

Notes	To render a page in a particular layout, use a URL like the following: `http://localhost:8080/png/sample.qxp?layout=2&page=3`		

Pages

The `pages` render modifier lets you render multiple pages. The `pdf` and `postscript` namespaces support this parameter.

Parameters	`pages`	String (page range)	Lets you specify which pages to render.
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `raw`, `pdf`, `screenpdf`, `swf`		
Alerts	This page range is invalid.	HTTP Error #500 QuarkXPress Server Error #147 This alert displays if you try to render a page range that exceeds the number of pages in the project.	
	The renderer for this image type has no way of rendering the desired objects.	HTTP Error #406 This alert displays if you use the `pages` parameter with the `jpeg`, `eps`, `png`, or `qxpdoc` render type.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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 in a particular layout, use a URL like the following: `http://localhost:8080/pdf/sample.qxp?layout=2&pages=2,3`		

Scale

The `scale` render modifier lets you specify the scale at which content is rendered.

Parameters	`scale`	Float	Lets you specify a scaling percentage. The valid values are from .1 (10%) to 8 (800%) on Mac OS or 6.92 (692%) on Windows.
Compatible with	`eps`, `jpeg`, `png`, `raw`		
Alerts	Invalid scale parameter.	HTTP Error #500 This alert displays if an invalid scale value is provided. *What to do*: Enter a valid scale value.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`

Spread

The `spread` render modifier lets you render a single spread.

Parameters	`spread`	Integer	Lets you specify which spread to render. Spread numbers start with 1. The first spread is spread 1. In a facing-page document, spread 1 consists of the first page.
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `raw`, `pdf`, `screenpdf`, `swf`		
Alerts	The requested spread does not exist.	HTTP Error #500 This alert displays if you specify an invalid spread.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	To add a spread to a project, use code like the following:
	`Spread spread = new Spread();`
	`spread.UID = "5";`
	`spread.operation = "CREATE";`
	`layout.spread = new Spread[]{spread};`
	Spread is located at the following place in the object hierarchy:
	`ModifierRequest < Project < Layout < Spread`
	To delete a spread, set its `operation` attribute to "DELETE".		

Spreads

The `spreads` render modifier lets you render layouts in spreads mode, so that pages in spreads are rendered side-by-side rather than as individual pages.

Parameters	`spreads`	Boolean (1 | 0 | true | false | yes | no)	Lets you specify whether to render spreads (true) or individual pages (false).
Compatible with	`eps`, `jpeg`, `png`, `postscript`, `raw`, `pdf`, `screenpdf`, `swf`		
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`

Using content modifiers

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

Inserting text

This topic explains how to import text into a box. Any existing text in the box is replaced.

Parameters	*[box name]*	String	The name of the target box.
			Specify the name and location of the imported file with the `file:` prefix. The imported file must be present in the document pool.
			To import a file that is in a subfolder of the document pool on Mac OS, use a path like the following:
			`file:subfolder:MyFile.ext`
			To import a file that is in a subfolder of the document pool on Windows, use a path like the following:
			`file:subfolder\MyFile.ext`

Response	A preview of the project with the imported text.		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if the imported file is not present in the document pool.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 11:27:42 — jpeg/sample.qxp — Type: image/jpeg — Size: 31715 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	Box names are case-sensitive. 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. You can use "&" to change the contents of multiple boxes in one request. For example: `http://localhost:8080/sample.qxp?Headline=headline.txt&Story=file:Story.doc` You can import an XTags file generated by QuarkXPress.

Applying a font at import

This topic explains how to apply a font to a new text flow. When you use this method, QuarkXPress Server ignores the original font of the target text box and inserts the new text with the font specified by the parameter.

Parameters	`fontname`	String	The name of the font to be applied.
Response	A preview of the project with the font applied to the imported text.		
Alerts	The specified font is not available.	This alert displays if you specify a font that is unavailable.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	To apply Comic Sans MS to text in the box named "HeadBox," use a URL like the following: `http://localhost:8080/png/sample.qxp?HeadBox=Headline&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

This topic explains how to import a picture into an empty box or replace an existing picture with a new one.

Parameters	*[box name[*	String	The name of the target box.
			Specify the name and location of the imported file with the `file:` prefix. The imported file must be present in the document pool.
			To import a file that is in a subfolder of the document pool on Mac OS, use a path like the following:
			`file:subfolder:MyFile.ext`
			To import a file that is in a subfolder of the document pool on Windows, use a path like the following:
			`file:subfolder\MyFile.ext`

Response	A preview of the project with the imported picture.		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if the imported file is not present in the document pool.	
	The specified file failed to load in the picture box.	HTTP Error #500 This alert displays if you attempt to import an invalid picture file.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	Box names are case-sensitive.
	You can use "&" to change the contents of multiple boxes in one request. For example:
	`http://localhost:8080/sample.qxp?`
	`Logo=file:logo.jpeg&TopPicture=file:TopPicture.eps`

Saving a projects with a new name

The `saveas` content modifier lets you save modified QuarkXPress projects in any supported format to the document pool or to any network location accessible to QuarkXPress Server.

If you send a `saveas` request to QuarkXPress Server Manager using HTTP or the Web services interface while the common doc pool switch is set to off in the QuarkXPress Server Manager client, the file is saved to all registered QuarkXPress Server instances. If the common doc pool is enabled, the file can be saved to any one registered QuarkXPress server instance.

Parameters	`newname`	String	Lets you specify a name for the saved-as project.
	`path`	String	Lets you specify a location for the saved-as project (other than the document pool).
	`savetopool`	true | false	Lets you specify whether the project should be saved to the document pool. The default value for this paramter is true. However, if you specify a `path` value, the default value changes to false, which means if you want the project saved to the document pool, you must explicitly set `savetopool` to true.
	`replace`	true | false	Lets you specify whether the saved project should replace a project with the same name. The default value is true.
Response	The message "Document successfully saved."		
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you supply an incorrect file name or the file is not in document pool.	
	Bad filename/ pathname.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you supply an incorrect file name or the file is not in document pool.	
	The file path is invalid.	HTTP Error #500 This alert displays if you supply an invalid `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 displays if you try to save a project to a folder with read-only access.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 11/16/2005 15:41:42 — saveas/5mb.qxp — Type: — Size: 28 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 11/16/2005 15:42:12 — Error — Error Code: 10371 — The file path is invalid.	
Example, GET URL	To save a PDF file named "Customer1.pdf" in the folder `HDD:temp` and also in the document pool, use a URL like the following. Note that this URL will cause the saved-as file to replace any existing file with the same name. `http://localhost:8080/saveas/pdf/sample.qxp?newname=Customer1&path=HDD:temp&savetopool=true`	

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);`	

Importing XML with placeholders

This topic explains how to import XML data into boxes using QuarkXPress placeholders.

To use this feature, you must have a QuarkXPress project that has been set up with placeholders that correspond to the element types in a source XML file. For more information, see *A Guide to XML Import*.

Parameters	`thexmldoc`	XML	Lets you specify the XML file containing the data to import. The path can be absolute or relative to the location of the XML file. You can also supply XML as a string.
	`layout`	String	Lets you specify which layout to render. The first layout is layout 1. You can also specify a layout by name.

	`paginate`	XML	Lets you specify the XML file containing the data to import. The `paginate` parameter reates enough pages in the target layout to accommodate the records in the XML. This parameter works only with the `pdf`, `postscript`, `qxp`, and `ppml` render types. If you use it with any other render type, the server returns only the first page of the paginated layout. If you do not supply an XML string or file (for example: `http://localhost:8080/pdf/Sample.qxp?paginate`), QuarkXPress Server attempts to use the XML file that was associated with the layout in QuarkXPress.
Response	The layout with the imported XML.		
Alerts	Invalid XML String	HTTP Error #500 This alert displays if you supply an invalid XML string in 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 displays, an error message is written to the QuarkXPress Server error log file. For example: 8/9/2005 12:38:42 — Error — Error Code: 10396 — Invalid XML String.		

Example, GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`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>`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`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>`
	Alternatively, you can specify a path to a file containing the XML:
	`http://localhost:8080/Sample.qxp?paginate=file:MacHD:Sample.xml`		

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);
```

Using XML modify

The `modify` parameter lets you modify a QuarkXPress project using XML.

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 "*Constructing a project*".

The `xml` namespace takes two arguments: the name of the project to be modified, and a `modify` parameter with the string or the path of the XML file 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
```

DTD	Modifier DTD		
Parameters	`modify`	String	Lets you specify an XML file or string that describes how to create a project. The path can be absolute or a relative path in the document pool. Use the `file:` indicator to specify the path.
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.		
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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.

Modifying box properties and content

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

- "*BOX (Modifier DTD)*"
- "*ID (Modifier DTD)*"
- "*TEXT (Modifier DTD)*"
- "*PICTURE (Modifier DTD)*"
- "*GEOMETRY (Modifier DTD)*"
- "*CONTENT (Modifier DTD)*"
- "*SHADOW (Modifier DTD)*"
- "*FRAME (Modifier DTD)*"
- "*PLACEHOLDER (Modifier DTD)*"
- "*METADATA (Modifier DTD)*"

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	A 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 displays if you specify an invalid XML file or request a document that is not in the document pool.
	Bad filename/pathname.	HTTP Error #404 QuarkXPress Server Error #–37 This alert displays if you specify an invalid file name or path.
	The XML document is not valid or well formed.	HTTP Error #500 This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert displays if you supply an invalid value in the XML.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 11:27:42 — jpeg/sample.qxp — Type: image/jpeg — Size: 31715 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example:	

Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:C:\updateBox.xml`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:MacHD:xml:updateBox.xml`
	You can also supply a string that consists of valid XML commands. 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
```

For `ModifierFileRequest`, the member contents are 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.project = 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

To edit the geometrical properties of an existing box in a QuarkXPress project, use the following object hierarchy:

```
ModifierRequest < Project < Layout < Spread < Box < Geometry
```

The `Geometry` object has the following 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
suppressOutput
```

The `Runaround` object has the following properties:

```
bottom
edited
invert
left
noise
outset
outsideOnly
pathName
restrictToBox
right
smoothness
threshold
top
type
```

Creating boxes

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

- "*BOX (Modifier DTD)*"
- "*ID (Modifier DTD)*"
- "*TEXT (Modifier DTD)*"
- "*PICTURE (Modifier DTD)*"
- "*GEOMETRY (Modifier DTD)*"
- "*CONTENT (Modifier DTD)*"
- "*SHADOW (Modifier DTD)*"
- "*FRAME (Modifier DTD)*"

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>
```

<table>
<tr><td>Response</td><td colspan="2">A preview of the QuarkXPress project with new box created in specified position.</td></tr>
<tr><td rowspan="4">Alerts</td><td>File not found.</td><td>HTTP Error #404
QuarkXPress Server Error #–43
This alert displays if you specify an invalid XML file or request a document that is not in the document pool.</td></tr>
<tr><td>Bad filename/ pathname.</td><td>HTTP Error #404
QuarkXPress Server Error #–37
This alert displays if you specify an invalid file name or path.</td></tr>
<tr><td>The XML document is not valid or well formed.</td><td>HTTP Error #500
This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.</td></tr>
<tr><td>The XML document contains an invalid tag value.</td><td>HTTP Error #500
This alert displays if you supply an invalid value in the XML.</td></tr>
<tr><td>Logs</td><td colspan="2">If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example:
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 displays, an error message is written to the QuarkXPress Server error log file. For example:
4/12/2007 14:51:50 — Error — Error Code: 10207 — The XML document is not valid or well formed. Project: /table.qxp</td></tr>
<tr><td>Example, GET URL</td><td colspan="2">When QuarkXPress Server is running on Windows, use a URL like the following:
<code>http://localhost:8080/sample.qxp?modify=
file:C:\createBox.xml</code>
When QuarkXPress Server is running on Mac OS, use a URL like the following:
<code>http://localhost:8080/sample.qxp?modify=
file:MacHD:xml:createBox.xml</code>
You can also supply a string that consists of valid XML commands. For example:
<code>http://localhost:8080/sample.qxp?modify=<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></code></td></tr>
<tr><td>Example, object model</td><td colspan="2">To create a new box, use code like the following:
<code>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";</code></td></tr>
</table>

```
geometry.layer = "Default";

box.geometry = geometry;
box.boxType = "CT_TEXT";

box.operation = "CREATE";
spread.box = new Box[]{box};
```

Use the following object hierarchy:

```
ModifierRequest < Project < Layout < Spread < Box < Geometry
```

Deleting boxes

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

- "*BOX (Modifier DTD)*"
- "*ID (Modifier DTD)*"

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	A preview of the QuarkXPress project with the box deleted.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you specify an invalid XML file or request a document that is not in the document pool.
	Bad filename/ pathname.	HTTP Error #404 QuarkXPress Server Error #–37 This alert displays if you specify an invalid file name or path.
	The XML document is not valid or well formed.	HTTP Error #500 This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert displays if you supply an invalid value in the XML.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 11:27:42 — jpeg/sample.qxp — Type: image/jpeg — Size: 31715 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example:	

Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:C:\deleteBox.xml`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:MacHD:xml:deleteBox.xml`
	You can also supply a string that consists of valid XML commands. 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.

Grouping and ungrouping items

To group boxes using XML modify, use XML like the following:

```
<BOX BOXTYPE="CT_TEXT" COLOR="White">
  <ID NAME="MainStoryText" UID="217"/>
</BOX>

<BOX BOXTYPE="CT_PICT">
  <ID NAME="MainStoryPhoto" UID="218"/>
</BOX>

<GROUP>
  <ID NAME="MainStoryGroup" UID="300" OPERATION="CREATE"/>
  <BOXREF NAME="MainStoryText" UID="217"/>
  <BOXREF NAME="MainStoryPhoto" UID="218"/>
</GROUP>
```

To add a box to an existing group, use XML like the following:

```
<GROUP>
  <ID NAME="MainStoryGroup" UID="300"/>
  <BOXREF NAME="MainStoryText" UID="217"/>
  <BOXREF NAME="MainStoryPhoto" UID="218" OPERATION="CREATE"/>
</GROUP>
```

To remove a box from an existing group, use XML like the following:

```
<GROUP>
  <ID NAME="MainStoryGroup" UID="300"/>
  <BOXREF NAME="MainStoryHead" UID="216"/>
  <BOXREF NAME="MainStoryText" UID="217"/>
  <BOXREF NAME="MainStoryPhoto" UID="218" OPERATION="DELETE"/>
</GROUP>
```

To ungroup an existing group, use XML like the following:

```
<GROUP>
  <ID NAME="MainStoryGroup" UID="300" OPERATION="DELETE"/>
</GROUP>
```

To proportionally scale all of the items in a group, add a `<GEOMETRY>` element that indicates the new size of the group, like so:

```
<GROUP>
  <ID NAME="MainStoryGroup" UID="300"/>
  <GEOMETRY>
    <POSITION>
      <TOP>10.0</TOP>
      <LEFT>10.0</LEFT>
      <BOTTOM>50.0</BOTTOM>
```

```
      <RIGHT>70.0</RIGHT>
    </POSITION>
  </GEOMETRY>
</GROUP>
```

- The order of the <BOXREF> elements in a <GROUP> indicates the order in which the boxes were selected prior to grouping. The z-order of boxes in the layout is determined by the order of the <BOX> elements in the XML, from rearmost to frontmost.

- XML representations of groups created by versions of QuarkXPress Server prior to 8.5 are ignored during construct and modify calls, as they were in earlier versions of QuarkXPress Server.

Creating tables

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

- "*SPREAD (Modifier DTD)*"
- "*TABLE (Modifier DTD)*"
- "*COLSPEC (Modifier DTD)*"
- "*COLUMN (Modifier DTD)*"
- "*ROW (Modifier DTD)*"
- "*CELL (Modifier DTD)*"

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	A preview of the QuarkXPress project with new table created in the specified position.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 4/10/2007 17:54:37 — tab.qxp — Type: image/jpeg — Size: 9049 — Client: 127.0.0.1
Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following: `http://localhost:8080/sample.qxp?modify=file:C:\createTable.xml`

	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=`
	`file:MacHD:xml:createTable.xml`
	You can also supply a string that consists of valid XML commands. 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	To add a new table to an existing spread, use code like the following:
	`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};`
	Use the following object hierarchy:
	`ModifierRequest < Project < Layout < Spread < Table`
	To delete a table, provide the table's name or ID and set the `operation` attribute to "DELETE".

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 (Modifier DTD)*"
- "*ID (Modifier DTD)*"
- "*TEXT (Modifier DTD)*"
- "*STORY (Modifier DTD)*"
- "*PARAGRAPH (Modifier DTD)*"
- "*FORMAT (Modifier DTD)*"
- "*DROPCAP (Modifier DTD)*"

- "*TABSPEC (Modifier DTD)*"
- "*TAB (Modifier DTD)*"
- "*RULE (Modifier DTD)*"
- "*RICHTEXT (Modifier DTD)*"

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	A 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 displays if you specify an invalid XML file or request a document that is not in the document pool.
	Bad filename/ pathname.	HTTP Error #404 QuarkXPress Server Error #–37 This alert displays if you specify an invalid file name or path.
	The XML document is not valid or well formed.	HTTP Error #500 This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.

	There is no box with the specified identifier.	HTTP Error #500 This alert displays if the box specified by the child text node of an `<ID>` element does not exist.
	The text size value is outside the valid range.	HTTP Error #500 This alert displays if the value specified in a `<SIZE>` element is invalid. *What to do*: Specify a value between 2 and 720 points.
	The specified color is not available to the document	HTTP Error #500 This alert displays if the value specified in a `<COLOR>` element is invalid.
	The specified font is not available	HTTP Error #500 This alert displays if the value specified in a `<FONT>` element is invalid or the specified font is not present on the server.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert displays if you supply an invalid value in the XML.
	The specified box cannot be modified.	HTTP Error #500 This alert displays if you try to modify text properties on a box that is not a text box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 11:27:42 — jpeg/sample.qxp — Type: image/jpeg — Size: 31715 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 8/5/2005 13:32:10 — Error — Error Code: 10006 — There is no box with the specified identifier.	

Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=`
	`file:C:\modifier.xml`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=`
	`file:MacHD:xml:modifier.xml`
	You can also supply a string that consists of valid XML commands. 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`
	For `ModifierFileRequest`, the member contents are 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 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	To edit the properties of an existing text box in a QuarkXPress project, use the following object hierarchy:
	`ModifierRequest < Project < Layout < Spread <`
	`Box < Text < Story < Paragraph < RichText`
	For a list of the `RichText` object's properties, see the JavaDoc installed with QuarkXPress Manager.
	The `Story` object also contains some text-related properties: `fitTextToBox`, `includeStylesheets`, `convertQuotes`, and `clearOldText`.

Notes	The `<FITTEXTTOBOX>` attribute depends on two preferences: **Allow Text to Grow** and **Font Size**. To set these preferences in QuarkXPress Server, choose **QuarkXPress > Server > Preferences** and then click **Modifier** in the list on the left. If you prefer to set text fitting preferences on a story-by-story basis, use the `<FITTEXT>` element type rather than the `<FITTEXTTOBOX>` attribute (for more information, see "*FITTEXT (Modifier DTD)*").

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 (Modifier DTD)*"
- "*ID (Modifier DTD)*"
- "*PICTURE (Modifier DTD)*"

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	A preview of the QuarkXPress project with image modifier tags applied to the picture boxes.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you specify an invalid XML file or request a document that is not in the document pool.
	Bad filename/ pathname.	HTTP Error #404 QuarkXPress Server Error #–37 This alert displays if you specify an invalid file name or path.
	The XML document is not valid or well formed.	HTTP Error #500 This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.

	There is no box with the specified identifier.	HTTP Error #500 This alert displays if the box specified by the child text node of the `<ID>` element does not exist.
	The value of Scale Across should be between 10% and 1000%.	HTTP Error #500 This alert displays if the value of the child text node of a `<SCALEACROSS>` element is invalid.
	The Value of Scale Down should be between 10% and 1000%.	HTTP Error #500 This alert displays if the value of the child text node of a `<SCALEDOWN>` element is invalid.
	The value of Offset Across is in invalid range.	HTTP Error #500 This alert displays if the value of the child text node of the `<OFFSETACROSS>` element is invalid.
	The value of Offset Down is in invalid range	HTTP Error #500 This alert displays if the value of the child text node of the `<OFFSETDOWN>` element is invalid.
	The value of Picture Angle must be between –360 and 360 degrees.	HTTP Error #500 This alert displays if the value of the child text node of the `<ANGLE>` element is invalid.
	The value of Picture Skew must be between –75 and 75 degrees.	HTTP Error #500 This alert displays if the value of the child text node of the `<SKEW>` element is invalid.
	The XML document contains an invalid tag value.	HTTP Error #500 This alert displays if you supply an invalid value in the XML.
	The specified box cannot be modified.	HTTP Error #500 This alert displays if you try to modify picture properties on a box that is not a picture box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 11:27:42 — jpeg/sample.qxp — Type: image/jpeg — Size: 31715 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 8/10/2005 10:39:07 — Error — Error Code: 10339 — The specified file failed to load in the picture box.	

Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:C:\imageProperties.xml`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/sample.qxp?modify=file:MacHD:xml:imageProperties.xml`	

	You can also supply a string that consists of valid XML commands. 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`
	For `ModifierFileRequest`, the member contents are used to set the file path or send the XML itself.
	`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	To edit the properties of an existing text box in a QuarkXPress project, use the following object hierarchy: `ModifierRequest < Project < Layout < Spread < Box < Picture` For a list of the `Picture` object's properties, see the JavaDoc installed with QuarkXPress Manager.
Notes	You cannot replace an image with the Modifier XTensions software.

	If you specify <FITPICTURETOBOX>, <FITBOXTOPICTURE>, and <FITPICTURETOBOXPRO> for a picture, only the first of these elements will be applied.

Importing data

Imports text or image data into a project. You can use import any text or picture file format supported by QuarkXPress, including XPress Tags files.

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

- "*BOX (Modifier DTD)* "
- "*ID (Modifier DTD)*"
- "*PICTURE (Modifier DTD)*" (this is not a required element when importing data)
- "*TEXT (Modifier DTD)*"
- "*STORY (Modifier DTD)*"
- "*CONTENT (Modifier DTD)*"

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>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	A preview of a QuarkXPress project with a value in the data import XML tags applied to the text boxes.	
Alerts	File not found.	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you specify an invalid XML file or request a document that is not in the document pool.
	The XML document is not valid or well formed.	HTTP Error #500 This alert displays if the XML you supply is not well-formed or does not adhere to the Modifier DTD.

	There is no box with the specified identifier.	HTTP Error #500 This alert displays if the box specified by the child text node of the `<ID>` element does not exist.
	The specified box is not a picture or text box.	HTTP Error #500 This alert displays if you request a box that is not a text box or a picture box.
	A locked layer cannot be manipulated.	HTTP Error #500 This alert displays if you request data from a box on a locked layer. *What to do*: Open the project in QuarkXPress, display the **Layers** palette, and unlock the box's layer.
	Unable to read picture (#106)	HTTP Error #500 QuarkXPress Server Error #–109 This alert displays if you try to import a text file into a picture box.
	Bad filename/ pathname	HTTP Error #404 QuarkXPress Server Error #–37 This alert displays if you try to import an invalid or nonexistent file into a box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/5/2005 18:11:54 — sample.qxp — Type: image/jpeg — Size: 65982 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 8/5/2005 18:01:59 — Error — Error Code: 10343 — A locked Layer cannot be manipulated.	

Example GET URL	When QuarkXPress Server is running on Windows, use a URL like the following:
	`http://localhost:8080/Sample.qxp?modify=file:c:\file.xml`
	When QuarkXPress Server is running on Mac OS, use a URL like the following:
	`http://localhost:8080/Sample.qxp?modify=file:HDD:file.xml`
	You can also supply a string that consists of valid XML commands. For example:
	`http://localhost:8080/sample.qxp?modify=<PROJECT><LAYOUT><ID UID="Layout1"/><SPREAD><ID UID="1"/><BOXBOXTYPE="CT_TEXT"><ID NAME="TREES"/><CONTENT>C:\docs\file1.jpg</CONTENT></BOX></SPREAD></LAYOUT></PROJECT>`
	When specifying a path, use URLs like the following:
	`http://localhost:8080/Sample.qxp?textboxname@dataimport=file:c:\file.txt`
	`http://localhost:8080/Sample.qxp?pictureboxname@dataimport=c:\file.jpg`
	You can import text directly into a box from the URL string. For example:
	`http://localhost:8080/Sample.qxp?textboxname@dataimport=Newdata`	

	When you import a file that uses style sheets, you can control how those style sheets are handled. For example:
	`http://localhost:8080/Documentname?`
	`textboxname@dataimport=file:c:\file.doc&`
	`textboxnameincludestylesheets@dataimport=yes`
	You can control how quotation marks are handled at import. For example:
	`http://localhost:8080/Documentname?`
	`textboxname@dataimport=file:c:\file.doc&`
	`textboxnameconvertquotes@dataimport=yes`

Example, object model	Request object names:
	`ModifierRequest`
	`ModifierStreamRequest`
	`Project`
	`RichText`
	`Text`
	`ID`
	`Box`
	`Layout`
	`ModifierFileRequest`
	For `ModifierFileRequest`, the member contents are 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	BoxParam XTensions software lets you import only files in the document pool. Modifier XTensions software, however, lets you import files located anywhere on the server computer or any accessible network location.

Exporting Job Jackets files during 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.

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

Using XML deconstruct and construct

The `xml` namespace deconstructs a project according to the Modifier DTD. The `construct` namespace lets you 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 `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. For more information, see "*Modifier DTD (annotated)*."

- 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.

- The `deconstruct` namespace/request no longer exists. If you try to use it in this version of QuarkXPress Server, an error is returned.

Deconstructing a project

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 (see "*Modifier DTD (annotated)*").

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 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.

- Projects can also refer to resources defined with the QuarkXPress Server **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.

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 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 box with the given ID or name.
	`boxes`	Returns XML only for the boxes with the IDs or names supplied as a comma -separated list.
	`XSL`	Specifies the path of an XSL file for transforming the returned XML. Use the `file:` indicator to specify the path.

	`layout`	Specifies the name or number of the layout containing the box to render. The first layout is layout 1. Note that this parameter works only with the `box` parameter.
	`relativegeometry`	Tells the `xml` namespace to describe `<GEOMETRY>` elements using `<RELPOSITION>` rather than `<POSITION>`. This allows an item's position to be defined either in relation to the page or in relation to the entire spread.
	`relativetopage`	Use only with the `relativegeometry` parameter. Tells the `xml` namespace to describe `<GEOMETRY>` elements using `<RELPOSITION>` elements in which `ORIGIN@RELATIVETO="page"` (as opposed to `"spread"`).
	`copyfitinfo`	QuarkXPress Server returns copyfitting information for QuarkCopyDesk articles by default. To retrieve copyfitting information when deconstructing a QuarkXPress project, include `copyfitinfo=true` in the `xml` request. For example: `http://localhost:8080/xml/sample.qxp?copyfitinfo=true`
	Refer to the Modifier DTD	
Response	Sample response:	

```
<?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 MEDIATYPE="PRINT">
    <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"
         FORMATTEDNAME="1">
        <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</SUPPRESSOUTPUT>
          <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&amp;quot;"
```

	<pre>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 Server Documents: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></pre>

Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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`
	You can also deconstruct QuarkCopyDesk articles. To deconstruct a QuarkCopyDesk article, use the following:
	`http://localhost:8080/xml/copydesk/abc.qcd`

Example, Object Model	Request object name: `XMLRequest`
	<pre>XMLRequest xmlRequest = new XMLRequest();
	QRequestContext context = new QRequestContext();
	context.setDocumentName("SAMPLE_DOCUMENT.qxp");
	context.setResponseAsURL(false);
	context.setRequest(xmlRequest);
	QContentData response =
	QRequestProcessor.getInstance().processRequest(context);
	System.out.println(response.getTextData());</pre>

Constructing a project

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. For example:

```
http://QXPServer8:8080/construct/project1.qxp?
modify=file:path to XML file on server
```

or:

```
http://QXPServer8:8080/construct/project1.qxp?modify=XML string
```

➡ 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.

➡ If you are using QuarkXPress Server Manager, you can send a similar command with a QuarkXPress Server Manager URL or through Web services.

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 more information, see "*Modifier DTD (annotated)*.")

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

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

Construct

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

Namespace	`construct`		
DTD	Modifier DTD		
Parameters	`modify`	String	The string or the path of the XML file that describes how to create the project. Use the `file:` indicator to specify the path.
	`qxpdocver`	7 | 8 | japanese6 | korean6	Indicates the QuarkXPress version format to use. For example: `http://QXPServer8:8080/construct/qxpdoc/project1.qxp?qxpdocver=7`

<table>
<tr><td>Example GET URL</td><td colspan="2">http://QXPServer8:8080/construct/
project1.qxp?modify=file:sample.xml</td></tr>
<tr><td>Example XML</td><td colspan="2"><?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></td></tr>
<tr><td>Response</td><td colspan="2">A new QuarkXPress project.</td></tr>
<tr><td rowspan="4">Alerts</td><td>File not found.</td><td>HTTP Error #404
QuarkXPress Server Error #–43
This alert displays if you specify an invalid XML file or request a document that is not in the document pool. For example, this error can occur if an image or text file file mentioned in a <CONTENT> element is invalid or missing.</td></tr>
<tr><td>Bad filename/pathname.</td><td>HTTP Error #404
QuarkXPress Server Error #–37
This alert displays if you specify an invalid file name or path.</td></tr>
<tr><td>The XML document is not valid or well formed.</td><td>HTTP Error #500
This alert displays if the XML you supply is not well-formed or do not adhere to the Modifier DTD.</td></tr>
<tr><td>The XML document contains an invalid tag value.</td><td>HTTP Error #500
This alert displays if you supply an invalid value in the XML.</td></tr>
<tr><td>Logs</td><td colspan="2">If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example:
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.</td></tr>
<tr><td>Example, object model</td><td colspan="2">Request Object Names:
XMLRequest
ConstructRequest
ConstructFileRequest
ConstructStreamRequest
To construct a new QuarkXPress project by editing an existing document, first deconstruct a QuarkXPress project using code like the following:
XMLRequest dcnstrq = new XMLRequest();
rc.request = dcnstrq;</td></tr>
</table>

	Next, alter the project by manipulating the XML. When you're done, pass the modified XML document to `ConstructStreamRequest` to create a new QuarkXPress project. For example:
	`ConstructStreamRequest cnstrq =`
	`  new ConstructStreamRequest();`
	`cnstrq.modify = Buffer; // Byte[] for the modified XML`
	`rc.request = cnstrq;`
	
	`QuarkXPressRenderRequest qxprq =`
	`  new QuarkXPressRenderRequest();`
	`cnstrq.request = qxprq;`
	Alternatively, you can deconstruct a QuarkXPress project using code like the following:
	`QManagerSDKSvcService svc =`
	`  new QManagerSDKSvcService()`
	`Project proj = svc.getDOM("document.qxp");`
	Next, alter the project by manipulating the XML. When you're done, pass the modified `Project` instance to `ConstructRequest` to create a new QuarkXPress project. For example:
	`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 with the string or the path of the XML file that describes how to create the project:
	`http://localhost:8080/qxpdoc/construct/project1.qxp?`
	`modify=file:path to XML file on server`
	`http://localhost:8080/qxpdoc/construct/project1.qxp?`
	`modify=<xml-string>`

Construct and modify

The `modify` parameter lets you modify existing projects. For example:

```
http://QXPServer8:8080/project1.qxp?
modify=file:path to XML file on server
```

or:

```
http://QXPServer8:8080/project1.qxp?modify=XML string
```

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.

For more information, see "*Modifier DTD (annotated)*."

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.

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`).

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>
```

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.

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.

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>
```

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>
    <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>

</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.

➡ The z-order (stacking order) of boxes in the layout is determined by the order of the `<BOX>` elements in the XML, from rearmost to frontmost.

Fitting a box to text or a picture

The `<FIT>` element type lets you automatically adjust the size of a box to fit the text or picture in that box.

The default behaivior is to not fix a box to its content. To use this feature, you must supply `<MAX>` and `<MIN>` elements. Each `<MAX>` or `<MIN>` element lets you specify a maximum or minimum size for the box, a maximum or minimum location for the resized box, or a maximum or minimum scale percentage for the box. Note that you can use different types of `<MAX>` and `<MIN>` elements in a `<FIT>` element, but you can use only one `<MAX>` element and one `<MIN>` element per `<FIT>` element.

The `FIT@POINT` attribute lets you indicate the direction in which the box should grow or shrink. The available options are `TOPLEFT`, `BOTTOMLEFT`, `TOPRIGHT`, and `BOTTOMRIGHT`.

The `FIT@AVOIDBOXESBY` attribute lets you specify the distance between the `POINT` side or corner of a resized box and any other items around it. A box will expand only until it is this distance from an adjacent item.

The FIT@PROPORTIONAL attribute lets you specify whether the resized box should have the same aspect ratio as the original box.

For example:

```
<BOX>
    <ID UID="5"/>
    <GEOMETRY>
        <POSITION>
            <TOP>224.001</TOP>
            <LEFT>110.003</LEFT>
            <BOTTOM>381</BOTTOM>
            <RIGHT>253.253</RIGHT>
        </POSITION>
        <FIT POINT="BOTTOMLEFT" PROPORTIONAL="true">
            <MAX>
                <LOCATION X="320" Y="560"/>
            </MAX>
            <MIN>
                <SIZE HEIGHT="100" WIDTH="10"/>
            </MIN>
        </FIT>
    </GEOMETRY>
<BOX/>
```

➡ To use this feature, you must have FitBoxToContent XTensions software loaded.

➡ For pictures, <FIT> is equivalent to PICTURE@FIT="FITBOXTOPICTURE". <MAX> and <MIN> have no effect.

Working with groups

To add boxes to a group, create a <GROUP> element and then insert <BOXREF> elements that refer to the boxes you want in the group. For example, the group described below includes the two boxes described above it:

```
 <BOX BOXTYPE="CT_TEXT" COLOR="White">
    <ID NAME="MainStoryText" UID="217"/>
</BOX>

<BOX BOXTYPE="CT_PICT">
    <ID NAME="MainStoryPhoto" UID="218"/>
</BOX>

<GROUP>
    <ID NAME="MainStoryGroup" UID="300"/>
    <BOXREF NAME="Main StoryText" UID="217"/>
    <BOXREF NAME="Main StoryPhoto" UID="218"/>
</GROUP>
```

You can nest one group within another by adding a <BOXREF> that refers to the child group, like so:

```
<GROUP>
    <ID NAME="MainStoryGroup" UID="300"/>
    <BOXREF NAME="MainStoryText" UID="217"/>
    <BOXREF NAME="MainStoryPhoto" UID="218"/>
</GROUP>

<BOX BOXTYPE="CT_PICT">
     <ID NAME="Masthead" UID="001"/>
</BOX>

<GROUP>
    <ID NAME="MainStoryPage" UID="218"/>
    <BOXREF NAME="Masthead" UID="001"/>
    <BOXREF NAME="MainStoryGroup" UID="300"/>
</GROUP>
```

To anchor a group in a text box, use XML like the following. Note that you must set `BOX@ANCHOREDGROUPMEMBER="true"` for all boxes in the group, and set `GROUP@ANCHOREDIN` for the anchored group.

```
<BOX BOXTYPE="CT_TEXT" COLOR="White" ANCHOREDGROUPMEMBER="true" >
    <ID NAME="MainStoryText" UID="217"/>
</BOX>

<BOX BOXTYPE="CT_PICT" ANCHOREDGROUPMEMBER="true" >
    <ID NAME="MainStoryPhoto" UID="218"/>
</BOX>

<GROUP ANCHOREDIN="MainStoryText">
    <ID NAME="DropCapGroup" UID="300"/>
    <BOXREF NAME="DropCapLetter" UID="217"/>
    <BOXREF NAME="DropCapBackground" UID="218"/>
</GROUP>

<BOX BOXTYPE="CT_TEXT" COLOR="White">
    <ID NAME="MainStoryText" UID="217"/>
    <TEXT>
        <STORY>
            <PARAGRAPH>
                <ANCHOREDBOXREF OFFSET="0">DropCapGroup
                </ANCHOREDBOXREF>
            </PARAGRAPH>
        </STORY>
    </TEXT>
</BOX>
```

The order of the `<BOXREF>` elements in a `<GROUP>` indicates the order in which the boxes were selected prior to grouping. The z-order of boxes in the layout is determined by the order of the `<BOX>` elements in the XML, from rearmost to frontmost.

Working with pictures

The `<PICTURE>` element supports a variety of 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>
```

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.

A text <BOX> element in a deconstructed project can also contain <PLACEHOLDER> elements, which allow XML Import XTensions software to insert text from a different XML source.

<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, 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, 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. For example:

```
<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."

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>
```

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 &hardReturn; entity to create the paragraph break. For example:

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

➡ 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.

➡ QuarkXPress Server returns copyfitting information for QuarkCopyDesk articles by default. To retrieve copyfitting information when deconstructing a QuarkXPress project, include `copyfitinfo=true` in the `xml` request.

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.

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>
```

Adding text and picture cells to tables

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.

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>
```

Merging and splitting 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>
```

Breaking 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" 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>
```

Working with sections

The Section feature lets you change the numbering system for a layout or a range of pages in a layout. To use this feature, you create a section start on a particular page. In that section start, you can specify a number format, a starting page number, and an optional prefix. For example:

```
<PAGE FORMATTEDNAME="A1" MASTER="A-Master A" POSITION="RIGHTOFSPINE">
    <ID UID="1"/>
    <SECTION FORMAT="ROMAN" NUMBER="1" PREFIX="A" OPERATION="CREATE"/>
</PAGE>
```

Once you have inserted a `<SECTION>` element, QuarkXPress Server will apply section-specific numbering and formatting to automatic page numbers. To insert automatic page numbers, use the `RICHTEXT@PAGENUMBERCHAR` attribute:

```
<TEXT>
    <STORY STORYDIRECTION="HORIZONTAL">
        <PARAGRAPH MERGE="false" PARASTYLE="Normal">
            <RICHTEXT MERGE="false">This is page </RICHTEXT>
            <RICHTEXT MERGE="false" PAGENUMBERCHAR="CURRENTPAGE"/>
            <RICHTEXT MERGE="false">. The story continues on page
              </RICHTEXT>
            <RICHTEXT MERGE="false" PAGENUMBERCHAR="NEXTPAGE"/>
            <RICHTEXT MERGE="false">. This story is continued from page
              </RICHTEXT>
            <RICHTEXT MERGE="false" PAGENUMBERCHAR="PREVIOUSPAGE"/>
        </PARAGRAPH>
    </STORY>
</TEXT>
```

To remove a section break, use XML like the following:

```
<PAGE FORMATTEDNAME="A1" MASTER="A-Master A" POSITION="RIGHTOFSPINE">
    <ID UID="1"/>
    <SECTION OPERATION="DELETE"/>
</PAGE>
```

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.

The content of each Composition Zones item is provided by a layout called the *composition layout*, which can be internal or external. Each `<COMPOSITIONZONE>` element includes a `TYPE` attribute that indicates whether its composition layout is internal or external.

- For internal Composition Zones items, each Composition Zones item is represented as an additional `<LAYOUT>` element within the `<PROJECT>` element. The `LAYOUTREF` element within the `<COMPOSITIONZONE>` element indicates the name of the `<LAYOUT>` that corresponds to that particular Composition Zones item.

- For external Composition Zones items, the PATH attribute indicates the location of the project containing the associated composition layout. However, a copy of the layout is also stored within the project as an additional <LAYOUT> element.

  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 BLENDSTYLE="SOLID" BOXTYPE="CT_USER" COLOR="none"
      LAYOUTREF="Layout 2" PATH="/projects/ExternalZone1.qxp" TYPE="EXTERNAL">

        <ID NAME="Box9" UID="9"/>
        ...
      </COMPOSITIONZONE>
    </SPREAD>
  </LAYOUT>
  <LAYOUT SHAREDSTATUS="ALLPROJECTS">
    <ID NAME="Layout 2" UID="2"/>
    <SPREAD>...</SPREAD>
  </LAYOUT>
</PROJECT>
```

Using XSL transformation

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.

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?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.

➡ 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`.

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.

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.

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>Text</RICHTEXT>
                            </TEXTPH>
                            <TEXTPH NAME="STANDFIRST">
                                   <PARAGRAPH PARACHAR="HARDRETURN"
                                      PARASTYLE="1st para"/>
                                   <RICHTEXT>Text</RICHTEXT>
                            </TEXTPH>
                            <TEXTPH NAME="BODY">
                                    <PARAGRAPH PARACHAR="HARDRETURN"
                                       PARASTYLE="Body"/>
                                    <RICHTEXT>Text</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>
```

- 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.

- 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.

Working with metadata

You can attach box-level metadata to a QuarkXPress project created from XML using the Modifier DTD. For example, if you import a picture from a content management system into a box, you can store the unique ID of that picture (and other information, such as the last-modified date) with the box containing that picture. When you deconstruct the project, you can read the metadata (for example, to track the usage of licensed pictures).

You can attach metadata to 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.

To create a new box with metadata, use XML like the following. In this example, QuarkXPress Server creates a box named "box1" and asssociates `Asset`, `Date`, and `Password` key-value pairs with it.

```
<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" PAGE="1">
        <POSITION>
            <TOP>5</TOP>
            <LEFT>5</LEFT>
            <BOTTOM>10</BOTTOM>
            <RIGHT>10</RIGHT>
        </POSITION>
    </GEOMETRY>
</BOX>
```

To delete metadata that is associated with a box, use XML like the following:

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

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.

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
```

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.

Using administrative request handlers

Administrative request handlers let you change the behavior of QuarkXPress Server. The built-in administrative request handlers are described in the topics below

You can add your own request handlers. During the `DDSSETUPCBCODE` callback, QuarkXPress Server XTensions software registers itself as a request handler via `AddCustomRequestHandler`, using the QuarkXPress Server XTensions API. The first parameter of this API is a pointer to a request handler function implemented in QuarkXPress Server XTensions software. The second parameter is a namespace string that identifies the request. When a user submits a request that has the same namespace string as a suffix to the request URL, QuarkXPress Server calls the request handler function with all the user-specified parameters in the `ServerRequest` structure. The request handler

function then processes the request and submits the reply in a `ServerReply` structure, which QuarkXPress Server communicates back to the user agent.

Addfile

Use the `addfile` request handler to put a document or image file in the document pool. An `addfile` request is always a POST request because it uses binary content.

If you send an `addfile` request to QuarkXPress Server Manager using HTTP or the Web services interface while the common doc pool switch is set to off in the QuarkXPress Server Manager client, the file is uploaded to all registered QuarkXPress Server instances. If the common doc pool is enabled, the file can be uploaded to any one registered QuarkXPress server instance.

Namespace	`addfile`		
Parameters	`uploadfile`	Binary file or MIME-type file	Contains the actual binary content of the file to be uploaded. This can be a QuarkXPress file, a Word file, a text file, or a file with a MIME-type such as EPS, JPEG, PNG, or PICT.
Response	The message "File upload completed."		
Alerts	The file system document pool is not enabled.	HTTP Error #404 This alert displays if you attempt to upload a document when the file system document pool is not enabled. *What to do*: Check **Enable File System Document Pool** in the **Server Configuration** dialog box.	
	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in theQuarkXPress Server Manager client **Server Configuration** dialog box.	
	Cannot find required volume or folder	HTTP Error #500 QuarkXPress Server Error #120 This alert displays if you attempt to upload a document that is in a subfolder that does not exist in the document pool while **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.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2004 19:24:12 — addfile/p1.qxp — Type: text/html — Size: 22 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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`		

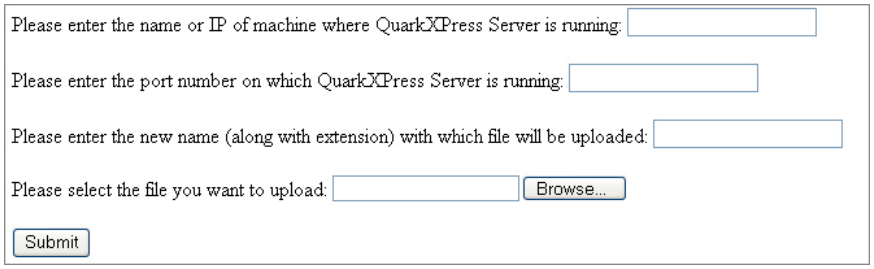
<table>
<tr><td>Example, object model</td><td>Request object name: AddFileRequest

```
//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);
```

The object model uses SOAP to transfer data, and 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).</td></tr>
<tr><td>Notes</td><td>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>
```
</td></tr>
</table>

The following example demonstrates how to use an HTML form to create a POST request that uses the `addfile` request handler. The form looks like this:

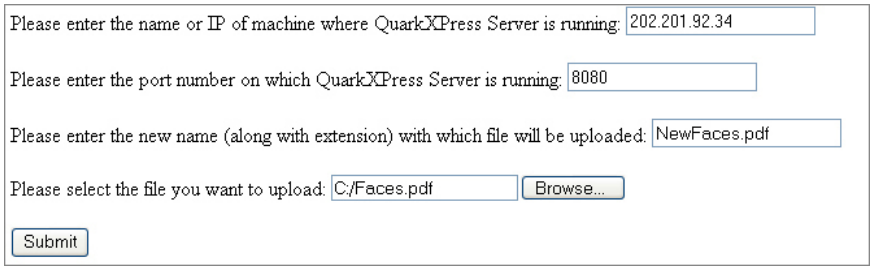


To use this form:

1. Enter the name or IP address of the computer on which QuarkXPress Server is running.
2. Enter the port number in the port number field.
3. Enter the file name along with the extension in the file field. Click **Browse** if you need to find the file on your computer. The file will be uploaded with this name.
4. Click **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, assume you want to upload a file named "Faces.pdf" (located at the root of the C drive) to an instance of QuarkXPress Server running at IP address 202.201.92.34 and port 8080, and that you want the name of the uploaded file on the server to be "NewFaces.pdf." Here's how you would accomplish this in the HTML form:



The HTML code to generate the above sample file is as follows:

```
<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">`

When you click Submit, the `UploadDocument()` function is called. This function is defined inside a script tag. It combines the information that has been entered in the form to create a URL for the `addfile` request, then sends this URL to QuarkXPress Server for processing. The code for 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>
```

Cplatform

The `cplatform` request handler provides Unicode support in QuarkXPress Server. It tells the server that the client browser/computer is running on a specified platform (Windows or Mac OS).

➡ 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	Indicates the platform on which on the request was generated. The valid values are "mac" for Mac OS and "win" for Windows.
Response	A preview of the document.		
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	The following Windows URL imports a series of Greek characters into the text box named "Story." `http://localhost:8080/sample.qxp?Story=  μ &clang=EL&cplatform=win`		

Example, Object Model	Request object name: `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	The Unicode languages supported by QuarkXPress Server are Finnish, Portuguese, Brazilian Portuguese, Spanish, Slovakian, Hungarian, Polish, Czech, Greek, Russian, Turkish, and Romanian.		

Delete

The `delete` request handler removes a specified document or folder from the document pool.

If you send a `delete` request to QuarkXPress Server Manager using HTTP or the Web services interface while the common doc pool switch is set to off in the QuarkXPress Server Manager client, the file or folder is uploaded to all registered QuarkXPress Server instances.

If the common doc pool is enabled, the file or folder can be deleted from any one registered QuarkXPress server instance.

Namespace	`delete`	
Response	The message "File deleted successfully."	
Alerts	File not found	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you try to delete a file that does not exist in the document pool.
	Folder cannot be deleted. It may still contain files.	HTTP Error #405 This alert displays if you try to delete a folder that is not empty. *What to do*: 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 displays if you try to delete an open file.
	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in theQuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 20:37:57 — delete/2000.qxp — Type: text/html — Size: 26 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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);`	

Clang

The `clang` request handler notifies QuarkXPress Server that the client browser/computer is running in a particular Unicode language. It also specifies that the values provided in

an HTTP request are in a particular Unicode language. The `clang` request handler works for all supported render types.

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

Parameters	`clang`	String	Specifies that the values provided in an HTTP request are in the specified Unicode language. The valid values are: 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 displays if you provide an invalid `clang` value.	
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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	The following URL imports a series of Greek characters into the text box named "Story." `http://localhost:8080/sample.qxp?Story=  μ &clang=EL`		

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	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. You can specify content in text boxes and the document pool path in the Unicode language. You can also import Unicode text into a text box. You can specify Unicode text inside TextModifier, DataImport, and Modifier XML. You can even provide file paths and box names in a Unicode language in ModifierXT and XML Import. If a document is created with Unicode text on one platform and the document is uploaded to another platform — for example, if a document is created on a Mac OS computer and is uploaded to Windows — some Unicode text may display differently. A file containing Unicode text must be saved in UTF-8 or UTF-16 format. If you import Unicode text into a box and apply a font that does not support the Unicode character set, the text may display incorrectly. This problem can also occur if the Font Fallback feature is turned off in QuarkXPress Server (**QuarkXPress Server > Preferences > Font Fallback**). You can send Unicode data to the server without `clang` and `cplatform` in the URL in the following ways: as URL-encoded UTF-8 strings in the URL, as UTF-8 encoded XML files in a construct/modify call, as UTF-8 text files in data import, and as Word/RTF files.

Fileinfo

The `fileinfo` request handler returns XML that contains the creation date, modification date, and file size of a document.

Namespace	`fileinfo`	
Parameters	`action=get`	Lets you retrieve the creation date of a file in UTC format. For example: `http://localhost:8080/fileinfo/sample.qxd?action=get&creationdate`
	`action=set`	Lets you set the creation and modification dates of a file in UTC format. For example: `http://localhost:8080/fileinfo/sample.qxp?action=set&creationdate=10-06-2007 12:12:37 UTC&modificationdate=10-06-2007 12:12:37 UTC`

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 displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.

Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 18:26:48 — fileinfo/Brochure_Base.qxd — Type: text/xml — Size: 191 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `fileinfo` parameter request.

Flush

The `flush` request handler flushes a document from the cache.

Namespace	`flush`	
Response	The message "CACHE FLUSH COMPLETED."	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 11/30/2005 17:32:45 — flush/project1 — Type: text/html — Size: 21 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `flush` parameter request.

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	The message "CACHE FLUSH COMPLETED."	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 11/30/2005 17:37:46 — flushall — Type: text/html — Size: 21 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `flushall` parameter request. When you issue a `flushall` request, the memory usage value in the status monitor becomes zero.	

Getdocinfo

The `getdocinfo` request handler returns XML information about a QuarkXPress project in the document pool. The returned information includes the project version, the platform on which it was saved, the number of layers, page properties, the length and width of the page in points, the number of pages, the names of imported picture files, the names of

any required fonts, the names and IDs of any relevant XTensions modules, and (for documents saved in QuarkXPress 6.0 or later) information about synchronized content.

<table>
<tr><td>Namespace</td><td colspan="2"><code>getdocinfo</code></td></tr>
<tr><td>Response</td><td colspan="2">The XML response looks like the following:
<pre><? 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></pre></td></tr>
<tr><td>Alerts</td><td>Incorrect administration realm user name and password.</td><td>HTTP Error #401
This alert displays if you specify an invalid administrator user name and password.
What to do: Use the user name and password set in the QuarkXPress Server Manager client Server Configuration dialog box.</td></tr>
<tr><td>Logs</td><td colspan="2">If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example:
8/3/2005 17:37:56 — getdocinfo/sample.qxp — Type: text/xml — Size: 590 — Client: 127.0.0.1
If an alert displays, an error message is written to the QuarkXPress Server error log file. For example:
8/3/2005 17:49:23 — Error — Error Code: 10022 — Incorrect administration realm user name and password.</td></tr>
<tr><td>Example GET URL</td><td colspan="2"><code>http://localhost:8080/getdocinfo/sample.qxp</code></td></tr>
<tr><td>Example, object model</td><td colspan="2">Request object name: <code>GetDocInfoRequest</code>
<pre>sdk.QRequestContext rc = new sdk.QRequestContext();

if(!this.DocumentSettings1.documentName.Text.Equals(""))</pre></td></tr>
</table>

	`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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `getdocinfo` parameter request.

Getdocpoollist

The `getdocpoollist` request handler returns an XML description of all files and folders in the local document pool, including name, size, type, modification date and time, and absolute and relative path.

Namespace	`getdocpoollist`	
Response	XML description of files and folders in the local document pool.	
Parameters	`directory`	Use this parameter to get information about a particular directory 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 displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/4/2005 10:34:48 — getdocpoollist — Type: text/xml — Size: 62098 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `fileinfo` parameter request.	

Getprefs

The `getprefs` request handler returns the current preference settings for QuarkXPress Server in XML format.

Namespace	`getprefs`

Response	An XML description of QuarkXPress Server preference settings.	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/4/2004 10:10:43 — getprefs — Type: text/xml — Size: 2636 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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` request handler returns preference settings for server configuration and Status Monitor. It does not return other preference settings, such as the settings for Deconstruct and PDF workflow. If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `getprefs` parameter request.	

Getprocessid

The `getprocessid` request handler returns the process IDs of the master QuarkXPress Server instance and of all subrender processes running on the computer.

Namespace	`getprocessid`

Response	An XML description of the process IDs of the master QuarkXPress Server instance and of all subrender processes running on the computer. For example:
	`<PROCESSID>`
	`<MASTER>`
	`<ID>3936</ID>`
	`<STATUS>BUSY</STATUS>`
	`</MASTER>`
	`<SUBRENDERERS>`
	`<SUBRENDERER>`
	`<ID>1736</ID>`
	`<STATUS>BUSY</STATUS>`
	`</SUBRENDERER>`
	`</SUBRENDERERS>`
	`</PROCESSID>`

Example GET URL	`http://localhost:8080/getprocessid`

Example, object model	Request object name: `????`
	`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);`

Getprojinfo

The `getprojinfo` request handler returns XML information about a QuarkXPress project in the document pool. The returned information identifies the operating system, the version of QuarkXPress in which the project was created, the size of the project, the page properties for the project's layouts, and information about named boxes and synchronized text.

Namespace	`getprojinfo`	

Response	The XML response looks like the following:
	`<? 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 displays if you specify a QuarkXPress 4.0 or 5.0 document.
	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password.

		What to do: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2005 17:38:03 — getprojinfo/sample.qxp — Type: text/xml — Size: 386 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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 saved in QuarkXPress 6.0 and later. If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `getprojinfo` parameter request.	

Getserverinfo

The `getserverinfo` request handler returns XML information about QuarkXPress Server. The returned information includes the platform on which QuarkXPress Server is running, the version of QuarkXPress Server, a list of installed fonts and server XTensions modules, the relevant XTensions server XTensions module IDs, the startup parameters, and the output styles with which the server is running. Disabled server XTensions modules are not listed in the response.

Namespace	`getserverinfo`	
Response	XML containing information about this QuarkXPress server instance.	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. What to do: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 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 displays, an error message is written to the QuarkXPress Server error log file. For example:	

	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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `getserverinfo` parameter request.

Preflight

Use the `preflight` request handler to check a project for missing fonts and missing pictures prior to output. You can also use this request handler to determine if the platform on which a project was created is different from the platform on which QuarkXPress Server is running.

Namespace	`preflight`	

Response	The XML response looks like the following:
	`<?xml version="1.0" encoding="UTF-8" standalone="no" ?>`
	`<PREFLIGHT>`
	`  <PLATFORMMISMATCH>TRUE</PLATFORMMISMATCH>`
	`  <MISSINGFONT>MidashiGoPro-MB31</MISSINGFONT>`
	`  <MISSINGPICTURE>/QuarkXPress Server Documents/images/illus_eps.eps`
	`  </MISSINGPICTURE>`
	`</PREFLIGHT>`	

Alerts	File not found	HTTP Error #404 QuarkXPress Server Error #–43 This alert displays if you try to delete a file that does not exist in the document pool.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 8/3/2004 19:24:12 — preflight/p1.qxp — Type: text/xml — Size: 22 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file.	

Example GET URL	To preflight a project in the root folder:
	`http://localhost:8080/preflight/abc.qxp`
	To preflight a binary file in a subfolder:
	`http://localhost:8080/preflight/sub1/abc.qxp`	

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

Setprefs

The `setprefs` request handler lets you set server preferences. To use this request handler, issue a `getprefs` request, determine the name of the tag that needs to be modified, and then submit a `setprefs` request with the using the name of this tag. For example, to turn off memory caching, you would first submit a `getprefs` request to the server. In the resulting XML, you would note that the name of the tag for memory caching tag is `AllowMemoryCaching`. Finally, you would submit a `setprefs` request to the server, like so: `http://localhost:8080/setprefs?AllowMemoryCaching=false`

Namespace	`setprefs`	
Response	The message "Preferences successfully set."	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 1/13/2006 16:04:28 — setprefs — Type: text/plain — Size: 29 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `setprefs` parameter request.

Shutdown

The `shutdown` request handler shuts down QuarkXPress Server. QuarkXPress Server then stops accepting new requests and shuts down after completing all pending transactions.

If you send this request to QuarkXPress Server Manager, a `shutdown` request is sent to all registered QuarkXPress servers.

Namespace	`shutdown`	
Response	The message "Shutdown request posted."	
Alerts	Incorrect administration realm user name and password.	HTTP Error #401 This alert displays if you specify an invalid administrator user name and password. *What to do*: Use the user name and password set in the QuarkXPress Server Manager client **Server Configuration** dialog box.
Logs	If the request succeeds, a transaction success message is written to the QuarkXPress Server transaction log file. This message includes the date, time, request type, project name, response type, response size in bytes, and client IP address. For example: 11/30/2005 17:40:16 — shutdown — Type: text/html — Size: 24 — Client: 127.0.0.1 If an alert displays, an error message is written to the QuarkXPress Server error log file. For example: 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	If a user name and password have been set in the **Server Configuration** dialog box, the browser requests that user name and password when you submit a `fileinfo` parameter request. 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.	

Modifier DTD (annotated)

The topics below provide an 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.

In the following topics:

- The "Construct" column refers to constructing a QuarkXPress project using the `construct` namespace.
- The "Modify" column refers to modifying a QuarkXPress project using the `modify` parameter.
- The "Deconstruct" column refers to deconstructing a QuarkXPress project using the `xml` namespace.

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

Entities (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct

`<!ENTITY etx "&#38;#x0003;">`
`<!ENTITY eot "&#38;#x0004;">`
`<!ENTITY enq "&#38;#x0005;">`
`<!ENTITY ack "&#38;#x0006;">`
`<!ENTITY softReturn "&#38;#x0007;">`
`<!ENTITY bs "&#38;#x0008;">`
`<!ENTITY hTab "&#38;#x0009;">`
	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.
			Entities that represent QuarkXPress special characters.
			Note: Some entities, such as `softreturn`, are different for QuarkXPress than they are in the Unicode specification.

Modifier DTD	Construct	Modify	Deconstruct
<!ENTITY lineFeed "
"> <!ENTITY vTab "&#x000B;"> <!ENTITY boxBreak ""> <!ENTITY hardReturn "
"> <!ENTITY so "&#x000E;"> <!ENTITY flexSpace "&#x000F;"> <!ENTITY dle "&#x0010;"> <!ENTITY dcOne "&#x0011;"> <!ENTITY dcTwo "&#x0012;"> <!ENTITY dcThree "&#x0013;"> <!ENTITY dcFour "&#x0014;"> <!ENTITY nak "&#x0015;"> <!ENTITY syn "&#x0016;"> <!ENTITY etb "&#x0017;"> <!ENTITY can "&#x0018;"> <!ENTITY em "&#x0019;"> <!ENTITY sub "&#x001A;"> <!ENTITY esc "&#x001B;"> <!ENTITY fs "&#x001C;"> <!ENTITY discReturn "&#x001D;"> <!ENTITY indentHere "&#x001E;"> <!ENTITY discHyphen "&#x001F;"> <!ENTITY shy ""> <!ENTITY ensp " "> <!ENTITY emsp " ">			

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
<!ENTITY threePerEmSpace " "> <!ENTITY fourPerEmSpace " "> <!ENTITY sixPerEmSpace " "> <!ENTITY figureSpace " "> <!ENTITY punctSpace " "> <!ENTITY thinsp " "> <!ENTITY hairSpace " "> <!ENTITY zeroWidthSpace ""> <!ENTITY hyphen "‐"> <!ENTITY ndash "–"> <!ENTITY mdash "—"> <!ENTITY wordJoiner ""> <!ENTITY nbsp " "> <!ENTITY ideographicSpace " ">			

PROJECT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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	Not applicable.	The name and path of the Job Jackets file associated

Modifier DTD	Construct	Modify	Deconstruct
	the Job Jackets 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.		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.

SAVEAS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

Modifier DTD	Construct	Modify	Deconstruct
`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 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.	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. 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.	Not applicable.

LAYOUT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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. 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`			

Modifier DTD	Construct	Modify	Deconstruct
`POINTSPERINCH CDATA #IMPLIED`	Not applicable.	Not applicable.	Specifies how many points to use per inch for measurements.
`MATHSUPERSET CDATA #IMPLIED`	Identifies the XPressMath superset (if any) used by this project.	Identifies the XPressMath superset (if any) used by this project.	Identifies the XPressMath superset (if any) used by this project.
`MEDIATYPE (PRINT | WEB | INTERACTIVE) #IMPLIED`	Not applicable.	Not applicable.	Specifies whether the layout is a Print, Web, or Interactive layout.
`SHAREDSTATUS (LAYOUT | ALLPROJECTS | THISPROJECT | none) #IMPLIED`	Not applicable.	Not applicable.	Specifies the sharing status of the layout, as specified in the **Layout > Advanced Layout Properties** dialog box in QuarkXPress.

ID (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT ID EMPTY>`	Lets you specify a name for a `LAYOUT`, `LAYER`, `BOX`, `LINKEDBOX`, `TABLE`, `GROUP`, or `COMPOSITIONZONE`. Lets you specify a unique ID for a `SPREAD` or `PAGE`.	Identifies an object by its `UID` or `NAME`. Note: QuarkXPress Server evaluates the ID element for a `NAME` value first and for a `UID` second. If a `NAME` is found, the `UID` is ignored.	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 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 `COMPOSITIONZONE`	The name of the `LAYOUT`, `LAYER`, `SPREAD`, `BOX`, `TABLE`, `GROUP`, or element to be modified.	The name of the parent element.

Modifier DTD	Construct	Modify	Deconstruct
	elements. QuarkXPress Server automatically assigns a UID to such elements. Ignored for 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.

LAYOUTPROPERTY (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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.

COLUMNGUIDES (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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.

ARTICLE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT ARTICLE (ID, RGBCOLOR?, COMPONENT+)>`	Describes an article (a series of one or more `COMPONENT` elements). New articles should not be created in a QuarkXPress project in systems working directly with QPS. Instead, create an article only within a QuarkCopyDesk® file. To assign an article in QPS®, use the QPS SDK.	Describes an article (a series of one or more `COMPONENT` elements).	Describes an article (a series of one or more `COMPONENT` 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.

COMPONENT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 automatically assigns a unique ID to components.	The unique ID of the `COMPONENT` to be modified.	Specified the unique ID of the .
`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	Specifies the ID of the QuarkXPress Server-assigned

Modifier DTD	Construct	Modify	Deconstruct
		box to which the COMPONENT belongs.	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.

SPREAD (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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.

PAGE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!ELEMENT PAGE (ID, SECTION?)>	A page to be created.	The page to be created or deleted. Note: To locate a page, for example, for creating a box, you use the GEOMETRY@PAGE attribute in the BOX element.	Indicates a page's absolute page number (in the ID@UID element) Note: Page 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	Identifies the master page from which to create a page. This value should be specified as a number, with	Identifies the master page that is applied to a page. Specified as a number, with "1" indicating the first master page.

Modifier DTD	Construct	Modify	Deconstruct
	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.	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.
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.
FORMATTEDNAME CDATA #IMPLIED	Not applicable.	Not applicable.	The string that displays in automatically created page numbers. A combination of the PREFIX, FORMAT, and NUMBER for this page's <SECTION> element.

SECTION (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!ELEMENT SECTION EMPTY>	Describes a section break in a layout.	Describes a section break in a layout.	Specifies a section in a QuarkXPress layout.
<!ATTLIST SECTION			
PREFIX CDATA #IMPLIED	The prefix to be added before each automatic page number inserted in this section.	The prefix to be added before each automatic page number inserted in this section.	The prefix to be added before each automatic page number inserted in this section.
OPERATION (CREATE | DELETE) #IMPLIED	Not applicable.	Specifies whether to create or delete the indicated section.	Not applicable.
FORMAT (NUMERIC | ROMAN | SMALLROMAN | ALPHA | SMALLALPHA | ASIANNUMBERS) #IMPLIED	The format of each automatic page number inserted in this section.	The format of each automatic page number inserted in this section.	The format of each automatic page number inserted in this section.
NUMBER CDATA #IMPLIED	The starting page number for this section.	The starting page number for this section.	The starting page number for this section.

BOX (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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@PAGE` attribute.	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 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 | DIAMOND | CIRCULAR |`	Specifies the type of blend applied to this box (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this box (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this box (linear, circular, rectangular, etc.).

Modifier DTD	Construct	Modify	Deconstruct
FULLCIRCULAR | none) "none"			
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.	Specifies the second color of the blend. The first color of the blend is the color applied to the box.	Specifies the second color of the blend. The first color of the blend is the color applied to the box.
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.
ANCHOREDGROUPMEMBER CDATA #IMPLIED	Specifies that this box is a member of the indicated anchored group.	Specifies that this box is a member of the indicated anchored group.	Specifies that this box is a member of the indicated anchored group.

METADATA (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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.

VALUE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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>

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
	<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.

TEXT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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

Modifier DTD	Construct	Modify	Deconstruct
`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) "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.
`TEXTALIGN (ASCENT | CENTER | BASELINE | DESCENT | none) "none"`	Specifies the part of a font to use for positioning characters on a line.	Specifies the part of a font to use for positioning characters on a line.	Indicates the part of a font being used for positioning characters on a line.
`TEXTALIGNWITHLINE (TOP | CENTER | BOTTOM | none) "none"`	Specifies how to align text to a line.	Specifies how to align text to a line.	Indicates text is aligned to a line.
`FLIPTEXT (true | false | none) "none">`	Flips the characters horizontally on a line.	Flips the characters horizontally on a line.	Indicates characters are horizontally flipped on a line.

INSET (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT INSET EMPTY>`	Specifies the distance between the inside border of a text box and the text.	Specifies the distance between the inside border of a text box and the text.	Indicates the distance between the inside border of a text box and the text.
`<!ATTLIST INSET`			
`MULTIPLEINSETS (true | false | none) "none"`	Specifies multiple insets.	Specifies multiple insets.	Indicates multiple insets.
`TOP CDATA #IMPLIED`	Specifies the distance between the top inside border of a text box and the text.	Specifies the distance between the top inside border of a text box and the text.	Indicates the distance between the top inside border of a text box and the text.
`BOTTOM CDATA #IMPLIED`	Specifies the distance between the bottom inside border of a text box and the text.	Specifies the distance between the bottom inside border of a text box and the text.	Indicates the distance between the bottom inside border of a text box and the text.
`RIGHT CDATA #IMPLIED`	Specifies the distance between the right inside border of a text box and the text.	Specifies the distance between the right inside border of a text box and the text.	Indicates the distance between the right inside border of a text box and the text.
`LEFT CDATA #IMPLIED`	Specifies the distance between the right inside border of a text box and the text.	Specifies the distance between the right inside border of a text box and the text.	Indicates the distance between the right inside border of a text box and the text.
`ALLEDGES CDATA #IMPLIED>`	Specifies the distance between the inside border of all sides of a text box and the text.	Specifies the distance between the inside border of all sides of a text box and the text.	Indicates the distance between the inside border of all sides of a text box and the text.

STORY (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT STORY (COPYFIT?, FITTEXT?, (PARAGRAPH | RICHTEXT | ANCHOREDBOXREF | LINKEDBOX | TEXTNODEPH | TEXTPH | HIDDEN | LIST | RUBI)*, OVERMATTER?)>`	Describes a text story in a text box or a chain of text boxes.	Describes a text story in a text box or a chain of text boxes.	Describes a text story in a text box or a chain of text boxes.
`<!ATTLIST STORY`			
`CLEAROLDTEXT (true | false) "true"`	Not applicable.	Clears any existing text from the box.	Not applicable.

Modifier DTD	Construct	Modify	Deconstruct
`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. Note: To control how text fits to a box on a story-by-story basis, use the `<FITTEXT>` element type (for more information, see "*FITTEXT (Modifier DTD)*").	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. Note: To control how text fits to a box on a story-by-story basis, use the `<FITTEXT>` element type (for more information, see "*FITTEXT (Modifier DTD)*").	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.

COPYFIT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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`

Modifier DTD	Construct	Modify	Deconstruct
			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.

PARAGRAPH (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT PARAGRAPH ((TABSPEC | RULE | FORMAT | RICHTEXT | ANCHOREDBOXREF | HIDDEN | GROUPCHARACTERS | RUBI)*)>`	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 Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	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 Jackets file or defined using the **Document Controls** submenu in QuarkXPress Server.	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 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.

Modifier DTD	Construct	Modify	Deconstruct
FAUXSTYLE (BOLD | ITALIC | BOLDITALIC | NONE) #IMPLIED	Not applicable.	Not applicable.	Indicates whether a paragraph contains a faux type style (such as a bold face that is constructed by software, as opposed to a bold font).

TEXTNODEPH (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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 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.

TEXTPH (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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			

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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.

GROUPCHARACTERS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT GROUPCHARACTERS ((RICHTEXT | HIDDEN)+)>`	Combines a series of characters into a unit always runs horizontally even if the story direction is vertical. Grouped characters do not break at the end of a line.	Combines a series of characters into a unit always runs horizontally even if the story direction is vertical. Grouped characters do not break at the end of a line.	Combines a series of characters into a unit always runs horizontally even if the story direction is vertical. Grouped characters do not break at the end of a line.
`<!ATTLIST GROUPCHARACTERS`			
`SCALEDIRECTION (HORIZONTAL | VERTICAL) #IMPLIED`	Specifies the direction in which text is scaled. Works only when the story direction is vertical.	Specifies the direction in which text is scaled. Works only when the story direction is vertical.	Specifies the direction in which text is scaled. Works only when the story direction is vertical.
`SCALEAMOUNT CDATA #IMPLIED`	Specifies the scaling percentage. Works only when the story direction is vertical.	Specifies the scaling percentage. Works only when the story direction is vertical.	Specifies the scaling percentage. Works only when the story direction is vertical.
`SENDING CDATA #IMPLIED`	Specifies the sending amount. (Sending is similar to kerning, but applicable as a fixed value over a range of text.)	Specifies the sending amount. (Sending is similar to kerning, but applicable as a fixed value over a range of text.)	Specifies the sending amount. (Sending is similar to kerning, but applicable as a fixed value over a range of text.)
`TRACKAMOUNT CDATA #IMPLIED`	Specifies the amount of tracking applied to text, in 1/200ths of an em space.	Specifies the amount of tracking applied to text, in 1/200ths of an em space.	Specifies the amount of tracking applied to text, in 1/200ths of an em space.

FORMAT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 amount of space in a paragraphs left indent.	Describes the amount of space in a paragraphs left indent.	Describes the 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 Controls** submenu in QuarkXPress Server.	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 Controls** submenu in QuarkXPress Server.	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.

Modifier DTD	Construct	Modify	Deconstruct
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.
MOJIGUMISET CDATA #IMPLIED	Identifies the mojigumi set (if any) applied to this paragraph.	Identifies the mojigumi set applied to this paragraph.	Identifies the mojigumi set (if any) applied to this paragraph.

KEEPLINESTOGETHER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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 in the paragraph or has a specific start and end.	Specifies whether this is for all lines in the paragraph or has a specific start and end.	Specifies whether this is for all lines 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	Specifies the number of lines at the end of a paragraph	Specifies the number of lines at the end of a paragraph

Modifier DTD	Construct	Modify	Deconstruct
	before wrapping text to keep lines together.	before wrapping text to keep lines together.	before wrapping text to keep lines together.

DROPCAP (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

LOCKTOGRID (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 is enabled.	Specifies whether LOCKTOGRID is enabled.	Specifies whether LOCKTOGRID 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.

TABSPEC (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT TABSPEC (TAB)+>`	Describes a group of tab stops.	Describes a group of tab stops.	Describes a group of tab stops.

TAB (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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">`			

RULE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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`			

Modifier DTD	Construct	Modify	Deconstruct
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 the right. A positive number moves the end-point to the left; a negative number moves the end-point to the right.	Specifies a distance to indent a rule farther from the right. A positive number moves the end-point to the left; a negative number moves the end-point to the right.	Specifies a distance a rule is indented farther 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	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

Modifier DTD	Construct	Modify	Deconstruct
		QuarkXPress Server, or an existing color created and saved in the project.	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 (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.

HIDDEN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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.	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.

Modifier DTD	Construct	Modify	Deconstruct
	XTensions that are not designed to handle inappropriate data may cause QuarkXPress Server to unexpectedly quit.	XTensions 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.

RICHTEXT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 Controls** submenu in QuarkXPress Server.	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.
`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.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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 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.	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).
`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.
`FONTSET CDATA #IMPLIED`	Identifies a font set that has been applied to text. Note that you can apply font sets during a Construct operation, but you cannot create them.	Identifies a font set that has been applied to text. Note that you can apply font sets during a Modify operation, but you cannot create them.	Identifies a font set that has been applied to text.
`FONTSETSIZE CDATA #IMPLIED`	Specifies the size of the font set that has been applied to text. (The base size of text	Specifies the size of the font set that has been applied to text. (The base size of text	Specifies the size of the font set that has been applied to text. (The base size of text

Modifier DTD	Construct	Modify	Deconstruct
	can be different from its font set size.)	can be different from its font set size.)	can be different from its font set size.)
`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 created and saved in the project.	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.
`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">   </RICHTEXT>`)	Specifies if the text will be nonbreaking or not. Used for special characters (e.g., for a thinspace: `<RICHTEXT NONBREAKING="true">   </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.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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.

Modifier DTD	Construct	Modify	Deconstruct
`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 to text.	Applies the OpenType tabular figures type style to text.	Identifies the OpenType tabular figures type style 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.

Modifier DTD	Construct	Modify	Deconstruct
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 notation). May also replace lowercase characters with alphabetic inferiors.	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.	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.
OT_ITALICS_FEATURE (true | false | none) "none"	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.	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.

Modifier DTD	Construct	Modify	Deconstruct
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 | none) "none" OT_JIS04FORMS (true | false | none) "none" OT_SIMPLIFIED_FORMS (true | false | none) "none" OT_TRADITIONAL_FORMS (true | false | none) "none"	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially designed for some Japanese OpenType fonts.	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially designed for some Japanese OpenType fonts.	Alternate glyph forms, such as JIS2004, JIS78, JIS90, Simplified, and Traditional. These glyph forms are specially 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 |	Specifies the dictionary preference used for hyphenation.	Specifies the dictionary preference used for hyphenation.	Identifies the dictionary preference used for hyphenation.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`Slovak | Russian | Romanian | Swedish | Turkish | Spanish | USEnglish | Catalan | Estonian | Lithuanian | Latvian | Icelandic | Slovenian | InternationalEnglish | SimplifiedChinese | TraditionalChinese | Japanese | Korean | Ukrainian | NynorskNorwegian | None | none) "none"`			
`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`	Unencoded Glyphs (UEG)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.	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.	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.
`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.

Modifier DTD	Construct	Modify	Deconstruct
`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.
`FAUXSTYLE (BOLD | ITALIC | BOLDITALIC | NONE) #IMPLIED`	Not applicable.	Not applicable.	Indicates whether the text contains a faux type style (such as a bold face that is constructed by software, as opposed to a bold font).
`PAGENUMBERCHAR (CURRENTPAGE | NEXTPAGE | PREVIOUSPAGE) #IMPLIED`	Represents an automatic page number character. If a `RICHTEXT` element with this attribute occurs in a section, section-specific numbering and formatting is applied to the page number. For more information, see "*Working with sections*."	Represents an automatic page number character. If a `RICHTEXT` element with this attribute occurs in a section, section-specific numbering and formatting is applied to the page number. For more information, see "*Working with sections*."	Represents an automatic page number character.

RUBITEXT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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 | RIGHT | LEFT) "ABOVE"`	This attribute 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).	This attribute 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).	This attribute 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.

ANCHOREDBOXREF (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 | BASELINE) "ASCENT"`	Determines whether the top of the anchored box will align with the top of the text	Determines whether the top of the anchored box will align with the top of the text	Determines whether the top of the anchored box will align with the top of the text

Modifier DTD	Construct	Modify	Deconstruct
	(ascent) or the bottom of the text (baseline).	(ascent) or the bottom of the text (baseline).	(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.

LINKEDBOX (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct

`<!ELEMENT LINKEDBOX (ID)>`	Represents a box or table cell into which text flows from the parent box. The child `ID` element identifies the box or table.
	To force text to run into the next box or cell in a chain, insert the `boxbreak` character entity where you want the text to break.
		Represents a box or table cell into which text flows from the parent box. The child `ID` element identifies the box or table.
		To force text to run into the next box or cell 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 or table cell where the text continues. Example:
			`<BOX>`
			`<ID NAME="Box5" UID="5"/>`
			`<TEXT>`
			`<STORY`
			`STORYDIRECTION="HORIZONTAL">`
			`<LINKEDBOX ENDOFFSET="94"`
			`STARTOFFSET="55">`
			`<ID NAME="Box6" UID="6"/>`
			`</LINKEDBOX>`
			`<LINKEDBOX ENDOFFSET="108"`
			`STARTOFFSET="95">`
			`<ID NAME="Box7" UID="7"/>`
			`</LINKEDBOX>`
			`<PARAGRAPH MERGE="false"`
			`PARASTYLE="Normal">`
			`<RICHTEXT MERGE="false">`
			`Text is here.</RICHTEXT>`
			`</PARAGRAPH>`
			`</STORY>`
			`</TEXT>`
			`</BOX>`

`<!ATTLIST LINKEDBOX`			
`STARTOFFSET CDATA #IMPLIED`	Not applicable.	Not applicable.	Offset of the first character in the next box or cell in the chain.
`ENDOFFSET CDATA #IMPLIED`	Not applicable.	Not applicable.	Offset of the last character in the next boxor cell in the chain.
`ROWCOUNT CDATA #IMPLIED`	If a `LINKEDBOX` is a table cell, this attribute identifies the row of the cell. Otherwise, not applicable.	If a `LINKEDBOX` is a table cell, this attribute identifies the row of the cell. Otherwise, not applicable.	If a `LINKEDBOX` is a table cell, this attribute identifies the row of the cell. Otherwise, not applicable.
`COLUMNCOUNT CDATA #IMPLIED`	If a `LINKEDBOX` is a table cell, this attribute identifies the column of the	If a `LINKEDBOX` is a table cell, this attribute identifies the column of the	If a `LINKEDBOX` is a table cell, this attribute identifies the column of the cell. Otherwise, not applicable.

Modifier DTD	Construct	Modify	Deconstruct
	cell. Otherwise, not applicable.	cell. Otherwise, not applicable.	

OVERMATTER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

PICTURE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 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.	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 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.	Not applicable.

Modifier DTD	Construct	Modify	Deconstruct
`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 value from –75 degrees to 75 degrees.	Specifies a skew angle for a picture as a floating-point value from –75 degrees to 75 degrees.	Specifies a skew angle for a picture as a floating-point 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, as an integer percentage from 0 to 100.	Specifies the opacity of a picture, as an integer percentage from 0 to 100.	Specifies the opacity of a picture, as an integer percentage from 0 to 100.
`PICBACKGROUNDCOLOR CDATA #IMPLIED`	Identifies the background color applied to a grayscale picture.	Identifies the background color applied to a grayscale picture.	Identifies the background color applied to a grayscale picture.
`PICBACKGROUNDSHADE CDATA #IMPLIED`	Specifies the shade of the background color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the shade of the background color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the shade of the background color applied to a grayscale picture, as an integer percentage from 0 to 100.

Modifier DTD	Construct	Modify	Deconstruct
PICBACKGROUNDOPACITY CDATA #IMPLIED	Specifies the opacity of the background color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the opacity of the background color applied to a grayscale picture, as an integer percentage from 0 to 100.	Specifies the opacity of the background color applied to a grayscale picture, 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 from being included in output.	Prevents a picture from being included in output.	Prevents a picture 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.
CLEARPICTURE (true | false) "false"	Not applicable.	Removes the picture (if any) from the box.	Not applicable.

CLIPPING (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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	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	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

Modifier DTD	Construct	Modify	Deconstruct
	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.	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.	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 CLIPPING@TYPE = ITEM or PICTUREBOUNDS. Moves the top edge of the clipping path by the specified number of points (positive=up, negative=down).	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).	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).
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 = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.	Valid when CLIPPING@TYPE = EMBEDDEDPATH, ALPHACHANNEL, or NONWHITEAREAS. Specifies a single outset or inset integer value in points to be used on all sides.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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) "none"`	Valid when `CLIPPING@TYPE` = `EMBEDDEDPATH`, `ALPHACHANNEL`, or `NONWHITEAREAS`. Indicates that only the outer edges of the clipping path should be used.	Valid when `CLIPPING@TYPE` = `EMBEDDEDPATH`, `ALPHACHANNEL`, or `NONWHITEAREAS`. Indicates that only the outer edges of the clipping path should be used.	Valid when `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.

SPLINESHAPE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 the shape is a pure rectangle.	Specifies whether the shape is a pure rectangle.	Specifies whether 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.

CONTOURS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

CONTOUR (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT CONTOUR (VERTICES)>`	A single contour within a spline shape.	A single contour within a spline shape.	A single contour within a spline 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.

VERTICES (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

VERTEX (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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 continuous.	Specifies whether the given vertex is also symmetrical — i.e., C2 continuous.	Specifies whether the given vertex is also symmetrical — i.e., C2 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.

LEFTCONTROLPOINT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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*.

VERTEXPOINT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 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*.

RIGHTCONTROLPOINT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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`.

Modifier DTD	Construct	Modify	Deconstruct
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.

GEOMETRY (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!ELEMENT GEOMETRY ((POSITION | RELPOSITION)? | MOVEUP | MOVEDOWN | MOVELEFT | MOVERIGHT | GROWACROSS | GROWDOWN | SHRINKACROSS | SHRINKDOWN | ALLOWBOXONTOPASTEBOARD | ALLOWBOXOFFPAGE | STACKINGORDER | SUPPRESSOUTPUT | RUNAROUND | LINESTYLE | SPLINESHAPE | FIT)*)>	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 = Elliptical box SH_LINE = Line SH_ORTHLINE = Orthogonal line (restricted to 45-degree angles) SH_SPLINEBOX = Freehand shape	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	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 90-degree angles) SH_SPLINEBOX = Freehand shape

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
	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.	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.	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. If the page number is followed by *, the box origin is on the left pasteboard. If the page number is followed by **, the box origin is on the right pasteboard. 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. If the page number is followed by *, the box origin is on the left pasteboard. If the page number is followed by **, the box origin is on the right pasteboard. 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. If the page number is followed by *, the box origin is on the left pasteboard. If the page number is followed by **, the box origin is on the right pasteboard. 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 value between –360 degrees and 360 degrees.	Specifies a rotation angle for a box or line as a floating-point value between –360 degrees and 360 degrees.	Specifies a rotation angle for a box or line as a floating-point 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.
CORNERSTYLE (ROUNDED | CONCAVE | RECTANGLE | BEVELED) #IMPLIED	Identifies the corner style (if any) applied to this box.	Identifies the corner style (if any) applied to this box.	Identifies the corner style (if any) applied to this box.
SKEW CDATA #IMPLIED	Specifies a skew value for the contents of this box or line as a floating-point value between –75 degrees and 75 degrees.	Specifies a skew value for the contents of this box or line as a floating-point value between –75 degrees and 75 degrees.	Specifies a skew value for the contents of this box or line as a floating-point value between –75 degrees and 75 degrees.

FIT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT FIT (MAX, MIN)>`	Lets you resize a box to fit its text or picture, within the limitations specified by the `<MAX>` and `<MIN>` elements. A box will expand or shrink only until it reaches the `<MIN>` or `<MAX>` size.	Lets you resize a box to fit its text or picture, within the limitations specified by the `<MAX>` and `<MIN>` elements. A box will expand or shrink only until it reaches the `<MIN>` or `<MAX>` size.	Not applicable.
`<!ATTLIST FIT`			
`POINT (TOPLEFT | TOP | TOPRIGHT | RIGHT | BOTTOMRIGHT | BOTTOM | BOTTOMLEFT | LEFT | CENTER) #REQUIRED`	Lets you specify the direction in which the box should be resized. To resize the box from the center, use `"CENTER"`.	Lets you specify the direction in which the box should be resized. To resize the box from the center, use `"CENTER"`.	Not applicable.
`AVOIDBOXESBY CDATA #IMPLIED`	Lets you specify the distance between the `POINT` side or corner of a resized box and any other items around it. A box will expand only until it is this distance from an adjacent item.	Lets you specify the distance between the `POINT` side or corner of a resized box and any other items around it. A box will expand only until it is this distance from an adjacent item.	Not applicable.
`PROPORTIONAL (true | false) "false"`	Lets you specify whether the resized box should have the same aspect ratio as the original box.	Lets you specify whether the resized box should have the same aspect ratio as the original box.	Not applicable.

MAX (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT MAX (LOCATION | SIZE | SCALETO)>`	Lets you specify the maximum location, size, or scale of a box for a fit-box-to-content operation.	Lets you specify the maximum location, size, or scale of a box for a fit-box-to-content operation.	Not applicable.

MIN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT MIN (LOCATION | SIZE | SCALETO)>`	Lets you specify the minimum location, size, or scale of a box for a fit-box-to-content operation.	Lets you specify the minimum location, size, or scale of a box for a fit-box-to-content operation.	Not applicable.

LOCATION (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT LOCATION EMPTY>`	Lets you specify the maximum or minimum location on the page of a box's upper-left corner for a fit-box-to-content operation.	Lets you specify the maximum or minimum location on the page of a box's upper-left corner for a fit-box-to-content operation.	Not applicable.
`<!ATTLIST LOCATION`			
`X CDATA #REQUIRED`	The largest or smallest allowable coordinate for the left side of the resized box.	The largest or smallest allowable coordinate for the left side of the resized box.	Not applicable.
`Y CDATA #REQUIRED`	The largest or smallest allowable coordinate for the top side of the resized box.	The largest or smallest allowable coordinate for the top side of the resized box.	Not applicable.

SIZE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT SIZE EMPTY>`	Lets you specify the maximum or minimum size of a box for a fit-box-to-content operation.	Lets you specify the maximum or minimum size of a box for a fit-box-to-content operation.	Not applicable.
`<!ATTLIST SIZE`			
`WIDTH CDATA #REQUIRED`	The largest or smallest allowable width for the resized box.	The largest or smallest allowable width for the resized box.	Not applicable.
`HEIGHT CDATA #REQUIRED`	The largest or smallest allowable height for the resized box.	The largest or smallest allowable height for the resized box.	Not applicable.

SCALETO (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT SCALETO EMPTY>`	Lets you specify the maximum or minimum size of a box for a fit-box-to-content operation.	Lets you specify the maximum or minimum size of a box for a fit-box-to-content operation.	Not applicable.
`<!ATTLIST SCALETO`			
`X CDATA #REQUIRED`	The largest or smallest allowable width for the resized box, as an integer percentage.	The largest or smallest allowable width for the resized box, as an integer percentage.	Not applicable.

Modifier DTD	Construct	Modify	Deconstruct
`Y CDATA #REQUIRED`	The largest or smallest allowable height for the resized box, as an integer percentage.	The largest or smallest allowable height for the resized box, as an integer percentage.	Not applicable.

RELPOSITION (Modifier DTD)

➡ To return item positions as `RELPOSITION` elements, use the `relativegeometry` parameter when deconstructing. For more information, see "*XML*."

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT RELPOSITION (ORIGIN, WIDTH, HEIGHT)>`	Specifies the position of a box or line, using coordinates measured in points from the upper-left corner of the page or spread.	Specifies the position of a box or line, using coordinates measured in points from the upper-left corner of the page or spread.	Specifies the position of a box or line, using coordinates measured in points from the upper-left corner of the page or spread.

ORIGIN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT ORIGIN EMPTY>`	Specifies an item's size and its position relative to the upper left corner of its page or spread.	Specifies an item's size and its position relative to the upper left corner of its page or spread.	Specifies an item's size and its position relative to the upper left corner of its page or spread.
`<!ATTLIST ORIGIN`			
`X CDATA #REQUIRED`	The distance between the left side of the item and the left edge of the page or spread.	The distance between the left side of the item and the left edge of the page or spread.	The distance between the left side of the item and the left edge of the page or spread.
`Y CDATA #REQUIRED`	The distance between the top side of the item and the top edge of the page or spread.	The distance between the top side of the item and the top edge of the page or spread.	The distance between the top side of the item and the top edge of the page or spread.
`RELATIVETO (PAGE | SPREAD) "SPREAD"`	Indicates whether the item's position is relative to the page or to the spread.	Indicates whether the item's position is relative to the page or to the spread.	Indicates whether the item's position is relative to the page or to the spread.

WIDTH (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT WIDTH (#PCDATA)>`	Indicates the width of an item.	Indicates the width of an item.	Indicates the width of an item.

HEIGHT(Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT HEIGHT (#PCDATA)>`	Indicates the height of an item.	Indicates the height of an item.	Indicates the height of an item.

POSITION (Modifier DTD)

Rather than using the `POSITION` element type, you can use the `RELPOSITION` element type to describe the position of `<GEOMETRY>` elements relative to the page or to the spread. To return item positions as `RELPOSITION` elements, use the `relativegeometry` parameter when deconstructing. For more information, see "*XML*."

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

MOVEUP (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

MOVEDOWN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

MOVELEFT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT MOVELEFT (#PCDATA)>`	Not applicable.	Moves a box to the left by the specified number of points.	Not applicable.

Modifier DTD	Construct	Modify	Deconstruct
		Note: You can move a box or line onto another page.	

MOVERIGHT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT MOVERIGHT (#PCDATA)>`	Not applicable.	Moves a box to the right by the specified number of points. Note: You can move a box or line onto another page.	Not applicable.

GROWACROSS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

GROWDOWN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

SHRINKACROSS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT SHRINKACROSS (#PCDATA)>`	Not applicable.	Shrinks a box horizontally to the left by the specified number of points.	Not applicable.

Modifier DTD	Construct	Modify	Deconstruct
		Note: A box can shrink on the same page or on other spreads and pages.	

SHRINKDOWN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

ALLOWBOXONTOPASTEBOARD (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT ALLOWBOXONTOPASTEBOARD (#PCDATA)>`	Not applicable.	Specifies whether 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.	Not applicable.

ALLOWBOXOFFPAGE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

STACKINGORDER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

SUPPRESSOUTPUT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

TOP (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT TOP (#PCDATA)>`	The distance between the box or lines top edge and the top of the page, in points.	The distance between the box or lines top edge and the top of the page, in points.	The distance between the box or lines top edge and the top of the page, in points.

LEFT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

BOTTOM (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT BOTTOM (#PCDATA)>`	The distance between the box or line's bottom edge	The distance between the box or line's bottom edge	The distance between the box or line's bottom edge

Modifier DTD	Construct	Modify	Deconstruct
	and the bottom of the page, in points.	and the bottom of the page, in points.	and the bottom of the page, in points.

RIGHT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

RUNAROUND (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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. 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	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	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

Modifier DTD	Construct	Modify	Deconstruct
	created based on the dark and light areas in the picture file. See the THRESHOLD attribute.	created based on the dark and light areas in the picture file. See the THRESHOLD attribute.	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, negative=left).	Valid when RUNAROUND@TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the runaround by the specified number of points (positive=right, negative=left).	Valid when RUNAROUND@TYPE = ITEM or PICTUREBOUNDS. Moves the right edge of the runaround by the specified number of points (positive=right, 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=AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this	Valid when RUNAROUND@TYPE=AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this	Valid when RUNAROUND@TYPE=AUTOIMAGE, ALPHACHANNEL, or NONWHITEAREAS. Specifies that areas smaller than this

Modifier DTD	Construct	Modify	Deconstruct
	number of points should be ignored when creating a runaround path.	number of points should be ignored when creating a runaround path.	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 edges of the runaround path should be used.	Valid when `RUNAROUND@TYPE=AUTOIMAGE`, `EMBEDDEDPATH`, `ALPHACHANNEL`, or `NONWHITEAREAS`. Indicates that only the outer edges of the runaround path should be used.	Valid when `RUNAROUND@TYPE=AUTOIMAGE`, `EMBEDDEDPATH`, `ALPHACHANNEL`, or `NONWHITEAREAS`. Indicates that only the outer 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.

LAYER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 **Visible** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether a layer is visible. Note: In QuarkXPress, this parameter overrides the **Visible** setting in the **Layers** pane of the **Preferences** dialog box (**QuarkXPress/Edit** menu).	Specifies whether a layer should be visible. Note: In QuarkXPress, this parameter 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 (**QuarkXPress/Edit** menu).	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).	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).

RGBCOLOR (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

LINESTYLE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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, arrow tail on left end `DOUBLEARROW` = Arrow heads on both ends	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, arrow tail on left end `DOUBLEARROW` = Arrow heads on both ends	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 end `DOUBLEARROW` = Arrow heads on both ends

CONTENTPH (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 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.

CONTENT (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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) "true"`	If true, straight quotation marks in an imported text file are converted to typesetter's quotation marks and double hyphens are converted to em dashes.	If true, straight quotation marks in an imported text file are converted to typesetter's quotation marks and double hyphens are converted to em dashes.	Not applicable.
`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.

Modifier DTD	Construct	Modify	Deconstruct
PAGETOIMPORT CDATA #IMPLIED	Indicates the page number of an imported PDF.	Indicates the page number of an imported PDF.	Indicates the page number of an imported PDF.
BOUNDINGBOX (MEDIABOX | CROPBOX | BLEEDBOX | TRIMBOX) #IMPLIED>	Identifies the bounding box type for an imported PDF. MEDIABOX includes the full imageable area. CROPBOX is the portion of the page that should be visible (not typically used for prepress). BLEEDBOX is the area defined by the crop marks, plus 3-5 additional millimeters. TRIMBOX is the area defined by the crop marks (in other words, the finshed page size).	Identifies the bounding box type for an imported PDF. MEDIABOX includes the full imageable area. CROPBOX is the portion of the page that should be visible (not typically used for prepress). BLEEDBOX is the area defined by the crop marks, plus 3-5 additional millimeters. TRIMBOX is the area defined by the crop marks (in other words, the finshed page size).	Identifies the bounding box type for an imported PDF. MEDIABOX includes the full imageable area. CROPBOX is the portion of the page that should be visible (not typically used for prepress). BLEEDBOX is the area defined by the crop marks, plus 3-5 additional millimeters. TRIMBOX is the area defined by the crop marks (in other words, the finshed page size).

SHADOW (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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 opacity of a drop shadow, specified as an integer percentage from 0 to 100.	Specifies the opacity of a drop shadow, specified as an integer percentage from 0 to 100.	Specifies the 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.

Modifier DTD	Construct	Modify	Deconstruct
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. The 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. The 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. The 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 0 to 144 points, with higher values creating blurrier edges.	Specifies the blur distance for a drop shadow, from 0 to 144 points, 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 shadow with the angles of other drop shadows in the layout.	Specifies whether to synchronize the angle of a drop shadow with the angles of other drop shadows in the layout.	Specifies whether to synchronize the angle of a drop 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	Specifies how a drop shadow is combined with its background. When true, the shadow color is combined with the background color or colors	Specifies how a drop shadow is combined with its background. When true, the shadow color is combined with the background color or colors

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
	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.	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.	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.
`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.

FRAME (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 in the project.	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.
`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.

Modifier DTD	Construct	Modify	Deconstruct
`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.

PLACEHOLDER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

TABLE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 columns.	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.	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.
`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`	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.).

Modifier DTD	Construct	Modify	Deconstruct
`| CIRCULAR | FULLCIRCULAR | none) "none"`			
`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 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.	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.	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.
`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`.
`ANCHOREDGROUPMEMBER CDATA #IMPLIED`	Specifies that this table is a member of the indicated anchored group.	Specifies that this table is a member of the indicated anchored group.	Specifies that this table is a member of the indicated anchored group.

PARENTTABLE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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.

TABLEBREAK (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

CHILDID (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

ADDCELLS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

Modifier DTD	Construct	Modify	Deconstruct
`<!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` attribute.	Not applicable.
`KEEPATTRIBUTE (true | false) "false">`	Not applicable.	Specifies whether an inserted row or column should adopt the same attributes as the `BASEINDEX` cell.	Not applicable.

DELETECELLS (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

COLSPEC (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

COLUMN (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 the second column from the left.	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.	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.
`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,	Specifies the opacity of the color applied to a column,	Specifies the opacity of the color applied to a column,

Modifier DTD	Construct	Modify	Deconstruct
	specified as an integer percentage from 0 to 100.	specified as an integer percentage from 0 to 100.	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.

ROW (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
<!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. Overrides the TABLE@COLOR attribute.	Identifies the color of a row. Overrides the TABLE@COLOR attribute.	Identifies the color of a row. Overrides the TABLE@COLOR attribute.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
	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.	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.	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.

HEADER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

FOOTER (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

CELL (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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 applied to a cell, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a cell, as an integer percentage from 0 to 100.	Specifies the shade of the color 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.

MODIFIER DTD (ANNOTATED)

Modifier DTD	Construct	Modify	Deconstruct
`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.

GRID (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

GRIDLINE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a line, as an integer percentage from 0 to 100.	Specifies the shade of the color applied to a line, 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.

GROUP (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT GROUP (ID, BOXREF*,GEOMETRY?, SHADOW?)>`	Not applicable.	Not applicable.	Describes a group of items.
`<!ATTLIST GROUP`			
`OPERATION (CREATE | DELETE) #IMPLIED`	Creates or deletes the group in the layout.	Creates or deletes the group in the layout.	Not applicable.
`ANCHOREDIN CDATA #IMPLIED>`	Not applicable.	Not applicable.	Indicates an anchored box in a text box and identifies its parent box.
`ANCHOREDGROUPMEMBER CDATA #IMPLIED`	Specifies that this group is a member of the indicated anchored group.	Specifies that this group is a member of the indicated anchored group.	Specifies that this group is a member of the indicated anchored group.

BOXREF (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT BOXREF EMPTY>`	Identifies a box that is a member of a `<GROUP>`.	Identifies a box that is a member of a `<GROUP>`.	Identifies a box that is a member of a `<GROUP>`.
`<!ATTLIST BOXREF`			
`UID CDATA #IMPLIED`	The `ID@UID` of a `<BOX>` that is part of a group.	The `ID@UID` of a `<BOX>` that is part of a group.	The `ID@UID` of a `<BOX>` that is part of a group.
`NAME CDATA #IMPLIED`	The `ID@NAME` of a `<BOX>` that is part of a group.	The `ID@NAME` of a `<BOX>` that is part of a group.	The `ID@NAME` of a `<BOX>` that is part of a group.
`OPERATION (CREATE | DELETE) #IMPLIED`	Creates or deletes the link that makes the target box part of a group. Note that deleting a `<BOXREF>` does not remove the corresponding box from the layout.	Creates or deletes the link that makes the target box part of a group. Note that deleting a `<BOXREF>` does not remove the corresponding box from the layout.	Not applicable.

COMPOSITIONZONE (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT COMPOSITIONZONE (ID, (FRAME | GEOMETRY | SHADOW)*)>`	Describes a Composition Zones item. (Applies only to the `xml` namespace.)		
`<!ATTLIST COMPOSITIONZONE`			

Modifier DTD	Construct	Modify	Deconstruct
`LAYOUTREF CDATA #IMPLIED`	Not applicable.	Not applicable.	Identifies the layout referenced by this Composition Zones item.
`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 #IMPLIED`	Not applicable.	Not applicable.	Indicates an anchored Composition Zones and identifies its parent Composition Zones.

Modifier DTD	Construct	Modify	Deconstruct
`BLENDSTYLE (SOLID | LINEAR | MIDLINEAR | RECTANGULAR | DIAMOND | CIRCULAR | FULLCIRCULAR | none) "none"`	Specifies the type of blend applied to this box (linear, circular, rectangular, etc.).	Specifies the type of blend applied to this box (linear, circular, rectangular, etc.).	Specifies the type of blend applied 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.	Specifies the second color of the blend. The first color of the blend is the color applied to the box.	Specifies the second color of the blend. The first color of the blend is the color applied to the box.
`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.

LIST (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!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.

RUBI (Modifier DTD)

Modifier DTD	Construct	Modify	Deconstruct
`<!ELEMENT RUBI (RUBITEXT, (RICHTEXT |`	Specifies a region of base text and the rubi text to include with that text. Note the	Specifies a region of base text and the rubi text to include with that text. Note the	Specifies a region of base text and the rubi text to include with that text. Note the

Modifier DTD	Construct	Modify	Deconstruct
`ANCHOREDBOXREF | HIDDEN)+)>`	second and subsequent children of the RUBI element `(RICHTEXT | ANCHOREDBOX | HIDDEN)+` declare the base text to which the rubi text is to be applied.	second and subsequent children of the RUBI element `(RICHTEXT | ANCHOREDBOX | HIDDEN)+` declare the base text to which the rubi text is to be applied.	second and subsequent children of the RUBI element `(RICHTEXT | ANCHOREDBOX | HIDDEN)+` declare the base text to which the rubi text is to be applied.

Sample applications

The topics below describe the sample applications distributed with QuarkXPress Server.

Sample applications: QXP Server Manager

These sample applications are available in the QuarkXPress Server installation package.

JSP samples: This sample application consists of Web pages that demonstrate ways in which the object model can be used to post QuarkXPress Server requests for various operations. To use these pages, put them in the `tomcat/webapps` directory. (If QuarkXPress Server Manager is installed, these samples are already available as a Web site in Tomcat at `tomcaturl/ClientSDKsamples`.)

To use this application, set the endpoint address for the Web service calls in the "web.xml" file: `web-app – >CLIENT_SDK_URL – >Value`. The files are located in the "ClientSDKSamples_JSP.zip" file.

ASP.NET samples: This sample application consists of Web pages that demonstrate different ways the object model can be used to post QuarkXPress Server requests for various operations. To use this application:

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

Sample applications legal notice

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 ©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 ©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 ©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, 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 ©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 ©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 ©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 ©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, Inc. All rights reserved. This Apple software is supplied to you by Apple, 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, 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 ©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.

Contacting 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: *EnterpriseSupport@quark.com*

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® 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 notices

Index

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z