StarOffice™ XML File Format Working Draft

Technical Reference Manual

Draft 9

December 2000

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

December 2000

Copyrights and Trademarks

Contents

Preface

About This Manual

This manual describes the StarOffice ™ XML file format. XML is the new native file format for the StarOffice ™ suite, replacing the old binary file format. Our goal is twofold: to have a complete specification encompassing all StarOffice components, and to provide an open standard for office documents. In our opinion, XML is ideal as an open standard because of the free availability of XML specifications and document type declarations (DTDs), and the XML support for XSL, XSLT, XLink, SVG, MathML, and many other important and emerging standards. One single XML format applies to different types of documents, for example, the same definition applies to tables in text documents and tables in spreadsheets.

This working draft manual contains the current specification of the StarOffice XML file format. As the term "working draft" implies, the StarOffice XML file format is work in progress. This fact has the following implications for this manual:

- The specification contained in this working draft is not complete. The XML specification for many of the StarOffice features has not yet been decided or documented.
- This working draft may contain specifications that are not currently implemented in the StarOffice XML import and export filters. This draft should also not omit specifications for any features that are already implemented in the StarOffice XML filters but there may be exceptions to this.
- The specifications described in this working draft may change. This is especially true for specifications that are not currently implemented in the StarOffice XML filters, but may also be the case for specifications that are already implemented. The reasons for changing the specifications include changes to related working drafts like XSL-FO or SVG, suggestions from reviewers of the manual, errors or inconsistencies that are found, or problems with new specifications that can only be resolved by changing existing specifications.
- This working draft may contain errors, missing information, or incomplete specifications.

Who Should Read This Manual

This manual is intended for software developers, both internal and external to Sun Microsystems®.

Structure of This Manual

This manual contains the following sections:

- Chapter 1, Introduction to StarOffice XML
- Chapter 2, Common Document Content

- Chapter 3, Text Content
- Chapter 4, Table Content
- Chapter 5, Graphic Content
- Chapter 6, Indexing
- Chapter 7, Chart Content
- Glossary
- Index

Related Documentation

The following documents provide additional XML-related information:

- Extensible Markup Language (XML) 1.0, W3C Recommendation http://www.w3.org/TR/REC-xml.html
- Scalable Vector Graphics (SVG) 1.0 Specification, W3C Working Draft http://www.w3.org/TR/2000/03/WD-SVG-20000303/index.html
- Namespaces in XML, World Wide Web Consortium http://www.w3.org/TR/REC-xml-names
- XSL Transformations (XSLT) Version 1.0, W3C Recommendation http://www.w3.org/TR/xslt
- XML Path Language (XPath) Version 1.0, W3C Recommendation http://www.w3.org/TR/xpath
- XML Linking Language (XLink) Version 1.0, W3C Candidate Recommendation http://www.w3.org/TR/xlink
- Extensible Stylesheet Language (XSL) Version 1.0, W3C Working Draft http://www.w3.org/TR/xsl
- HTML 4.01 Specification, W3C Recommendation http://www.w3.org/TR/html401
- ISO 8601, http://www.iso.ch/markete/8601.pdf
- ISO 639, http://www.oasis-open.org/cover/iso639a.html
- ISO 3166, http://www.oasis-open.org/cover/country3166.html

At the time of writing this document, some of these related documents are working drafts. Please note that any information from these drafts that is used in this document may change.

Conventions

The following conventions are used in this manual:

Convention	Description
italic type	Italic type is used to indicate complete titles of manuals and to emphasize text.
boldface type	Boldface type indicates an item that is contained in the glossary.
`courier font`	Courier font is used to indicate all XML elements and attributes and their values.

Terminology

The following terms are used frequently in this manual and have a specific meaning in the context of the manual:

Term	Meaning
Rules	This term is used in the tables that explain the StarOffice XML elements and attributes. In this context, the term Rules means the ways in which you can use the element or attribute, what specific values are acceptable and not acceptable, and any other specific points that you need to know about using the element or attribute.

CHAPTER **1**

Introduction to StarOffice XML

This chapter introduces the structure and basic design features of the StarOffice XML file format in StarOffice documents. The chapter contains the following sections:

- Namespaces
- Structure of StarOffice XML Documents
- Document Information
- Configuration Information
- Scripting
- Styles
- Forms
- Document Content
- White-Space Processing and EOL Handling
- Document Validation
-

1.1 Namespaces

lists the StarOffice XML namespaces and their prefixes. For more information about XML namespaces, please refer to the Namespaces in XML specification, located at http://www.w3.org/TR/REC-xml-names

Table 1: StarOffice XML Namespaces

Prefix	**Description**	**Namespace**
`office`	For all common pieces of information that are not contained in another, more specific namespace.	`http://openoffice.org/2000/office`
`style`	For elements and attributes that describe the style and inheritance model used by StarOffice as well as some common formatting attributes.	`http://openoffice.org/2000/style`
`script`	For elements and attributes that represent scripts or events.	`http://openoffice.org/2000/script`
`api`	For elements and attributes that are related to the StarOffice API.	`http://openoffice.org/2000/api`
`form`	For elements and attributes that describe forms and controls.	`http://openoffice.org/2000/form`
`text`	For elements and attributes that may occur within text documents and text parts of other document types, such as the contents of a spreadsheet cell.	`http://openoffice.org/2000/text`
`table`	For elements and attributes that may occur within spreadsheets or within table definitions of a text document.	`http://openoffice.org/2000/table`
`meta`	For elements and attributes that describe meta information.	`http://openoffice.org/2000/meta`
`number`	For elements and attributes that describe data style information.	`http://openoffice.org/2000/datastyle`
`draw`	For elements and attributes that describe graphic content.	`http://openoffice.org/2000/drawing`
`presentation`	For elements and attributes that describe presentation content.	`http://openoffice.org/2000/presentation`
`chart`	For elements and attributes that describe chart content.	`http://openoffice.org/2000/chart`
`xlink`	The **XLink** namespace.	`http://www.w3.org/1999/xlink`
`fo`	The **XSL** formatting objects and properties namespace.	`http://www.w3.org/1999/XSL/Format`
`svg`	The **SVG** namespace.	`http://www.w3.org/2000/svg`

1.2 Structure of StarOffice XML Documents

Each structural component in a StarOffice XML document is represented by an **element**, with associated **attributes**. The structure of XML documents applies to all StarOffice applications. There is no difference between a text document, a spreadsheet or a drawing, apart from the content. Also, all document types may contain different styles. You can exchange document content that is common to all document types from one type of document to another.

1.2.1 Document Root

The **document root element** is the primary element of a StarOffice XML document. It contains the entire document. All types of StarOffice XML documents use the same type of document root element, for example, text documents, spreadsheets and drawing documents.

XML Code: `<office:document>`

Rules: All StarOffice XML documents must have a document root element.

All other sections of a StarOffice XML document are represented by elements that are contained within the document root element.

DTD:

```
<!ELEMENT office:document (office:meta?, office:configs?,
                 office:scripting?,
                 office:font-decls?,
                 (office:styles|office:use-styles)?,
                 (office:automatic-styles|
                  office:use-automatic-styles)?,
                 (office:master-styles|
                  office:use-master-styles)?,
                 office:forms?, office:body)>
```

1.2.2 Primary Document Characteristics

You define primary document characteristics in the document root element using the following attributes:

- Class
- Version

Class

The `class` attribute identifies the document class of a StarOffice XML document. Document classes that you can assign are as follows:

- Text
- Online-text
- Spreadsheet
- Drawing
- Presentation

Although the document structure is the same for all document classes, most applications can only deal with a certain class of documents. For example, if you read a spreadsheet document using a word processor application there is always some loss of information. The class attribute enables an application to detect the document class without parsing the document. This is particularly useful in the following situations:

- When applications can deal with several document classes.
- When an **XSLT** transformation to another format should be applied, and there are several stylesheets available where each stylesheet is specific to a certain document class.

XML Code:	`office:class`
Rules:	You must specify a document class attribute for every StarOffice XML document.
DTD:	`<!ATTLIST office:document office:class (text|online-text|spreadsheet|drawing|presentation) #REQUIRED>`

Version

A StarOffice XML file can contain the version number of the file format. The version number is in the format `revision.version`. If the file has a version and the StarOffice application recognizes the DTD that belongs to this version, it may validate the document. Otherwise, the application does not need to validate the document, but the document must be well formed.

The `version` attribute provides the version number of the document.

XML Code:	`office:version`
Rules:	The `version` attribute is attached to the root element.
DTD:	`<!ATTLIST office:document office:version CDATA #IMPLIED>`
Notes:	The version number of the technical preview is 0.9. You can attach custom attributes to selected elements, which StarOffice application keeps in case the document is exported. If you use this feature, then there must be no version number, because this could result in validating errors that prevent the document from loading.

1.3 Document Information

In this manual, information about a StarOffice XML document is called **metadata**. Examples of metadata are:

- Document title
- Author
- Document creation date

You specify metadata within the `meta` element.

XML Code: `<office:meta>`

Rules: This element contains metadata.

DTD:

```
<!ENTITY % meta "(meta:generator?,dc:title?,
              dc:description?,dc:subject?,
              meta:initial-creator?,meta:creation-date?,
              dc:creator?,dc:date?,
              meta:printed-by?,meta:print-date?,
              dc:keywords?,meta:language?,
              meta:editing-cycles?,meta:editing-duration?,
              meta:target-frame?,meta:auto-reload?,
              meta:template?,meta:user-defined*)
<!ELEMENT office:meta %meta;>
```

Note: In the DTD, the element names correspond to the Dublin Core Element Set (http://purl.oclc.org/dc/). This is indicated by the `dc` namespace prefix.

1.4 Configuration Information

Configuration information provides information about the state of the application that created a document. This may contain the recommended printer device for printing the document or the name of a default database that is used if form controls are inserted. You use the `configs` element for configuration information about your StarOffice XML document.

XML Code:	`<office:configs>`
Rules:	The `configs` element contains configuration information which can be used by: – Any application. – A specific application, such as StarOffice Writer or StarOffice Calc. – A class of application, such as word processors or spreadsheets.
DTD:	`<!ELEMENT office:configs (office:config*)>`

1.4.1 Configuration Items

Configuration items are contained in the `config` element.

XML Code:	`<office:config>`
Rules:	The `config` element contains configuration information that is usable by: – Any application. – A specific application, such as StarOffice Writer or StarOffice Calc. – A certain class of application. The application or application class is specified by the `office:class` attribute.

DTD:	`<!ELEMENT office:config ANY>`
	`<!ATTLIST office:class CDATA "any">`

1.5 Scripting

The `scripting` element contains all information related to scripting in a document. The element contains the scripts and a table of all events that are global to the document. The `scripting` element is optional.

XML Code:	`<office:scripting>`
Rules:	If a document contains neither scripts nor events that are global to the document, then you omit the `scripting` element.
DTD:	`<!ELEMENT office:scripting (script:script+|(script:script*, script:events))>`

1.6 Styles

A StarOffice XML document contains the following types of **styles**:

- **Common styles**

The XML representations of the styles that are available in the StarOffice user interface are referred to as styles, or, where a differentiation from the other types of styles is required, they are referred to as common styles. The term "common" indicates that this is the type of style that a StarOffice user, who is not interested in the StarOffice XML file format, considers to be a style.

- **Automatic styles**
  An automatic style contains formatting properties that, in the user interface view of a document, are assigned to an object such as a paragraph. The term "automatic" indicates that the style is generated automatically at export time. In other words, formatting properties that are immediately assigned to a specific object are represented by an automatic style within a StarOffice XML document. This way, a separation of content and layout is achieved.

- **Master styles**
  A master style is a common style that contains formatting information and additional content that is displayed with the document content when the style is applied. An example of a master style is a StarOffice ™ Draw master page. In this case, the additional content is any shapes that are displayed as the background of the draw page. Another example of master styles are page masters. In this case, the additional content is the headers and footers. Please note that the content that is contained within master styles is additional content that influences the representation of a document but does not change the content of a document.

As far as the StarOffice user is concerned, all types of styles are part of the document. They represent the output device-independent layout and formatting information that the author of a document has used to create or edit the document. The assumption is that the author of the document wants this formatting and layout information to be preserved when the document is reloaded or displayed on a certain device, because this is common practice for documents created by word processors.

This type of style information differs from **CSS** or XSLT style sheets that are used to display a document. An additional style sheet for CSS, XSLT, and so on, is required to display a StarOffice XML document on a certain device. This style sheet must take into account the styles in the document as well as the requirements and capabilities of the output device. The ideal case is that this style sheet depends on the output device only.

1.6.1 Location of Styles

Common and automatic styles have the same StarOffice XML representation, but they are contained within two distinct container elements, as follows:

- `<office:styles>` for common styles
- `<office:automatic-styles>` for automatic styles

Master styles are contained within a container element of its own:

- `<style:master-styles>`

Common, automatic, and master styles can be contained within the physical XML file that contains the content of the document, or they can be contained within three separate StarOffice XML files:

- If the styles are contained within the XML file that contains the content, they are called **internal styles** and they are represented as children of the `<office:document>` element.
- If the styles are not contained in the XML file, they are called **external styles** and they are represented as root elements.

There cannot be internal and external styles of one kind simultaneously.

If any of the style container elements is not contained within the file that contains the document content, it must be referenced by one of the following elements:

- `<office:use-styles>`

- `<office:use-automatic-styles>`
- `<office:use-master-styles>`

1.6.2 Examples of Styles

The following examples illustrate the different types of StarOffice XML styles.

Example: Internal StarOffice XML styles

```
<office:document ...>
  <office:styles>
    ...
  </office:styles>
  <office:automatic-styles>
    ...
  </office:automatic-styles>
</office:document>
```

Example: External styles contained in three files residing in the same folder

File `styles.sxs`:

```
<office:styles ...>
    ...
</office:style>
```

File a`styles.sxs`:

```
<office:automatic-styles ...>
  ...
</office:automatic-style>
```

File `doc.sxw`:

```
<office:document ...>
  <office:use-styles xlink:type="simple" xlink:href="styles.sxs"/>
  <office:use-automatic-styles xlink:type="simple"
                               xlink:href="astyles.sxs"/>
</office:document>
```

Example: DTD

DTD:	`<!ENTITY % style "(style:style|text:list-style|text:outline-style|number:number-style|number:percentage-style|number:currency-style|number:date-style|number:time-style|number:boolean-style|number:text-style)">` `<!ELEMENT office:styles %style;*>` `<!ELEMENT office:automatic-styles (style:use-styles,%style;*)>` `<!ELEMENT office:master-styles (draw:master-page|style:page-master)*>` `<!ELEMENT office:use-styles EMPTY>` `<!ATTLIST office:use-styles xlink:href %url; #REQUIRED>` `<!ATTLIST office:use-styles xlink:type (simple) #FIXED "simple">` `<!ATTLIST office:use-styles xlink:actuate (onLoad) "onLoad">` `<!ELEMENT office:use-automatic-styles EMPTY>` `<!ATTLIST office:use-automatic-styles xlink:href %url; #REQUIRED>` `<!ATTLIST office:use-automatic-styles xlink:type (simple) #FIXED "simple">` `<!ATTLIST office:use-automatic-styles xlink:actuate (onLoad) "onLoad">` `<!ELEMENT office:use-master-styles EMPTY>` `<!ATTLIST office:use-master-styles xlink:href %url; #REQUIRED>` `<!ATTLIST office:use-master-styles xlink:type (simple) #FIXED "simple">` `<!ATTLIST office:use-master-styles xlink:actuate (onLoad) "onLoad">`
Note:	Styles in a document are linked to the document by an XML element instead of a `<?xml-stylesheet?>` processing instruction.

1.7 Forms

The `forms` element contains all of the forms in a StarOffice XML document.

XML Code:	`<office:forms>`
Rules:	Information to be supplied.
DTD:	`<!ELEMENT office:forms (form:form)*>`

1.8 Document Content

The `body` element contains the content of a document in one or more page sequences, as follows:

- The content distribution of text documents is specified in Section .
- A spreadsheet contains one page sequence for every table that is contained in the document.
- A drawing contains one page sequence for every page.

XML Code:	`<office:body>`
Rules:	*Information to be supplied.*
DTD:	`<!ELEMENT office:body ANY>`

1.9 White-Space Processing and EOL Handling

The **W3C** XML specification requires that white space characters are ignored for elements that have element content, in other words that contain elements but not text. This condition applies to the following white-space and end-of-line (**EOL**) Unicode characters:

- HORIZONTAL TABULATION (0x0009)
- LINE FEED (0x000A)
- CARRIAGE RETURN (0x000D)
- SPACE (0x0020))

For any other element, white-spaces are preserved by default. Unless otherwise stated, there is no special processing for any of the four white-space characters. For some elements, different white-space processing may take place, for example the paragraph element.

The XML specification also requires that any of the four white-space characters that is contained in an attribute value is normalized to a SPACE character.

One of the following characters may be used to represent line ends:

- LINE FEED
- CARRIAGE RETURN
- The sequence of the characters CARRIAGE RETURN and LINE FEED

Conforming to the XML specification, all the possible line ends are normalized to a single LINE FEED character.

As a consequence of the white-space and EOL processing rules, any CARRIAGE RETURN characters that are contained either in the text content of an element or in an attribute value must be encoded by the character entity `&#x0D;`. The same applies to the HORIZONTAL TABULATION and LINE FEED characters if they are contained in an attribute value.

1.10 Document Validation

In general, an XML document may be validated or not. If it is validated, it must match the DTD exactly. Sometimes it is not appropriate to validate an XML document, as in the following cases:

- Documents that are created by another version of StarOffice, or another application, than the current one.
- Documents that contain a custom extension.

Both kinds of documents may contain attributes, attribute values or elements that are unknown to the application that processes the file. The forwards-compatible processing rules describe how an application should handle such elements and attributes to get the most from the contents of the document.

1.10.1 Processing the Current Version

Validating and forward-compatible processing is controlled by the `version` attribute. Every application has a **current file format** version, which stores all the information contained in the document without losing any information when the document is read again. Beside this current version, an application may also be able to process documents with other versions. For simplicity, it is assumed that these versions are also covered by the concept of a current version. For every version, there is a specific DTD that may be used to validate documents.

shows the relationships between a current version of a document, consisting of a major version and a minor version, and the way it is processed by an application:

Table 1: Processing Relationships For Current Document Versions

If the major version of the document is...	**And/Or**	**Forward-compatible processing is...**	**Validation Status**
...the same as the current major version...	...and the minor version of the document is less or the same as the current minor version...	Disabled	The document may be validated, but it does not need to be.
...the same as the current major version...	...and the minor version of the document is greater than the current minor version...	Enabled	The document must not be validated. The only type of information that may be lost is information about features that are supported by the more recent version of StarOffice.
...different from the current major version...	...or if there is no version contained in the document at all...	Enabled	The document must not be validated.

CHAPTER **2**

Common Document Content

This chapter contains the following sections:

- Metadata
- Formatting Properties and Styles
- Page Styles and Layout
- Data Styles
- Frames
- Forms and Controls
- Hyperlinks
- Number Format
- Scripts
- Event Tables
- Change Tracking
- Configurations

2.1 Metadata

Metadata is general information about a document. In a StarOffice XML document, all of the metadata elements are contained in an `<office:meta>` element, usually located at start of the document.

XML Code:	`<office:meta>`
Rules:	This element contains metadata elements.
DTD:	see below

```
<!ENTITY % meta "(meta:generator?,dc:title?,
              dc:description?,dc:subject?,
              meta:initial-creator?,meta:creation-date?,
              dc:creator?,dc:date?,
              meta:printed-by?,meta:print-date?,
              dc:keywords?,meta:language?,
              meta:editing-cycles?,meta:editing-duration?,
              meta:target-frame?,meta:auto-reload?,
              meta:template?,meta:user-defined*,
              meta:document-statistic?)
<!ELEMENT office:meta %meta;>
```

2.1.1 Generator

The `<meta:generator>` element contains a string that identifies the application or tool that was used to create or last modify the XML document.

XML Code:	`<meta:generator>`
Rules:	If the application that created the document could not provide an identifier string, the application does not export this element. If another application modifies the document and it cannot provide a unique identifier, it is not allowed to export the original identifier belonging to the application that created the document.
DTD:	`<!ELEMENT meta:generator (#PCDATA)>`

2.1.2 Title

The `<dc:title>` element specifies the title of the document.

XML Code:	`<dc:title>`
Rules:	
DTD:	`<!ELEMENT dc:title (#PCDATA)>`

2.1.3 Description

The `<dc:description>` element contains a brief description of the document.

XML Code:	`<dc:description>`
Rules:	
DTD:	`<!ELEMENT dc:description (#PCDATA)>`

2.1.4 Subject

The `<dc:subject>` element specifies the subject of the document.

XML Code:	`<dc:subject>`
Rules:	
DTD:	`<!ELEMENT dc:subject (#PCDATA)>`

2.1.5 Keywords

The `<meta:keywords>` element contains keywords for the document. The metadata can contain any number of `<meta:keyword>` elements, each element specifying one keyword.

XML Code:	`<meta:keywords>`
Rules:	This element can contain one keyword.

DTD:	`<!ELEMENT meta:keywords (meta:keyword)*>`
	`<!ELEMENT meta:keyword (#PCDATA)>`

2.1.6 Initial Creator

The `<meta:initial-creator>` element specifies the name of the person who created the document initially.

XML Code:	`<meta:initial-creator>`
Rules:	
DTD:	`<!ELEMENT meta:initial-creator (#PCDATA)>`

2.1.7 Creator

The `<dc:creator>` element specifies the name of the person who last modified the document.

XML Code:	`<dc:creator>`
Rules:	
DTD:	`<!ELEMENT dc:creator (#PCDATA)>`
Notes:	The name of this element was chosen for compatibility with the Dublin Core.

2.1.8 Printed By

The `<meta:printed-by>` element specifies the name of the last person who printed the document.

XML Code:	`<meta:printed-by>`
Rules:	
DTD:	`<!ELEMENT meta:printed-by (#PCDATA)>`

2.1.9 Creation Date and Time

The `<meta:creation-date>` element specifies the date and time when the document was created initially.

XML Code:	`<meta:creation-date>`
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 20 for a pointer to ISO 8601.

DTD:	`<!ENTITY % c-date-time "(#PCDATA)">`
	`<!ELEMENT meta:creation-date %c-date-time;>`

2.1.10 Modification Date and Time

The `<dc:date>` element specifies the date and time when the document was last modified.

XML Code:	`<dc:date>`
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 20 for a pointer to ISO 8601.
DTD:	`<!ELEMENT dc:date %c-date-time;>`
Notes:	The name of this element was chosen for compatibility with the Dublin Core.

2.1.11 Print Date and Time

The `<meta:print-date>` element specifies the date and time when the document was last printed.

XML Code:	`<meta:print-date>`
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 20 for a pointer to ISO 8601.
DTD:	`<!ELEMENT meta:print-date %c-date-time;>`

2.1.12 Document Template

The `<meta:template>` element contains a URL for the document template that was used to create the document. The URL is specified as an XLink.

XML Code:	`<meta:template>`
Rules:	This element conforms to the XLink Specification. See page 20 for a pointer to this specification.
DTD:	`<!ELEMENT meta:template EMPTY>` `<!ATTLIST meta:template xlink:type (simple) #FIXED "simple">` `<!ATTLIST meta:template xlink:role CDATA #IMPLIED>` `<!ATTLIST meta:template xlink:actuate (onRequest) #IMPLIED>`

The attributes associated with the `<meta:template>` element are:

- Template location
- Template title
- Template modification date

Template Location

An `xlink:href` attribute specifies the location of the document template.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST meta:template xlink:href %url; #REQUIRED>`

Template Title

The `xlink:title` attribute specifies the name of the document template.

XML Code:	`xlink:title`
Rules:	
DTD:	`<!ATTLIST meta:template xlink:title CDATA #IMPLIED>`

Template Modification Date and Time

The `meta:date` attribute specifies the date and time when the template was last modified, prior to being used to create the current document.

XML Code:	`meta:date`
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 20 for a pointer to ISO 8601.

DTD:	`<!ENTITY %date-time "CDATA">`
	`<!ATTLIST meta:template meta:date %date-time; #IMPLIED>`

2.1.13 Automatic Reload

The `<meta:auto-reload>` element specifies whether a document is reloaded or replaced by another document after a certain period of time has elapsed.

XML Code:	`<meta:auto-reload>`
Rules:	
DTD:	`<!ELEMENT meta:auto-reload EMPTY>`

The attributes associated with the `<meta:auto-reload>` element are:

- Reload URL
- Reload delay

Reload URL

If a loaded document should be replaced by another document after a certain period of time, the `<meta:auto-reload>` element is presented as an XLink. An `xlink:href` attribute identifies the URL of the replacement document.

XML Code:	`xlink:href`
Rules:	

DTD:	`<!ATTLIST meta:auto-reload xlink:type (simple) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:show (replace) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:actuate (onLoad) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:href %url; #IMPLIED>`

Reload Delay

The `meta:delay` attribute specifies the reload delay.

XML Code:	`meta:delay`
Rules:	To conform with ISO 8601, the format of the value of this attribute is `PnYnMnDTnHnMnS`. See Section 5.5.3.2 of ISO 8601 for more detailed information on this time format. See page 20 for a pointer to ISO 8601.

DTD:	`<!ENTITY % duration "CDATA">`
	`<!ATTLIST meta:auto-reload meta:delay %duration; "P0S">`

2.1.14 Hyperlink Behavior

The `<meta:hyperlink-behaviour>` element specifies the default behavior for hyperlinks in the document.

XML Code:	`<meta:hyperlink-behaviour>`
Rules:	
DTD:	*To be supplied*

The attribute associated with the `<meta:hyperlink-behaviour>` element is:

- Target frame

Target Frame

The `meta:target-frame-name` attribute specifies the name of the default target frame in which to display a document referenced by a hyperlink.

XML Code:	`meta:target-frame-name`
Rules:	This attribute can have one of the following values: • `_self` : The referenced document replaces the content of the current frame. • `_blank` : The referenced document is displayed in a new frame. • `_parent` : The referenced document is displayed in the parent frame of the current frame. • `_top` : The referenced document is displayed in the topmost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name : The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<meta:hyperlink-behavour>` element. See page 20 for a pointer to the XLink Specification. If the value of the `meta:target-frame-name` attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the `meta:target-frame-name` attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.

DTD:	`<!ATTLIST meta:hyperlink-behaviour meta:target-frame-name CDATA #IMPLIED>`
	`<!ATTLIST meta:hyperlink-behaviour xlink:show (new|replace) #IMPLIED>`

2.1.15 Language

The `<dc:language>` element specifies the default language of the document.

XML Code:	`<dc:language>`
Rules:	The manner in which the language is represented is similar to the language tag described in RFC 1766 , located at http://info.internet.isi.edu/in-notes/rfc/files/rfc1666.txt. It consists of a two-letter Language Code taken from the ISO 639 standard optionally followed by a hyphen (-) and a two-letter Country Code taken from the ISO 3166 standard. See page 20 for pointers to ISO 639 and ISO 3166.
DTD:	`<!ENTITY % c-language "(#PCDATA)">` `<!ELEMENT dc:language %c-language;>`

2.1.16 Editing Cycles

The `<meta:editing-cycles>` element specifies the number of editing cycles the document has been through.

XML Code:	`<meta:editing-cycles>`
Rules:	The value of this element is incremented every time the document is saved. The element contains the number of editing cycles as text.
DTD:	`<!ENTITY % c-number "(#PCDATA)">` `<!ELEMENT meta:editing-cycles %c-number;>`

2.1.17 Editing Duration

The `<meta:editing-duration>` element specifies the total time spent editing the document.

XML Code:	`<meta:editing-duration>`
Rules:	The duration is represented in the manner described in Section 5.5.3.2 of ISO 8601. See page 20 for a pointer to ISO 8601.
DTD:	`<!ENTITY % c-duration "(#PCDATA)">` `<!ELEMENT meta:editing-duration %c-duration;>`

2.1.18 User-defined Metadata

The `<meta:user-defined>` element specifies any additional user-defined metadata for the document.

XML Code:	`<meta:user-defined>`
Rules:	Each instance of this element can contain one piece of user-defined metadata. The element contains: • A `meta:name` attribute, which identifies the name of the metadata element. • The value of the element, which is the metadata.
DTD:	`<!ELEMENT meta:user-defined (#PCDATA)>` `<!ATTLIST meta:user-defined meta:name CDATA #REQUIRED>`

2.1.19 Document Statistics

The `<meta:document-statistic>` element specifies the statistics of the document, for example, the page count, word count, and so on. The statistics are specified as attributes of the `<meta:document-statistic>` element and the statistics that are exported with the document depend on the document type and the application used to create the document.

Document Type	**Document Statistics Attributes**

Text	`meta:page-count`
	`meta:table-count`
	`meta:draw-count`
	`meta:ole-object-count`
	`meta:paragraph-count`
	`meta:word-count`
	`meta:character-count`
	`meta:row-count`

Spreadsheet	`meta:page-count`
	`meta:table-count`
	`meta:cell-count`
	`meta:object-count`

Graphic	`meta:page-count` `meta:object-count`

XML Code: `<meta:document-statistic>`

Rules:

DTD:

```
<!ELEMENT meta:document-statistic EMPTY>
<!ATTLIST meta:document-statistic meta:page-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:table-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:draw-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:ole-object-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:paragraph-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:word-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:character-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:row-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:cell-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:object-count %
positiveInteger; #IMPLIED>
```

2.1.20 Sample Metadata

Example: Sample metadata in a StarOffice XML document

```
<office:meta>
  <dc:title>Title of the document</dc:title>
  <dc:description>Description/Comment for the document</dc:description>
  <meta:initial-creator>User Name</meta:initial-creator>
  <meta:creation-date>1999-10-18T12:34:56</meta:creation-date>
  <dc:creator>User Name</dc:creator>
  <dc:date>1999-10-19T15:16:17</dc:date>
  <meta:printed-by>User Name</meta:printed-by>
  <meta:print-date>1999-10-20T16:17:18</meta:print-date>
  <dc:subject>Description of the document</dc:subject>
  <meta:duration-time>PT5H10M10S</meta:editing-duration>
  <meta:keywords>
    <meta:keyword>First keyword</meta:keyword>
    <meta:keyword>Second keyword</meta:keyword>
    <meta:keyword>Third keyword</meta:keyword>
  </meta:keywords>
  <meta:template xlink:type="simple"
    xlink:href="file:///c|/office52/share/template/german/finance/budget.
vor"
    xlink:title="Template name"
    meta:date="1999-10-15T10:11:12" />
  <meta:auto-reload
    xlink:type="simple"
    xlink:href="file:///..."
    meta:delay="P60S" />
  <dc:language>de-DE</dc:language>
  <meta:user-defined meta:name="Field 1">Value 1</meta:user-defined>
  <meta:user-defined meta:name="Field 2">Value 2</meta:user-defined>
</office:meta>
```

2.2 Formatting Properties and Styles

Many objects in a StarOffice document have formatting properties. A formatting property influences the visual representation of an object but it does not contribute to the content or structure of the document. Examples of formatting properties are:

- Font family
- Font size
- Font color
- Page margins

In a StarOffice XML document, formatting properties are only stored within styles. This differs from the StarOffice User Interface, where you can assign formatting properties to an object directly or you can apply a style to the object. Assigning formatting properties to an object directly has the same effect as assigning an unnamed style with the same properties to that object. Therefore, user interface styles remain unchanged conceptually in the StarOffice XML file format, while formatting properties assigned directly to an object are assumed to be unnamed styles. In order to use unnamed styles, they are assigned a name and therefore become automatic styles.

There are two main reasons for using styles to store formatting properties:

1. You can keep the format and layout of the document separate from the document content.
2. If two or more objects have the same formatting properties and styles assigned, the formatting properties that are assigned to the objects directly can be represented by a single automatic style for all objects. This saves disk space and allows styles to integrate seamlessly into the overall document style.

2.2.1 Formatting Property Sets

A document can contain several style elements. To acquire a common set of formatting properties, you use a `<style:properties>` element which is included as a child element of any style element. This container element offers two important advantages, as follows:

- Formatting properties can be addressed by CSS or XSL stylesheets regardless of the style type.
- Styles contain additional information that is not a formatting property, for example, the style name and parent style. It is good practice to separate this type of information.

XML Code:	`<style:properties>`
Rules:	You must use this container element to store a common set of formatting properties.
DTD:	`<!ELEMENT style:properties ANY>`

2.2.2 Simple Formatting Properties

Most formatting properties are simple and can be represented as attributes of the `<style:properties>` element. Where possible, XSL attributes are used to represent formatting properties. In this specification, the namespace prefix `fo` is used for XSL properties, that is properties that are part of the XSL-FO namespace. In general, formatting properties that cannot be represented by XSL properties are part of the `style` namespace.

In StarOffice, there are some formatting properties that you cannot specify without specifying one or more additional formatting properties. If the required properties are missing, a default value is assumed. This specification highlights the properties where this limitation applies.

Example: Simple style properties

This example shows a formatting property container that specifies an upper margin of 1 cm as well as a lower margin of 0.5 cm:

```
<style:properties fo:margin-left="1cm" fo:margin-bottom=".5cm"/>
```

2.2.3 Complex Formatting Properties

If a formatting property is too complex to be represented by XML attributes, it is represented by an XML element. Each such property is represented by an element type of its own.

Example: Complex formatting properties

This is an example of a formatting property container that specifies upper and lower margins as well as tab stop position at 2 and 4 cm.

```
<style:properties>
  <style:tab-stops>
     <style:tab-stop style:position="2cm"/>
     <style:tab-stop style:position="4cm"/>
  </style:tab-stops>
</style:properties>
```

2.2.4 Styles

Some style families are very similar in structure and can be represented by the same element. For example, the `<style:style>` element can represent paragraph, text, and frame styles.

XML Code:	`<style:style>`
Rules:	
DTD:	`<!ELEMENT style:style (style:properties?,style:map*)>`
Note:	The same elements can represent common and automatic styles but the difference is that they are contained in different container elements. An exception to this is automatically generated styles for elements contained in the `<office:styles>` elements, which are marked with the `style:automatic` attribute.

The attributes associated with the `<style:style>` element are:

- Style name
- Style family
- Automatic
- Parent style
- Next style
- List style
- Master page name
- Automatically update
- Formatting properties
- Outline level numbering

Style Name

The `style:name` attribute identifies the name of the style.

XML Code:	`style:name`
Rules:	
DTD:	`<!ENTITY % style-name "CDATA">` `<!ATTLIST style:style style:name %style-name; #IMPLIED>`
Notes:	This attribute, combined with the `style family` attribute, uniquely identifies a style. You cannot have two styles with the same family and the same name. For automatic styles, a name is generated during document export. If the document is exported several times, you cannot assume that the same name is generated each time. In an XML document, the name of each style is a unique name that is independent of the language selected for the StarOffice user interface. These style names are the same as the names used by the StarOffice API and are usually the names used for the English version of the user interface.

Style Family

The `style:family` attribute identifies the family of the style, for example, paragraph, text, or frame.

XML Code:	`style:family`
Rules:	
DTD:	`<!ATTLIST style:style style:family (paragraph|text|frame) #REQUIRED>`

Automatic

The `style:automatic` attribute specifies whether or not the style is an automatic style.

XML Code:	`style:automatic`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:style style:automatic %boolean; #IMPLIED>`

Parent Style

The `style:parent-style-name` attribute specifies the name of the parent style.

XML Code:	`style:parent-style-name`
Rules:	If a parent style is not specified, a default parent style defined by the application is used.
	The parent style cannot be an automatic style and if you specify a parent style that is not defined, an error occurs.
DTD:	`<!ATTLIST style:style style:parent-style-name %style-name; #IMPLIED>`

Next Style

The `style:next-style-name` attribute specifies the style to use as the next paragraph, text, or frame style.

XML Code:	`style:next-style-name`
Rules:	If you do not include this attribute, by default the current style is used as the next style.
	If you specify a next style that is undefined, an error occurs.
	If you specify a next style that is automatic, the attribute is ignored.
DTD:	`<!ATTLIST style:style style:next-style-name %style-name; #IMPLIED>`
Note:	This concept is only supported by some style families, including the paragraph styles family.

List Style

A paragraph style can have an associated list style. This applies to automatic and common styles.

XML Code:	`style:list-style-name`
Rules:	If you specify a list style, it is *only* applied to paragraphs that are contained in a list, where the list does not specify a list style itself, and the list has no list style specification for any of its parents.
DTD:	`<!ATTLIST style:style style:list-style-name %style-name; #IMPLIED>`

Master Page Name

A paragraph or table style can have an associated `style:master-page-name` attribute. This applies to automatic and common styles. If this attribute is associated with a style, a page break is inserted when the style is applied and the specified master page is applied to the preceding page.

XML Code:	`style:master-page-name`
Rules:	This attribute is ignored if it is associated with a paragraph style that is applied to a paragraph within a table.
DTD:	`<!ATTLIST style:style style:master-page-name %style-name; #IMPLIED>`

Automatically Update

The `style:auto-update` attribute determines whether or not styles are automatically updated when the formatting properties of an object that has the style assigned to it are changed. For example, if you have a paragraph style that contains a formatting property specifying that paragraph text is centered, and this paragraph style is applied to a paragraph. If you manually change the formatting of the paragraph text to be right-aligned and the value of the `style:auto-update` is `true,` the paragraph style is automatically updated to reflect the new paragraph formatting and every paragraph that uses the paragraph style is also modified to right-align the paragraph text.

XML Code:	`style:auto-update`
Rules:	This attribute can have a value of `true` or `false`.
DTD:	`<!ATTLIST style:style style:auto-update %boolean; "false">`
Implementation limitation:	This feature is only supported in StarOffice Writer documents.

Formatting Properties

If a style has formatting attributes assigned, the style element contains a formatting property container element called `<style:properties>`. See Section 2.2.1 for detailed information about this element.

Outline Numbering Level

See Section 3.6.2 for information on the outline numbering level for a style.

Sample Style

Example: StarOffice XML representation of the paragraph style "Text body"

```
<style:style style:name="Text body" style:family="paragraph"
             style:parent-style-name="Standard"
             style:pool-id="2049">
  <style:properties fo:margin-top="0cm" fo:margin-bottom=".21cm"/>
</style:style>
```

2.2.5 Style Mappings

The elements and attributes described in this section only apply to conditional styles.

The `<style:map>` element specifies the mapping to another style, if certain conditions exist.

XML Code:	`<style:map>`
Rules:	This element is contained in the style element of a conditional style. There is one element for every condition that the style uses.
DTD:	`<!ELEMENT style:map EMPTY>`
Implementation limitation:	Conditional styles are only supported by StarOffice Writer paragraph styles.

The attributes associated with the `<style:map>` element are:

- Condition
- Applied style
- Base cell address

Condition

The `style:condition` attribute specifies the condition in which a style map should be applied.

XML Code:	`style:condition`
Rules:	The value of this attribute is a Boolean expression. The syntax of the expression is similar to the XPath syntax. The following conditions are valid for paragraph styles: • `list-level()=`*n*, where *n* is a number between 1 and 10 • `outline-level()=`*n*, where *n* is a number between 1 and 10 • `table()` and `table-header()` • `section()` • `header()` and `footer()` • `footnote()` and `endnote()` • `Condition ::= TrueFunction | TrueCondition` • `TrueFunction ::= is-true-formula(Formula) | cell-content-is-between(Value, Value) | cell-content-is-not-between(Value, Value)` • `TrueCondition ::= Expression` • `Expression ::= GetFunction Operator Value` • `GetFunction ::= cell-content()` • `Operator ::= '<' | '>' | '<=' | '>=' | '=' | '!='` • `Value ::= NumberValue | String | Formula` **Note:** A `NumberValue` is a whole or decimal number. A `String` comprises one or more characters surrounded by quotation marks. A `Formula` is a formula (see 4.7.2) without the equals (=) sign at the beginning. You must include an `Operator`. The number in a `NumberValue` or `Formula` cannot contain comma separators for numbers of 1000 or greater. The conditions that apply for different types of styles may differ.
DTD:	`<!ATTLIST style:map style:condition CDATA #REQUIRED>`
Notes:	If an application detects a condition that it does not recognize, it must ignore the entire `<style:map>` element.

Applied Style

The `style:apply-style-name` attribute specifies the style to apply when the condition specified by the `style:condition` attribute is `true`.

XML Code:	`style:apply-style-name`
Rules:	If the referenced style is undefined or is an automatic style, an error occurs.
DTD:	`<!ATTLIST style:map style:apply-style-name %style-name #REQUIRED>`

Base Cell Address

The `style:base-cell-address` attribute specifies the base cell for relative addresses in formulas.

XML Code:	`style:base-cell-address`
Rules:	This attribute only applies to cell styles where the condition contains a formula.
	The value of this attribute must be an absolute cell address with a table name.
DTD:	`<!ATTLIST style:map style:base-cell-address %cell-address; #IMPLIED>`

Sample Style Mapping

Example: Style mapping

```
<style:style style:name="Text body" style:family="paragraph"
             style:parent-style-name="Standard"
             style:next-style-name="Text body">
  <style:properties fo:margin-top="0cm" fo:margin-bottom=".21cm"/>
  <style:map style:condition="footnote" style:apply-style-name="footnote"/>
  <style:map style:condition="heading(1)"
             style:apply-style-name="Heading 1"/>
  <style:map style:condition="heading(2)"
             style:apply-style-name="Heading 2"/>
</style:style>
```

2.3 Page Styles and Layout

The style and layout of the pages in a document is determined by:

- Page Masters
- Master Pages

A **page master** describes the physical properties or geometry of a page, for example, page size, margins, header height, and footer height.

A **master page** is a template for pages in a document. It contains a reference to a page master which specifies the physical properties of the page and can also contain static content that is displayed on all pages in the document that use the master page. Examples of static content are headers, footers, or background graphics.

If a text or spreadsheet document is displayed in a paged layout, the master pages are instantiated to generate a sequence of pages containing the document content. When a master page is instantiated, an empty page is generated with the properties of the page master and the static content of the master page. The body of the page is then filled with content. If multiple pages in a document use the same master page, the master page can be instantiated several times within the document.

In text and spreadsheet documents, you can assign a master page to paragraph and table styles using a `style:master-page-name` attribute. Each time the paragraph or table style is applied to text, a page break is inserted before the paragraph or table. The page that starts at the page break position uses the specified master page.

In drawings and presentations, you can assign master pages to drawing pages using a `style:parent-style-name` attribute.

Note: The StarOffice XML paging methodology differs significantly from the methodology used in XSL. In XSL, headers and footers are contained within page sequences that also contain the document content. The content of headers and footers can be changed or omitted without affecting the document content. In StarOffice XML, headers and footers are contained in page styles.

2.3.1 Page Master

The `<style:page-master>` element specifies the physical properties of a page.

XML Code:	`<style:page-master>`
Rules:	This element contains a `<style:properties>` element which specifies the formatting properties of the page and three optional elements that specify the properties of headers, footers, and a footnote section.
DTD:	`<!ELEMENT style:page-master (style:properties, style:header-style?, style:footer-style?, style:footnote-layout?)>`

The attributes that you can associate with the `<style:page-master>` element are:

- Name
- Page usage
- Page size
- Page number format
- Paper tray
- Print orientation
- Margins
- Border
- Border line width
- Padding
- Shadow
- Background
- Columns
- Register-truth
- Print
- Print page order
- First page number
- Scale

Name

The `style:name` attribute specifies the name of the page master.

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST style:page-master style:name %styleName #REQUIRED>`

Page Usage

The `style:page-usage` attribute specifies the type of pages that the page master should generate.

XML Code:	`style:page-usage`
Rules:	The value of this attribute can be: • `all` – all pages are the same • `left` – all pages are left pages • `right` – all pages are right pages • `mirror` – left and right pages are the same except that the margins are mirrored
DTD:	`<!ATTLIST style:page-master style:page-usage (all|left|right|mirrored) "all">`
Note:	XSL supports left and right pages but uses a different model to StarOffice XML.

Page Size

The `fo:page-width` and `fo:page-height` attributes specify the physical size of the page.

XML Code:	`fo:page-width` `fo:page-height`
Rules:	The `fo:page-width` attribute must correspond to the orientation of the page. For example, if a page is printed in portrait, the `fo:page-width` attribute specifies the width of the shorter page side. If the page is printed in landscape, the `fo:page-width` attribute specifies the width of the longer page side.

DTD:	`<!ATTLIST style:properties fo:page-width %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:page-height %length; #IMPLIED>`

Note:	XSL supports some more value that are not supported by StarOffice XML.

Page Number Format

You can specify a default number format for page styles, which is used to display page numbers within headers and footers. See Section 2.9 for detailed information on number format attributes.

XML Code:	`style:num-format` `style:num-letter-sync`
Rules:	The `style:num-format` attribute can be empty.

DTD:	`<!ATTLIST style:properties style:num-format CDATA #REQUIRED>`
	`<!ATTLIST style:properties style:num-letter-sync %boolean; "false">`

Paper Tray

The `style:paper-tray-number` attribute specifies the paper tray to use when printing the document. The numbers assigned to the printer trays depends on the printer.

XML Code:	`style:paper-tray-number`
Rules:	If the value of this attribute is `default`, the default tray specified in the printer configuration settings is used.

DTD:	`<!ENTITY % positiveNumberOrDefault "CDATA">`
	`<!ATTLIST style:properties style:paper-tray-number &positiveNumberOrDefault; #IMPLIED>`

Print Orientation

The `style:print-orientation` attribute specifies the orientation of the printed page.

XML Code:	`style:print-orientation`
Rules:	The value of this attribute can be `portrait` or `landscape`.
DTD:	`<!ATTLIST style:properties style:print-orientation (portrait|landscape) #IMPLIED>`

Margins

The margins attributes specify the size of the page margins. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:margin-top`
	`fo:margin-bottom`
	`fo:margin-left`
	`fo:margin-right`

Border

The border attributes specify the border properties of the page. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:border`
	`fo:border-top`
	`fo:border-bottom`
	`fo:border-left`
	`fo:border-right`

Note:	XSL does not have border properties for page masters.

Border Line Width

If a page contains borders, the border line width attributes specify the properties of the border lines of the page. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`style:border-line-width`
	`style:border-line-width-top`
	`style:border-line-width-bottom`
	`style:border-line-width-left`
	`style:border-line-width-right`

Padding

The padding attributes specify the padding properties of the page. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:padding`
	`fo:padding-top`
	`fo:padding-bottom`
	`fo:padding-left`
	`fo:padding-right`

Shadow

See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on this attribute.

XML Code:	`style:shadow`

Background

The background attributes specify the background properties of the page. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:background-color` and `<style:background-image>`
Note:	XSL does not have background properties for page masters.

Columns

The `<style:column>` attribute specifies if the page contains columns. See Paragraph Formatting Properties in Chapter 3 of this manual for detailed information on this attribute.

XML Code:	`<style:column>`

Register-truth

The `style:register-truth-ref-style-name` attribute references a paragraph style. The line distance specified of the paragraph style is used as the reference line distance for all paragraphs that have the register-truth feature enabled.

XML Code:	`style:register-truth-ref-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties style:register-truth-ref-style-name %styleName #IMPLED>`

Print

The `style:print` attribute specifies which components in a spreadsheet document to print.

XML Code:	`style:print`
Rules:	The value of this attribute is a list of the following values separated by blanks: • Headers • Grid • Annotations • Objects (graphics or OLE Objects) • Charts • Drawings • Formulas • Zero values
DTD:	`<!ATTLIST style:properties style:print CDATA #IMPLED>`

Print Page Order

The `style:print-page-order` attribute specifies the order in which data in a spreadsheet is numbered and printed when the data does not fit on one printed page.

XML Code:	`style:print-page-order`
Rules:	The value of this attribute can be `ttb` or `ltr`. Use `ttb` to print the data vertically from the left column to the bottom row of the sheet. Use `ltr` to print the data horizontally from the top row to the right column of the sheet.
DTD:	`<!ATTLIST style:properties style:print-page-order ("ttb" | "ltr") #IMPLED>`

First Page Number

The `style:first-page-number` attribute allows you to specify a number other than 1 for the first page.

XML Code:	`style:first-page-number`
Rules:	The value of this attribute can be an integer or `continue`. If the value is `continue`, the page number is the last page number incremented by 1.
DTD:	`<!ATTLIST style:properties style:first-page-number % positiveInteger; #IMPLED>`

Scale

The scale attributes specify how the application should scale the document for printing.

XML Code:	`style:scale-to` `style:scale-to-pages`
Rules:	You can use one the above attributes to specify how to scale the document for printing. If none of these attributes are present, the document is not scaled. If you use the `style:scale-to` attribute, the document is scaled to a percentage value. 100 percent is the normal value. You can enlarge or reduce all printed pages using this attribute. If you use the `style:scale-to-pages` attribute, you can specify the number of pages on which the the document should be printed. The document is then scaled to fit the defined number of pages.

DTD:	`<!ATTLIST style:properties style:scale-to %percentage; #IMPLED>`
	
	`<!ATTLIST style:properties style:scale-to-pages % positiveInteger; #IMPLED>`

2.3.2 Master Pages

This section of the manual describes the master page features that are supported by text and spreadsheets documents. The master pages used in drawings and presentations have some additional features that are described in section 5.2.

Master pages are contained within a master style element.

XML Code:	`<style:master-page>`
Rules:	In text and spreadsheet documents, this element contains the content of headers and footers. In drawings and presentations it contains background shapes.
DTD:	`<!ELEMENT style:master-page ( (style:header, style:header-left?)?, (style:footer, style:footer-left?)?, style:style*, (%shapes;)*, presentation:notes? )>`

The attributes that you can associate with the `<style:master-page>` attribute are:

- Page name
- Page master
- Next style name

Page Name

The `styles:name` attribute specifies the name of the master page. Each master page is referenced using its page name.

XML Code:	`styles:name`
Rules:	A page name is required for each master page and the name must be unique.
DTD:	`<!ATTLIST style:master-page style:name %styleName; #REQUIRED>`

Page Master

The `style:page-master-name` attribute specifies page master which contains the size, border, and orientation of the master page.

XML Code:	`style:page-master-name`
Rules:	This attribute is required for each master page.
DTD:	`<!ATTLIST style:master-page style:page-master-name %styleName; #REQUIRED>`

Next Style Name

The `style:next-style-name` attribute can be used to specify the master page used for the next page, if there is a next page.

XML Code:	`style:next-style-name`
Rules:	If the next style name is not specified, the current master page is used for the next page.
DTD:	`<!ATTLIST style:master-page style:next-style-name %styleName #IMPLIED>`
Note:	This attribute replaces the XSL page-sequence-master concept.

2.3.3 Headers and Footers

The header and footer elements specify the content of headers and footers. The `<style:header>` and `<style:footer>` elements contain the content of headers and footers. The two additional elements, `<style:header-left>` and `<style:footer-left>`, can be used to specify different content for left pages, if appropriate. If the latter two elements are missing, the content of the headers and footers on left and right pages is the same.

These elements are contained within a master page element.

XML Code:

```
<style:header>
<style:footer>
<style:header-left>
<style:footer-left>
```

Rules:

If the `style:page-usage` attribute associated with the page master has a value of `all` or `mirrored` and there are no `<style:header-left>` or `<style:footer-left>` elements, the header and footer content is the same for left and right pages.

If the `style:page-usage` attribute has a value of `left` or `right`, the `<style:header-left>` or `<style:footer-left>` elements are ignored.

The content of headers and footers is either:

- Standard text content, for example paragraphs, tables, or lists
- A sequence of any of the following elements; `<style:region-left>`, `<style:region-center>` and `<style:region-reight>`
- Empty, which switches off the display of all headers or footers. It is not possible to switch off the display of headers or footers for left pages only.

DTD:

```
<!ENTITY %hd-ft-content
         "( %text; |
            (style:region-left?|style:region-center?|
             style:region-right?) )">
<!ELEMENT style:header %hd-ft-content;>
<!ELEMENT style:footer %hd-ft-content;>
<!ELEMENT style:header-left %hd-ft-content;>
<!ELEMENT style:footer-left %hd-ft-content;>
```

Implementation limitation: StarOffice Writer only supports headers and footers that contain normal text, while StarOffice Calc only supports headers and footers that use the region elements.

2.3.4 Header and Footer Styles

The header and footer style elements specify the formatting properties for headers and footers on a page.

XML Code: `<style:header-style>` and `<style:footer-style>`

Rules: These elements must be contained within a page master element.

DTD:

```
<!ELEMENT style:header-style (style:properties?)>
<!ELEMENT style:footer-style (style:properties?)>
```

The attributes that you can associate with the header and footer elements are contained within a `<style:properties>` element. These attributes are:

- Fixed and minimum heights - see Section 2.6.9
- Left and right margins - see Section 2.6.9
- Bottom (for headers only) and top (for footers only) margins - see Section 2.6.9
- Borders - see Section 3.12.27 and 3.12.28
- Shadows – see Se ction 3.12.30
- Backgrounds – see Section 3.12.25 and 3.12.26

2.3.5 Footnote Layout

The footnote layout element specifies the layout for footnotes that are contained on a page.

XML Code:	`<style:footnote-layout>`
Rules:	This element must be contained in the page master element. It contains a `<style:properties>` element that specifies the maximum height and spacing of the footnote area and a `<style:footnote-sep>` element that specifies the separator line between the page body area and the footnote area. If this element is not present, a default footnote layout is used.
DTD:	`<!ELEMENT style:footnote-layout (style:properties?,style:footnote-sep?)>`

The attributes that you can associate with the `<style:footnote-layout>` element in the `<style:properties>` element are:

- Maximum height
- Spacing

Maximum Height

The `style:max-height` attribute specifies the maximum height of the footnote area.

XML Code:	`style:max-height`
Rules:	This attribute supports a value of `no-limit`, which allows the footnote area to increase until it equals the height of the page height.

DTD:	`<!ENTITY % lengthOrNoLimit "CDATA">`
	`<!ATTLIST style:properties style:max-height %lengthOrNoLimit #IMPLED>`

Spacing

The spacing attributes specify the distances before and after the line that separates the footnote area from the page body area.

XML Code:	`style:distance-before-sep` `style:distance-after-sep`
Rules:	These attributes are valid even if there is no footnote separator line specified.

DTD:	`<!ATTLIST style:properties style:distance-before-sep %length #IMPLED>`
	`<!ATTLIST style:properties style:distance-after-sep %length #IMPLED>`

2.3.6 Footnote Separator Line

This element specifies the separator line to use between the page body area and the footnote area.

XML Code:	`<style:footnote-sep>`
Rules:	This element can be contained in a `<style:footnote-layout>` element.
DTD:	`<!ELEMENT style:footnote-sep EMPTY>`

The attributes associated with the `<style:footnote-sep>` element are:

- Line width
- Line length
- Horizontal line alignment

Line Width

The `style:width` attribute specifies the width of the separator line.

XML Code:	`style:width`
Rules:	
DTD:	`<!ATTLIST style:footnote-sep style:width %length; #REQUIRED>`

Line Length

The `style:length` attribute specifies the length of the separator line.

XML Code:	`style:length`
Rules:	The value of this attribute is a percentage that relates to the width of the page excluding the page margins.
DTD:	`<!ATTLIST style:footnote-sep style:length %percentage; "100%">`

Horizontal Line Alignment

The `style:horizontal-align` attribute specifies how to horizontally align a line that is less than 100% long.

XML Code:	`style:horizontal-align`
Rules:	The value of this attribute can be `left`, `center`, or right
DTD:	`<!ATTLIST style:footnote-sep style:horizontal-align (left|center|right) "left">`

2.4 Font Declarations

In XSL and CSS, a font is described by its font family. The StarOffice XML file format also uses an additional set of attributes to describe a font. These additional attributes are evaluated if the font specified in the font family is not available, enabling the application to choose an alternative font. The additional attributes are:

- Style name
- Generic family

- Font pitch
- Character set

If a font is referenced, for example in a style, the additional font attributes can be either specified with the font family or using the font declarations element. A font declaration assigns a unique name to a set of font attributes. Font declarations help to reduce file sizes.

XML Code:	`<office:font-decls>`
Rules:	This element is a container for all font declarations. It must appear before any style container element.
DTD:	`<!ELEMENT office:font-decls (style:font-decl)*>`

2.4.1 Font Declaration

A font declaration assigns a set of font formatting properties to a unique font name

XML Code:	`<style:font-decl>`
Rules:	
DTD:	`<!ELEMENT style:font-decl EMPTY>`

The attributes that you associate with a `<style:font-decl>` element are:

- Font name
- Font properties

Font Name

This is the unique name of the font.

XML Code:	`style:name`
Rules:	
DTD:	`<!ENTITY % fontName "CDATA"`
	`<!ATTLIST style:font-decl style:name %fontName; #REQUIRED>`

Font Properties

The following font properties can be used to specify a font:

- Font family (see section)
- Font style name (see section)
- Generic font family (see section)
- Font pitch (see section)
- Font charset (see section)

The font family is required for every font declaration. All other properties are optional. See Text Formatting Properties in Chapter 3 of this manual for more information on these font properties.

XML Code:	`fo:font-family`
	`style:font-style-name`
	`style:font-family-generic`
	`style:font-pitch`
	`style:font-charset`

Rules:	

DTD:	`<!ATTLIST style:font-decl fo:font-family %string; #REQUIRED>`
	`<!ATTLIST style:font-decl style:font-style-name %string; #IMPLIED>`
	`<!ENTITY % fontFamilyGeneric "(roman|swiss|modern|decorative|script|system)">`
	`<!ATTLIST style:font-decl style:font-family-generic %fontFamilyGeneric; #IMPLIED>`
	`<!ENTITY % fontPitch "(fixed|variable)">`
	`<!ATTLIST style:font-decl style:font-pitch %fontPitch; #IMPLIED>`
	`<!ATTLIST style:font-decl style:font-charset CDATA #IMPLIED>`

2.5 Data Styles

Data styles describe how to display different types of data, for example, a number or a date. The elements and attributes that are used to represent data styles are contained in the namespace `http://openoffice.org/2000/datastyle`. The prefix `number` denotes the data styles namespace.

This section describes the StarOffice XML representation of the following data styles:

- Number style
- Currency style
- Percentage style
- Date style
- Boolean style
- Text style

2.5.1 Number Style

The `<number:number-style>` element describes the style for decimal numbers.

XML Code:	`<number:number-style>`
Rules:	This element can contain *one* of the following elements: • `<number:number>` • `<number:scientific-number>` • `<number:fraction>` These elements describe the display format of the number. The elements can be preceded or followed by `<number:text>` elements, which contain any additional text to be displayed before or after the number. In addition, this element can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % any-number "( number:number | number:scientific-number | number:fraction )">`
	`<!ENTITY % number-style-content "( number:text | (number:text?,%any-number;,number:text?) )">`
	`<!ELEMENT number:number-style ( style:properties?, %number-style-content;, style:map? )>`

The following elements can be used with the `<number:number-style>` element:

- Number
- Scientific number
- Fraction

Number

The `<number:number>` element specifies the display properties for a decimal number.

XML Code:	`<number:number>`
Rules:	This element is contained in the `<number:number-style>` element.
DTD:	`<!ELEMENT number:number EMPTY>`

See Section 2.5.10 for information on the attributes that you can associate with the number style elements.

Scientific Number

The `<number:scientific-number>` element specifies the display properties for a number style that should be displayed in scientific format.

XML Code:	`<number:scientific-number>`
Rules:	This element is contained in the `<number:number-style>` element.
DTD:	`<!ELEMENT number:scientific-number EMPTY>`

See Section 2.5.10 for information on the attributes that you can associate with the number style elements.

Fraction

The fraction element specifies the display properties for a number style that should be displayed as a fraction.

XML Code:	`<number:fraction>`
Rules:	This element is contained in the `<number:number-style>` element.
DTD:	`<!ELEMENT number:fraction EMPTY>`

See Section 2.5.10 for information on the attributes that you can associate with the number style elements.

2.5.2 Currency Style

The `<number:currency-style>` element describes the style for currency values.

XML Code:	`<number:currency-style>`
Rules:	This element can contain one `<number:number>` element and one `<number:currency-symbol>` element. It can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, this element can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % currency-symbol-and-text "number:currency-symbol, number:text?">`
	`<!ENTITY % number-and-text "number:number,number:text?">`
	`<!ENTITY % currency-style-content`
	`"( number:text |`
	`(number:text?,%number-and-text;,(currency-symbol-and-text)?) |`
	`(number:text?,%currency-symbol-and-text;,(number-and-text)?)`
	`)">`
	`<!ELEMENT number:currency-style ( style:properties?, %currency-style-content;, style:map? )>`

Currency Symbol

The `<number:currency-symbol>` element determines whether or not a currency symbol is displayed in a currency style.

XML Code:	`<number:currency-symbol>`
Rules:	The content of this element is the text that is displayed as the currency symbol. If the element is empty or contains white space characters only, the default currency symbol for the currency style or the language and country of the currency style is displayed. This element is contained in the `<number:currency-style>` element.
DTD:	`<!ELEMENT number:currency-symbol (#PCDATA)>`

If the currency symbol contained in a currency style belongs to a different language or country to that of the currency style, you can use the currency language and country attributes to specify the language and country of the currency symbol.

Currency Language and Country Attributes

XML Code:	`number:language` `number:country`
Rules:	

DTD:	`<!ATTLIST number:currency-symbol number:language CDATA #IMPLIED>`
	`<!ATTLIST number:currency-symbol number:country CDATA #IMPLIED>`

See Section 2.5.9 for information on the other attributes that you can associate with the currency style elements.

2.5.3 Percentage Style

The `<number:percentage-style>` element describes the style for percentage values.

XML Code:	`<number:percentage-style>`
Rules:	This element can contain one `<number:number>` element, which describes the display format for the percentage. The element can be preceded or followed by `<number:text>` elements, which contain any additional text to display before or after the percentage. In addition, the `<number:percentage-style>` element can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % percentage-style-content "( number:text | (number:text?,%number-and-text;) )">`
	`<!ELEMENT number:percentage-style ( style:properties?, %percentage-style-content;, style:map? )>`

Implementation limitation:	Currently, the StarOffice software requires this element to contain at least one `<number:text>` element and the text must contain a "%" character.

See Section 2.5.9 for information on the attributes that you can associate with the percentage style element.

2.5.4 Date Style

The `<number:date-style>` element describes the style for date values.

XML Code:	`<number:date-style>`
Rules:	This element can contain *one* instance of each of the following elements: `<number:day>`, `<number:month>`, `<number:year>`, `<number:era>`, `<number:day-of-week>`, `<number:week-of-year>`, `<number:quarter>`, `<number:hours>`, `<number:minutes>`, `<number:seconds>`, and `<number:am-pm>`. The `<number:date-style>` element can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % any-date "( number:day | number:month | number:year | number:era | number:day-of-week | number:week-of-year | number:quarter | number:hours | number:am-pm | number:minutes | number:seconds )">`
	`<!ENTITY % date-style-content "( number:text | (number:text?, (%any-date;,number:text?)+) )">`
	`<!ELEMENT number:date-style ( style:properties?, %date-style-content;, style:map? )>`
	
	Note: This DTD does not reflect the fact that some elements must not occur more than once.

See Section 2.5.9 for information on the attributes that you can associate with the date style elements.

The `<number:date-style>` element can contain the following elements:

- `<number:day>` – day of month
- `<number:month>` – month
- `<number:year>` – year
- `<number:era>` – era
- `<number:day-of-week>` – day of week
- `<number:week-of-year>` – week of year
- `<number:quarter>` – quarter

Day of Month

The `<number:day>` element specifies the day of the month in a date.

XML Code:	`<number:day>`
Rules:	If this element is used, it should be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:day EMPTY>`

The format attribute specifies whether the day of month element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For days, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the day of the month is displayed using one or two digits • `long` means that the day of the month is displayed using two digits
DTD:	`<!ATTLIST number:day number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Month

The month element specifies the month in a date.

XML Code:	`<number:month>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:month EMPTY>`

The textual representation attribute determines whether the name or number of a month is displayed in the month element of a date.

Textual Representation Attribute

XML Code:	`number:textual`
Rules:	If the value of this attribute value is `true`, the name of the month is displayed. If the attribute value is `false`, the number of the month is displayed.
DTD:	`<!ATTLIST number:month number:textual %boolean; "false">`

The `number:style` attribute specifies whether the month element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For months, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the abbreviated name of the month is displayed or the month is displayed using one or two digits • `long` means that the full name of the month is displayed or the month is displayed using two digits
DTD:	`<!ATTLIST number:month number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Year

The year element specifies the year in the date.

XML Code:	`<number:year>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:year EMPTY>`

The `number:style` attribute specifies whether the year element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For years, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the year is displayed using two digits • `long` means that the year is displayed using four digits
DTD:	`<!ATTLIST number:year number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Era

The era element specifies the era in which the year is counted.

XML Code:	`<number:era>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:era EMPTY>`

The `number:style` attribute specifies whether the era element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For eras, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the abbreviated era name is used • `long` means that the full era name is used
DTD:	`<!ATTLIST number:era number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Day Of Week

The day of week element specifies the day of the week in a date.

XML Code:	`<number:day-of-week>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:day-of-week EMPTY>`

The `number:style` attribute specifies whether the day of week element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For days of the week, the value of the `number:format-source` attribute is `fixed`: • `short` means that the abbreviated name of the day is displayed • `long` means that the full name of the day is displayed
DTD:	`<!ATTLIST number:day-of-week number:style (short|long) short>`

Week Of Year

The week of year element specifies the week of the year in the date.

XML Code:	`<number:week-of-year>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:week-of-year EMPTY>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Quarter

The quarter element specifies the quarter of the year in the date.

XML Code:	`<number:quarter>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:quarter EMPTY>`

The `number:style` attribute specifies whether the quarter element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For quarters, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the abbreviated name of the quarter is displayed, for example, Q1 • `long` means that the full name of the quarter is displayed, for example, Quarter 1
DTD:	`<!ATTLIST number:quarter-of-year number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

2.5.5 Time Style

The `<number:time-style>` element describes the style for time values.

XML Code:	`<number:time-style>`
Rules:	This element can contain *one* instance of any of the following elements: `<number:hours>`, `<number:minutes>`, `<number:seconds>` and `<number:am-pm>.` The `<number:time-style>` element can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % any-time "( number:hours | number:am-pm | number:minutes | number:seconds )">`
	`<!ENTITY % time-style-content "( number:text | (number:text?, (%any-time;,number:text?)+) )">`
	`<!ELEMENT number:time-style ( style:properties?, %time-style-content;, style:map? )>`
	
	Note: This DTD does not reflect the fact that some elements must not occur more than once.

See Section 2.5.9 for information on the attributes that you can associate with the time style elements.

The following elements can be contained in the `<number:time-style>` element:

- `<number:hours>` – hours
- `<number:minutes>` – minutes
- `<number:seconds>` – seconds
- `<number:am-pm>` – am/pm

Hours

The hours element specifies if hours are displayed as part of a date or time.

XML Code:	`<number:hours>`
Rules:	
DTD:	`<!ELEMENT number:hours EMPTY>`

The `number:style` attribute specifies whether the hours element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the time style. For hours, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the hours are displayed using at least one digit • `long` means that the hours are displayed using at least two digits
DTD:	`<!ATTLIST number:hours number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

Minutes

The minutes element specifies if minutes are displayed as part of a date or time.

XML Code:	`<number:minutes>`
Rules:	
DTD:	`<!ELEMENT number:minutes EMPTY>`

The `number:style` attribute specifies whether the minutes element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the time style. For minutes, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the minutes are displayed using at least one digit • `long` means that the minutes are displayed using at least two digits
DTD:	`<!ATTLIST number:minutes number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

Seconds

The seconds element specifies if seconds are displayed as part of a date or time.

XML Code:	`<number:seconds>`
Rules:	
DTD:	`<!ELEMENT number:seconds EMPTY>`

The `number:style` attribute specifies whether the seconds element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the time style. For seconds, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the seconds are displayed using at least one digit • `long` means that the seconds are displayed using at least two digits
DTD:	`<!ATTLIST number:seconds number:style (short|long) short>`

When you are displaying fractions, you can include fractions of seconds. The `number:decimal-places` attribute determines the number of decimal places to use when displaying fractions.

Decimal Places Attribute

XML Code:	`number:decimal-places`
Rules:	If this attribute is not used or if the value of the attribute is `0`, fractions are not displayed.
DTD:	`<!ATTLIST number:seconds number:decimal-places %number; "0">`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

AM/PM

The AM/PM element specifies if AM/PM is included as part of the date or time.

XML Code:	`<number:am-pm>`
Rules:	If a `<number:am-pm>` element is contained in a date or time style, hours are displayed using values from `1` to `12` only.
DTD:	`<!ELEMENT number:am-pm EMPTY>`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

2.5.6 Boolean Style

The `<number:boolean-style>` element describes the style for Boolean values.

XML Code:	`<number:boolean-style>`
Rules:	This element can contain one `<number:boolean>` element, which can be preceded or followed by `<number:text>` elements. In addition, it can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % boolean-style-content "(number:text| (number:text?, number:boolean,number:text?))">`
	`<!ELEMENT number:boolean-style ( style:properties?,%boolean-style-content;, style:map? )>`

Boolean

The `<number:boolean>` element contains the Boolean value of a Boolean style.

XML Code:	`<number:boolean>`
Rules:	
DTD:	`<!ELEMENT number:boolean EMPTY>`

See Section 2.5.9 for information on the attributes that you can associate with the Boolean style elements.

2.5.7 Text Style

The text style element describes the style for displaying text.

XML Code:	`<number:text-style>`
Rules:	This element can contain any number of `<number:text-content>` elements. It can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a `<style:properties>` element and a `<style:map>` element.
DTD:	`<!ENTITY % text-style-content "(number:text| (number:text?, number:text-content, number:text?))">` `<!ELEMENT number:text-style ( style:properties?,%text-style-content;, style:map? )>`
Notes:	The `<number:text-content>` elements represent the variable text content to display, while the `<number:text>` elements contain any additional fixed text to display.

See Section 2.5.9 for information on the attributes that you can associate with the text style elements.

Fixed Text

The `<number:text>` element contains any fixed text for a data style.

XML Code:	`<number:text>`
Rules:	This element is contained in the data styles element.
DTD:	`<!ELEMENT number:text (#PCDATA)>`

Text Content

The `<number:text-content>` element contains the text content of a text style.

XML Code:	`<number:text-content>`
Rules:	
DTD:	`<!ELEMENT number:text-content EMPTY>`

2.5.8 Common Data Style Elements

You can use some style elements with any of the primary data style elements. These elements are:

- Formatting properties
- Style mappings

Formatting Properties

The `<style:properties>` element specifies the text formatting properties to apply to any text displayed in the data style. See Section 2.2.1 for information on the formatting properties element.

Style Mappings

The `<style:map>` element specifies an alternative data style to map to if a certain condition exists. See Section 2.2.5 for information on the `<style:map>` element.

Rules for using this element with data style elements:	• This element must be the last child element in the data style element. • The style referenced by the `style:apply-style` attribute must be of the same type as the style containing the map. • The condition must be in the format `value()` *`op n`*, where *`op`* is a relational operator and *`n`* is a number. For Boolean styles the condition value must be `true` and `false`.

2.5.9 Common Data Style Attributes

Many of the data style attributes are applicable to more than one data style element. The following data style attributes are common to many of the data style elements:

- Name
- Language
- Country
- Title
- Volatility
- Automatic Order
- Format Source
- Time Value Truncation

Name

The `style:name` attribute specifies the name of the data style. It can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST number:number-style style:name %style-name; #REQUIRED>`

Language

The `number:language` attribute specifies the language of the style. The value of the attribute is a language code conforming with ISO639. StarOffice XML uses the language code to retrieve information about any display properties that are language-dependent. The language attribute can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:language`
Rules:	If a language code is not specified, either the system settings or the setting for the system's language are used, depending on the property whose value should be retrieved.
DTD:	`<!ATTLIST number:number-style number:language CDATA #IMPLIED>`

Country

The `number:country` attribute specifies the country of the style. The value of the attribute is a language code, conforming with ISO3166. StarOffice XML uses the country code to retrieve information about any display properties that are country-dependent. The language attribute can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:country`
Rules:	If a country is not specified, either the system settings or the setting for the system's country are used, depending on the property whose value should be retrieved.
DTD:	`<!ATTLIST number:number-style number:country CDATA #IMPLIED>`

Title

The `number:title` attribute specifies the title of the data style. It can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:title`
Rules:	
DTD:	`<!ATTLIST number:number-style number:title CDATA #IMPLIED>`

Volatility

Sometimes when a document is opened, not all of the styles are used. The unused styles can be retained or discarded; depending on the application you are using. The `style:volatile` attribute allows you to specify what to do with the unused styles. The volatility attribute can be used with any of the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`style:volatile`
Rules:	If the value of the attribute is `true`, the application keeps the style if possible. If the value is `false`, the application discards the unused styles.
DTD:	`<!ATTLIST number:number-style style:volatile %boolean; #IMPLIED>`
Note:	If a style is contained in a `<style:styles>` element, the default value of the `style:volatile` attribute is `true`. If a style is contained in a `<style:automatic-styles>` element, the default value is `false`.

Automatic Order

The `number:automatic-order` attribute can be used to automatically order data to match the default order for the language and country of the data style. This attribute is used with the following elements:

- `<number:currency-style>,` where number and currency symbols are reordered
- `<number:date-style>,` where the `<number:date-style>` child elements that are not `<number:text>` or `<style:properties>` elements are reordered

XML Code:	`number:automatic-order`
Rules:	The attribute value can be `true` or `false`.

DTD:	`<!ATTLIST number:currency-style number:automatic-order %boolean; "false">`
	or
	`<!ATTLIST number:date-style number:automatic-order %boolean; "false">`

Note:	If automatic ordering is enabled, but the language and country are not specified, the system settings for the order of numbers and currency symbols is used.

Format Source

The `number:format-source` attribute is used with the following elements:

- `<number:date-style>`
- `<number:time-style>`

This attribute specifies the source of the `short` and `long` display formats.

XML Code:	`number:format-source`
Rules:	The value of this attribute can be `fixed` or `language`. If the value is `fixed`, the application determines the value of `short` and `long`. If the value is `language`, the value of `short` and `long` is taken from the language and country of the style.
DTD:	`<!ATTLIST number:date-style number:format-source (fixed|language) fixed>`
Notes:	If the value of the `number:format-source` attribute is `language`, the meaning of short and short depends on the language and country of the date style, or, if neither of these are specified, StarOffice XML uses the system settings for short and long date and time formats.

Time Value Truncation

The `number:truncate-on-overflow` attribute is used with the `<number:time-style>` element. If a time or duration is too large to display using the default value range for a time component, (0 to 23 for `<number:hours>`), you can use the time value truncation attribute to specify if it can be truncated or the value range extended.

XML Code:	`number:truncate-on-overflow`
Rules:	
DTD:	`<!ATTLIST number:time-style number:truncate-on-overflow %boolean; "true">`

2.5.10 Common Number Style Attributes

Many of the number style attributes are applicable to more than one number style element. The following attributes are common to many of the number style elements:

- Decimal places
- Minimum integer digits
- Grouping separator
- Decimal replacement
- Minimum exponent digits
- Minimum numerator digits
- Minimum denominator digits
- Calendar system

Decimal Places

The `number:decimal-places` attribute specifies the number of decimal places to display. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`

XML Code:	`number:decimal-places`
Rules:	If this attribute is not specified, a default number of decimal places is used.
DTD:	`<!ATTLIST number:number number:decimal-places %number; #IMPLIED>`

Minimum Integer Digits

The `number:min-integer-digits` attribute specifies the minimum number of integer digits to display in a number, a scientific number, or a fraction. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`
- `<number:fraction>`

XML Code:	`number:min-integer-digits`
Rules:	If this attribute is not specified, a default number of integer digits is used.
DTD:	`<!ATTLIST number:number number:min-integer-digits %number; #IMPLIED>`

Grouping Separator

The `number:grouping` attribute specifies whether or not the integer digits of a number should be grouped using a separator character. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`
- `<number:fraction>`

XML Code:	`number:grouping`
Rules:	The grouping character that is used and the number of digits that are grouped together depends on the language and country of the style.
DTD:	`<!ATTLIST number:number number:grouping %boolean; "false">`

Decimal Replacement

If a number style specifies that decimal places are used but the number displayed is an integer, you can display replacement text instead of the decimal places. The `number:decimal-replacement` attribute specifies the replacement text. You can use this attribute with the `<number:number>` element.

XML Code:	`number:decimal-replacement`
Rules:	
DTD:	`<!ATTLIST number:number number:decimal-replacement CDATA #IMPLIED>`
Implementation limitation:	Currently, StarOffice only supports replacement text that consists of the same number of "-" characters as decimal places.

Minimum Exponent Digits

The `number:min-exponent-digits` attribute specifies the minimum number of digits to use to display an exponent. You can use this attribute with the `<number:scientific-number>` element.

XML Code:	`number:min-exponent-digits`
Rules:	
DTD:	`<!ATTLIST number:scientific-number number:min-exponent-digits %number; #IMPLIED>`

Minimum Numerator Digits

The `number:min-numerator-digits` attribute specifies the minimum number of digits to use to display the numerator in a fraction. You can use this attribute with the `<number:fraction>` element.

XML Code:	`number:min-numerator-digits`
Rules:	
DTD:	`<!ATTLIST number:fraction number:min-numerator-digits %number; #IMPLIED>`

Minimum Denominator Digits

The `number:min-denominator-digits` attribute specifies the minimum number of digits to use to display the denominator of a fraction. You can use this attribute with the `<number:fraction>` element.

XML Code:	`number:min-denominator-digits`
Rules:	
DTD:	`<!ATTLIST number:fraction number:min-denominator-digits %number; #IMPLIED>`

Calendar System

The `number:calendar` attribute specifies the calendar system used to extract parts of a date. You can use this attribute with the following elements:

- `<number:day>`
- `<number:month>`
- `<number:year>`
- `<number:era>`
- `<number:day-of-week>`
- `<number:week-of-year>`
- `<number:quarter>`

XML Code:	`number:calendar`
Rules:	If this attribute is not specified, the default calendar system is used.
DTD:	`<!ATTLIST number:day number:calendar CDATA #IMPLIED>`

2.6 Frames

A **frame** is a rectangular container where you can place content that you want to position outside the default text flow of a document. In StarOffice documents, frames can contain:

- Images
- Drawings
- Text boxes (StarOffice Writer documents only)
- Applets
- Floating frames
- Plug-ins
- StarOffice objects and common OLE objects

A frame has properties that apply to:

- The area around the frame or the frame neighborhood, for example, the anchor type, position or wrap mode.
- The frame content only, for example, the URL for a picture.

- Both frame neighborhood and content, for example, the frame size.

StarOffice XML does not differentiate between the different types of frame properties. Frame formatting properties are stored in an automatically generated style belonging to the `graphics` family. The way a frame is contained in a document depends on the file format type and is explained in the application-specific chapters of this document.

There are several elements used to represent the different frame types. In this manual, these elements are called **frame elements** .This section describes the following frame types and the elements used to represent them:

- Text Boxes
- Images
- Drawings
- Controls
- Plug-ins, applets, and floating frames
- Objects

2.6.1 Text Boxes

You can use a text box to place text in a container that is outside of the normal flow of the document.

XML Code:	`<draw:text-box>`
Rules:	
DTD:	`<!ELEMENT draw:text-box (%frame;*,%text;)>`
Note:	SVG does not support text boxes, while XSL has limited support for text boxes.

The attributes that you can include with the text boxes element are:

- Name
- Style
- Chain
- Position, size, and transformation (see Section 5.4.13)
- Layer ID (see Section 2.6.8)
- Z Index (see Section 2.6.8)

Name

The `draw:name` attribute specifies the name of the text box.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:text-box text:name CDATA #IMPLIED>`
Implementation limitation:	This attribute is only supported by StarOffice Writer. If a text box without a name is loaded into StarOffice Writer, it is inserted as a drawing text box.

Style

The `draw:style-name` attribute specifies the name of the style for the text box.

XML Code:	`draw:style-name`
Rules:	This attribute links to a `<style:style>` element belonging to the `graphic` style family.
DTD:	`<!ATTLIST draw:text-box draw:style-name &style-name; #REQUIRED>`

Chain

Text boxes can be chained, in other words, if the content of a text box exceeds its capacity, the content flows into the next text box in the chain.

XML Code:	`style:chain-next-name`
Rules:	The value of this attribute is the name of the next text box in the chain.
DTD:	`<!ATTLIST style:properties style:chain-next-name CDATA #IMPLIED>`
Implementation limitation:	Chained text boxes are only supported by StarOffice Writer.

2.6.2 Images

An image can be either:

- Contained in a StarOffice document as a link to an external resource

or

- Embedded in a StarOffice document

XML Code:	`<draw:image>`
Rules:	This element is an XLink, and therefore has some attributes with fixed values that describe the link semantics.
DTD:	`<!ELEMENT draw:image (svg:desc?,(draw:contour-polygon|draw:contour-path)?)>`

The attributes associated with the `<draw:image>` element are:

- Name (see Section 2.6.8)
- Style
- Image data
- Position, size, and transformation
- Filter name
- Layer ID (see Section 2.6.8)
- Z Index (see Section 2.6.8)

You can also use the following elements with the image element:

- Contour (see Section 2.6.7)
- Alternative Text (see Section 2.6.7)

Style

The `draw:style-name` attribute specifies the name of the style for the image.

XML Code:	`draw:style-name`
Rules:	This attribute links to a `<style:style>` element belonging to the `graphic` style family.
DTD:	`<!ATTLIST draw:text-box draw:style-name &style-name; #REQUIRED>`

Image Data

The `xlink:href` attribute links to an external file that contains the image data.

XML Code:	`xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate`
Rules:	
DTD:	`<!ATTLIST draw:image xlink:href %url; #REQUIRED>` `<!ATTLIST draw:image xlink:type (simple) #FIXED "simple">` `<!ATTLIST draw:image xlink:show (embed) "embed">` `<!ATTLIST draw:image xlink:actuate (onLoad) "onLoad">`

Position, Size, and Transformation

See Section 5.4.13 for information on using these attributes.

Filter Name

If required, the filter name can be represented as an attribute.

XML Code:	`draw:filter-name`
Rules:	This attribute contains StarOffice's internal filter name that was used to load the graphic.
DTD:	`<!ATTLIST draw:image draw:filter-name #IMPLIED>`

2.6.3 Drawings

See Section 5.4 for information on drawing shapes.

2.6.4 Controls

Every control is contained within a form and if the control is not hidden, the position of the control is represented by a frame. The frame contains a reference to the control. The frame is represented by the `<office:control>` element.

XML Code:	`<office:control>`
Rules:	This element contains a reference to the description of the control. It does not contain a description of the control itself. The description of the control is contained within a `<form:control>` element, as described in Section 2.7.2.
DTD:	`<!ELEMENT office:control (style:properties)>`

The attributes associated with the `<office:control>` element are:

- Control ID
- Control formatting properties

Control ID

The `office:control-id` attribute identifies the control to reference.

XML Code:	`office:control-id`
Rules:	The value of this attribute is the ID of the `<form:control>` element that contains the control description.
DTD:	`<!ATTLIST office:control office:control-id IDREF #REQUIRED>`

Control Formatting Properties

The control formatting properties include size, anchor type, wrap mode, and so on. The `<form:control>` element contains a properties element that contains the formatting properties for the control. See Section 2.7.2 for more information.

2.6.5 Plug-ins, Applets, and Floating Frames

Plug-ins, applets, and floating frames can be represented as objects in XML using the `<draw:object>` or `<html:object>` elements.

XML Code:	`<draw:object>` or `<html:object>`
Rules:	

DTD:	`<!ELEMENT draw:object (office:attributes?,office:desc?,`
	`office:contour?,office:param*)>`

Notes:	This element is similar to the HTML4 `<object>` element. SVG and XSL do not support plug-ins, applets, or floating frames.

The attributes associated with the `<draw:object>` or `<html:object>` elements are:

- Name (see Section 2.6.8)
- Style (see Section 2.6.8)
- Object properties
- Layer ID (see Section 2.6.8)
- Z Index (see Section 2.6.8)

You can also use the following elements with the `<draw:object>` or `<html:object>` elements:

- Object Parameters
- Contour (see Section 2.6.7)
- Alternative Text (see Section 2.6.7)

Object Properties

The object properties include size, anchor type, and so on. The object element contains an item set element that contains the object properties as attributes.

Object Parameters

The parameters element specifies the parameters for the object. *More information to be supplied.*

XML Code:	`<office:param>` or `<html:param>`
Rules:	

DTD:	`<!ELEMENT office:param EMPTY>`
	`<!ATTLIST office:param office:name CDATA #REQUIRED>`
	`<!ATTLIST office:param office:value CDATA #REQUIRED>`

Note:	This element is the same as the HTML4 `<param>` element.

2.6.6 Objects

StarOffice XML can represent some objects and it cannot represent other objects.

Representable Objects in StarOffice XML

Objects that can be represented in StarOffice XML are contained in the XML document. This applies to StarObjects and formulas in particular.

XML Code:	`<draw:foreignobject>` or `<svg:foreignobject>`
Rules:	
DTD:	`<!ELEMENT draw:foreignobject ANY>`
Notes:	This element is similar to the SVG element `<foreignobject>`. When reading a document, some applications refer to the namespace of the object to decide whether or not the object type is supported. You should register namespace names in the same way that you register class IDs and implement an SvInplaceObject for the XML representation of objects that have an unknown namespace name. The SvInplaceObject can display its content as a tree list box

Example: Object

```
<draw:foreignobject>
  <style:properties fo:width="5cm"/>
  <office:desc>This is a StarMath formula</office:desc>
  <math:math xmlns:math="...">
    <!-- some content -->
  </math:math>
</draw:foreignobject>
```

The attributes associated with the `<draw:foreignobject>` or `<svg:foreignobject>` elements are:

- Name (see Section 2.6.8)
- Style (see Section 2.6.8)
- Object properties
- Layer ID (see Section 2.6.8)
- Z Index (see Section 2.6.8)

You can also use the following element with the `<draw:foreignobject>` or `<svg:foreignobject>` elements:

- Contour (see Section 2.6.7)

Object Properties

The object properties include size, anchor type, and so on. The foreign object element contains an item set element that contains the object properties as attributes.

The representation of these properties differs from SVG.

Non-Representable Objects in StarOffice XML

Objects that cannot be represented using StarOffice XML are stored in an external file. The XML document contains a link to this file instead of the object itself. It can also contain an image that is displayed by applications that are not able to handle objects. The user can control whether or not an image is created using a user option.

The element used to represent the object link is the same as that used by applets, plugins, and floating frames. If a document contains several objects that cannot be represented in StarOffice XML, these elements are either stored in subsections of the same file or in separate files.

Note: It is not possible to include binary data in a XML document without encoding it.

Example: Link to an unrepresentable object

```
<draw:object office:classid="clsid:9999-99-9999-99"
                office:data="...">
  <style:properties fo:width="5cm"/>
  <draw:image office:href=/>
</text:object>
```

The attributes associated with non-representable object elements are:

- Internal name
- Name (see Section 2.6.8)
- Object properties
- OLE link

- Chart name

You can also use the following elements with the plugins, applets, and floating frames elements:

- Contour (see Section 2.6.7)
- Alternative Text (see Section 2.6.7)

Internal Name

Information to be supplied.

OLE Link

Information to be supplied.

Chart Name

Information to be supplied.

2.6.7 Common Frame Elements

The elements contained in this section can be used with several of the frame elements.

Contour

The `<draw:contour-polygon>` and `<draw:contour-path>` elements are used in the following elements:

- `<draw:image>`
- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`<draw:contour-polygon>`
	`<draw:contour-path>`
Rules:	These elements describe the contour of an image or object.
	See Section 5.4.4 for a description of the attributes associated with the `<draw:contour-polygon>` element. See Section for a description of the attributes associated with the `<draw:contour-path>` element.
DTD:	`<!ELEMENT draw:contour-polygon EMPTY>`
	`<!ELEMENT draw:contour-path EMPTY>`
	`<!ATTLIST draw:contour-polygon svg:width %length; #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:height %length; #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:viewbox CDATA #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:points %Points; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:width %length; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:height %length; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:viewbox CDATA #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:d %PathData; #REQUIRED>`
Note:	XSL and SVG do not support contours.

Alternative Text

The `<draw:desc>` element is used in the following elements:

- `<draw:image>`
- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`<draw:desc>`
Rules:	
DTD:	`<!ELEMENT draw:desc (#PCDATA)>`
Notes:	This element is the same as the SVG `<desc>` element. SVG does not support a description of foreign objects.

2.6.8 Common Frame Attributes

The attributes contained in this section can be used with several of the frame elements.

Name

The `office:name` attribute is used by the following elements:

- `<draw:text-box>`
- `<draw:image>`
- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`office:name`
Rules:	
DTD:	`<!ATTLIST draw:image office:name CDATA #IMPLIED>`
Note:	SVG does not support names of image or foreign objects.

Style

The `style:style` attribute is used by the following elements:

- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`style:style`
Rules:	
DTD:	`<!ATTLIST draw:object style:style CDATA #REQUIRED>`
Note:	SVG does not support styles.

Layer ID

The `office:layer-id` attribute is used by the following elements:

- `<draw:text-box>`
- `<draw:image>`
- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`office:layer-id`
Rules:	
DTD:	`<!ATTLIST draw:text-box office:layer-id %number; #IMPLIED>`
Note:	SVG does not support layers.

Z Index

The `draw:z-index` attribute is used by the following elements:

- `<draw:text-box>`
- `<draw:image>`
- `<draw:object>`
- `<draw:foreignobject>`

XML Code:	`draw:z-index`
Rules:	
DTD:	`<!ATTLIST draw:text-box draw:z-index %number; #IMPLIED>`

2.6.9 Frame Formatting Properties

The attributes and elements described in this section can be assigned to a graphic style.

Fixed and Minimum Widths

There are two types of frame widths; fixed widths and minimum widths.

XML Code:	**For fixed widths:**
	`svg:width`
	For minimum widths:
	`fo:min-width`

Rules:	The value of both types of width can be either a length or a percentage. The consequences of assigning both types of width to a frame simultaneously are undefined. If the anchor for the frame is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the frame is in a frame, the percentage value relates to the surrounding frame. In other cases, the percentage values relate to the width of the page or window.

DTD:	`<!ATTLIST style:properties svg:width %length_or_pecentage; #IMPLIED>`
	`<!ATTLIST style:properties fo:min-width CDATA #IMPLIED>`

Fixed and Minimum Heights

There are two types of frame heights; fixed heights and minimum heights.

XML Code:	**For fixed heights:**
	`svg:height`
	For minimum heights:
	`fo:min-height`

Rules:	The value of both types of height can be either a length or a percentage. The consequences of assigning both types of height to a frame simultaneously are undefined. If the anchor for the frame is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the frame is in a frame, the percentage value relates to the surrounding frame. In other cases, the percentage values relate to the height of the page or window.

DTD:	`<!ATTLIST style:properties fo:height CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:min-height CDATA #IMPLIED>`

Left and Right Margins

These properties determine the left and right margins to set around a frame. The properties are similar to those used to set the margins of a paragraph, as described in Section 3.12.19

XML Code:	`fo:margin-left` and `fo:margin-right`
Rules:	The value of these properties must be a length.

DTD:	`<!ATTLIST style:properties fo:margin-left %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:margin-right %length; #IMPLIED>`

Note:	In contrast to paragraph styles, when these properties are used for graphic styles as is the case for frames, percentage values are not supported.

Top and Bottom Margins

These properties determine the top and bottom margins to set around a frame. The properties are similar to those used to set the margins of a paragraph, as described in Section

XML Code:	`fo:margin-top` and `fo:margin-bottom`
Rules:	The value of these properties must be a length.

DTD:	`<!ATTLIST style:properties fo:margin-top %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:margin-bottom %length; #IMPLIED>`

Note:	In contrast to paragraph styles, when these properties are used for graphic styles as is the case for frames, percentage values are not supported.

Print Content

This property specifies whether or not you can print the content of a frame.

XML Code:	`style:print-content`
Rules:	
DTD:	`<!ATTLIST style:properties style:print-content %boolean; #IMPLIED>`
Note:	XSL does not support this property.

Protect

This property specifies whether the content, size, or position of a frame is protected.

XML Code:	`style:protect`
Rules:	The value of this property can be either `none` or a space separated list that consists of any of the values `content`, `position`, or `size`.
DTD:	`<!ATTLIST style:properties fo:protect CDATA #IMPLIED>`
Note:	XSL does not support this property.

Horizontal Position

The horizontal position of a frame specifies the horizontal alignment of the frame in relation to the specific area.

XML Code:	`style:horizontal-pos`
Rules:	The value of this property can be one of the following: `from-left`, `left`, `center`, `right`, `from-inside`, `inside`, or `outside`. The area that the position relates to is specified by the `style:horizontal-rel` property. The values `from-inside`, `inside` and `outside` correspond to the values `from-left`, `left`, and `right` on pages that have an odd page number and to the opposite values on pages that have an even page number. If the property value is `from-left` or `from-inside`, the `svg:x` attribute associated with the frame element specifies the horizontal position of the frame. Otherwise the `svg:x` attribute is ignored for text documents. It is also possible to use an `svg:x` attribute within a graphic style. If this is the case, then the attribute specifies a default position for new frames that are created using this style. Some values may be used in connection with certain frame anchor and relation types only.
DTD:	`<!ATTLIST style:properties style:horizontal-pos (from-left|left|center|right|from-inside|inside|outside)#IMPLIED>`

Horizontal Relation

The `style:horizontal-rel` property specifies the area to which the horizontal position of a frame relates. See the previous section for information on the `style:horizontal-pos` property.

XML Code: `style:horizontal-rel`

Rules: The value of this property can be one of the following: `page`, `page-content`, `page-start-margin`, `page-end-margin`, `frame`, `frame-content`, `frame-start-margin`, `frame-end-margin`, `paragraph`, `paragraph-content`, `paragraph-start-margin`, `paragraph-end-margin`, or `char`.

You can use some values with certain frame anchor types only.

The value `start-margin` determines the left margin, except when the horizontal position is `from-inside`, `inside` or `outside` and the anchor for the frame is on a page with an even page number, in which case it determines the right margin. The value `end-margin` determines the opposite margin to the `start-margin` values.

DTD:

```
<!ATTLIST style:properties style:horizontal-rel
          (page|page-content|page-start-margin|
           page-end-margin|frame|frame-content|
           frame-start-margin|frame-end-margin|
           paragraph|paragraph-content|paragraph-start-margin|
           paragraph-end-margin|char) #IMPLIED>
```

Vertical Position

The vertical position of a frame specifies the vertical alignment of the frame in relation to a specific area.

XML Code: `style:vertical-pos`

Rules: The value of this property can be one of the following: `from-top`, `top`, `middle`, or `bottom`. The area that the position relates to is specified by the `style:vertical-rel` property.

If the value of this property is `from-top`, the `svg:y` attribute associated with the frame element specifies the vertical position of the frame. Otherwise, the `svg:y` attribute is ignored for text documents.

It is also possible to use an `svg:y` attribute within a graphic style. If this is the case, the attribute specifies a default position for new frames that are created using this style.

Some values may be used in connection with certain frame anchor and relation types only.

DTD:

```
<!ATTLIST style:properties style:vertical-pos (from-
top|top|middle|bottom) #IMPLIED>
```

Vertical Relation

This `style:vertical-rel` property specifies the area to which the vertical position of a frame relates. See the previous section for information on the `style:vertical-pos` property.

XML Code:	`style:vertical-rel`
Rules:	The value of this property can be one of the following: `page`, `page-content`, `frame`, `frame-content`, `paragraph`, `paragraph-content`, `line`, `baseline`, or `char`. You can use some values with certain frame anchor types only.

DTD:	`<!ATTLIST style:properties style:vertical-rel`
	`          (page|page-content|frame|frame-content|`
	`           paragraph|paragraph-content|line|baseline|char)`
	`#IMPLIED>`

Frame Anchor

In text documents, every frame must have an anchor. Frame anchors are described in detail in Section 3.7.

Frame Background

The background properties for a frame are specified in the same way as the background properties for a paragraph. See Sections 3.12.25 and 3.12.26 for more information.

Border and Border Line Width

See Sections 3.12.27 and 3.12.28 for information on border and border line width properties.

Padding

See Section 3.12.29 for information on padding properties.

Shadow

See Section 3.12.30 for more information.

Columns

See Section 3.13.2 for information on frame column properties.

Editable

A text box can be editable even if the document in which it is contained is a read-only document. The editable property specifies if a text box is editable.

XML Code:	`style:editable`
Rules:	
DTD:	`<!ATTLIST style:properties style:editable %boolean; #IMPLIED>`
Note:	XSL does not support this property.
Implementation limitation:	This property is only supported by StarOffice Writer.

Wrapping

The wrapping attribute specifies how text around a frame is treated. For example, text can run around the left side of the frame, around the right side of the frame, or through the frame.

XML Code:	`style:wrap`
Rules:	
DTD:	`<!ATTLIST style:properties style:wrap (none|left|right|parallel|dynamic|run-through)`
Note:	XSL does not support the concept of wrapping. Instead, the wrap mode of a frame is determined by the anchor type. If wrapping is disabled or allowed for a single paragraph only, this can be simulated by an `fo:clear` property that is attached to an element following the frame.
Implementation limitation:	Currently, this property is only evaluated by StarOffice Writer.

Paragraph-only Wrapping

If the anchor position of a frame is a paragraph or a character, and the wrap mode specified by the `style:wrap` property is `left`, `right`, `parallel`, or `dynamic`, you can specify the number of paragraphs that wrap around the frame.

XML Code:	`style:number-wrapped-paragraphs`
Rules:	This property is only recognized by frames or styles that have a `style:wrap` property attached with a value of `left`, `right`, `parallel`, or `dynamic`. If the value is `no-limit`, there is no limit on the number of paragraphs that are allowed to wrap around a frame.
DTD:	`<!ATTLIST style:properties style:number-paragraphs-wrapped % number_or_no_limit; #IMPLIED>`
Implementation limitation:	This property is only evaluated by StarOffice Writer. Currently, if the value of this property is set to any number other than `1`, the effect on the frame is the same as if the value was set to `no-limit`.

Contour Wrapping

For some frame types you can specify that the text should wrap around the shape of the object in the frame rather than around the frame itself. This is called contour wrapping.

XML Code:	`style:wrap-contour`
Rules:	This property is only recognized by frames or styles that have a `style:wrap` property attached.
DTD:	`<!ATTLIST style:properties style:wrap-contour %boolean; #IMPLIED>`
Implementation Limitation:	This property is only evaluated by StarOffice Writer.

Contour Wrapping Mode

If the `style:wrap-contour` attribute is present, you can further specify how the text should wrap around the contour.

XML Code:	`style:wrap-contour-mode`
Rules:	This attribute is only recognized by frames or styles that already have the `style:wrap` and `style:wrap-contour` attributes attached. The value of the attribute can be `outside` or `full`. If the value of the attribute is `outside`, the text wraps around the general area to the left and right of the shape. If the value of the attribute is `full`, the text wraps around the shape and fills any possible spaces and indentations in the shape.
DTD:	`<!ATTLIST style:properties style:wrap-contour-mode (full|outside) #IMPLIED>`
Implementation Limitation:	This property is only evaluated by StarOffice Writer.

Run Through

If the value of the `style:wrap` attribute is `run-through`, you can further specify whether the content of the frame should be displayed in the background or in the foreground. This attribute is usually used for transparent objects.

XML Code:	`style:run-through`
Rules:	The value of this attribute can be `foreground` or `background`. If the value is `foreground`, the frame content is displayed in front of the text. If the value is `background`, the frame content is displayed behind the text.
DTD:	`<!ATTLIST style:attributed style:run-through (foreground|background)>`
Note:	XSL does not support this property.

Mirroring

The `style:mirror` attribute specifies whether or not an image is mirrored before it is displayed. The mirroring can be vertical or horizontal. Horizontal mirroring can be restricted to images that are only located on either odd or even pages.

XML Code:	`style:mirror`
Rules:	The value of this attribute can be `none`, `vertical`, `horizontal`, `horizontal-on-odd`, or `horizontal-on-even`. You can specify the value `vertical` and the various horizontal values together, separating them by a white space.
DTD:	`<!ATTLIST style:properties style:mirror CDATA #IMPLIED>`

Clipping

The `fo:clip` attribute specifies whether to display:

- A rectangular section of an image

or

- The entire image

XML Code:	`fo:clip`
Rules:	
DTD:	`<!ATTLIST style:properties fo:clip CDATA #IMPLIED>`
Note:	This attribute is the same in XSL.

2.6.10 Frame Events

You can assign events to a frame. The events that are attached to, for example, a text box or an image, are represented by an event element as described in Section 2.11. This element is contained within the frame type element, for example, the `<draw:text-box>` element or the `<draw:image>` element.

2.7 Forms and Controls

Forms and form controls are created by StarOffice API . Their XML representation contains the name of the StarOffice API that created the form or control and its properties.

The similarities with HTML are:

- Every control is contained in a form.
- Every control that is not hidden contains certain text or has an absolute position. This position is represented by a frame that contains a reference to the control rather than the control itself.

The differences to HTML are:

- Forms can be nested.
- Forms are not connected with the text flow and layout of a document. This does not apply to controls themselves.

2.7.1 Forms

The `<form:form>` element represents a user interface form and defines the contents and properties of the form.

XML Code:	`<form:form>`
Rules:	This element can be contained in the `<office:forms>` element of a document or in another forms element. The element contains the sub-forms and controls that are contained in the form, a `<starone:properties>` element that contains the properties of the form, and a `<script:events>` element that contains the events for the form.
DTD:	`<!ELEMENT form:form (starone:properties, script:events?, (form: form|form:control))*>`

The attribute associated with the `<form:form>` element is:

- Service name

Service Name

The `form:service-name` attribute specifies the name of the StarOffice API that created the form or control.

XML Code:	`form:service-name`
Rules:	
DTD:	`<!ATTLIST form:control form:service-name CDATA #REQUIRED>`

2.7.2 Controls

The `<form:control>` element represents a control that is contained within a form. Every item on a form is a control and is defined using the `<form:control>` element.

XML Code:	`<form:control>`
Rules:	This element can only be contained within a `<form:form>` element. The element contains a `<starone:properties>` element which contains the properties of the control, and a `<script:events>` element which contains the events of the control.
DTD:	`<!ELEMENT form:control (starone:properties, script:events?)>`

The attributes associated with the `<form:control>` element are:

- Service name (see Section 2.7.1)
- Control ID

Control ID

Controls that are not hidden are contained within a form and have a text or absolute position. This position is represented by a certain type of frame, which contains a reference to the element representing the control within the form element. Controls that are not hidden must have a `form:id` associated with them, which can be used to reference the controls.

XML Code:	`form:id`
Rules:	
DTD:	`<!ATTLIST form:control form:id CDATA #REQUIRED>`

2.7.3 Properties

The properties of a form or control are contained in a `<starone:properties>` element.

XML Code:	`<starone:properties>`
Rules:	This element contains an element for every property that is attached to a form or control. For every type class, there is a certain element that represents the properties for that class.

DTD:	`<!ENTITY % property (startone:string-prop|starone:short-prop|...)"`
	`<!ELEMENT starone:properties (%property;*)>`

2.7.4 Properties for Fundamental Type Classes

For every fundamental type class, there is an element that represents the properties of that class.

XML Code:	For string properties:
	`<starone:string-prop>`
	For short properties:
	`<starone:short-prop>`

Rules:	

DTD:	For string properties:
	`<!ELEMENT starone:string-prop (#PCDATA)>`
	For short properties:
	`<!ELEMENT starone:short-prop (#PCDATA)>`

Note: In the current and following sections, we will only show the DTD for string and short properties. The DTD for all other fundamental types uses the same syntax.

The attributes associated with the properties for fundamental type classes are:

- Property name
- Property value

Property Name

XML Code:	`starone:name`
Rules:	

DTD:	`<!ATTLIST starone:string-prop starone:name CDATA #REQUIRED>`
	`<!ATTLIST starone:short-prop starone:name CDATA #REQUIRED>`

Property Value

The value of the property value is the content of the property element.

2.7.5 Properties for Enumerated Type Classes

The representation of properties for an enumerated type is the same as for fundamental types, except that the properties reflection is specified separately.

XML Code:	For string properties:
	`<starone:string-prop>`
	For short properties:
	`<starone:short-prop>`

Rules:	
DTD:	`<!ELEMENT starone:enum-prop (#PCDATA)>`

The attributes associated with the properties for enumerated type classes are:

- Property name (see Section 2.7.4)
- Property reflection
- Property value (see Section 2.7.4)

Property Reflection

XML Code:	`starone:refl`
Rules:	
DTD:	`<!ATTLIST starone:enum-prop starone:refl CDATA #REQUIRED>`

2.7.6 Properties for Sequence Type Classes

The elements contained within the sequence element must be of the same type.

XML Code:	`<starone:seq-prop>`
Rules:	The list members are represented by elements that are very similar to the elements that represent properties, except that the `prop` in the element names is replaced by an `item` and they do not have a `starone:name` or a `starone:refl` attribute.

DTD:	`<!ENTITY % item (startone:string-item|starone:short-item|...)"`
	`<!ELEMENT starone:seq-prop (%item;+)>`
	`<!ELEMENT starone:string-item (#PCDATA)>`
	`<!ELEMENT starone:short-item (#PCDATA)>`

The attributes associated with the properties for sequence type classes are:

- Property name (see Section 2.7.4)
- Property reflection (see Section 2.7.5)
- Property value (see Section 2.7.4)

Example: Properties for sequence type classes

```
<starone:seq-prop starone:name="StringItemList" starone:refl="String">
  <starone:string-item>Item 1</starone:string-item>
  <starone:string-item>Item 2</starone:string-item>
  <starone:string-item>Item 3</starone:string-item>
  <starone:string-item>Item 4</starone:string-item>
</starone:seq-prop>
```

2.7.7 Properties for Other Type Classes

Information to be supplied.

2.7.8 Layout Forms

In HTML, forms have a connection to the text flow or layout of a document. A form starts somewhere in the document flow with a `<form>` start tag and ends with a `</form>` end tag. The content of the `<form>` element is the controls and any other document content. To be compatible with HTML and simplify conversions to HTML, an XML document can contain **layout form elements**.

These elements have the same semantics as the HTML `<form>` element, but they are created by the StarOffice software on user request only and they are never evaluated if a document is read.

XML Code:	`<office:layout-form>`
Rules:	This element has an attribute attached that contains the ID of the `<form:form>` element that contains the form description. If this element is inserted in a document, frame elements for hidden controls are also inserted.

DTD:	`<!ELEMENT office:layout-form %text-block;>`
	`<!ATTLIST office:layout-form office:form-id IDREF #REQUIRED>`

Note:	The StarOffice software inserts this element into a document on user request only. This is controlled by a certain user option.

Since the StarOffice software does not base forms on layout, the `<office:layout-form>` element only occasionally represents the content of forms correctly. The `<office:layout-form>` element presents the form content correctly when:

- The document does not contain nested forms.
- The order of the controls is the same in the control and in the document flow.
- Controls for different forms are not mixed in the document flow.
- A paragraph does not contain controls for two different forms. Controls from different forms can be located in different paragraphs provided there is no paragraph in between the paragraphs that contains controls for a different form.
- Controls of different forms are not contained in any table unless the forms are contained within a single table cell.

2.8 Hyperlinks

2.8.1 Simple Hyperlinks

A simple hyperlink is a link that locates one resource only.

XML Code:	`<office:a>`
Rules:	This element is an XLink and has some attributes with fixed values and describe the semantics of the link. The element's content is the frame that should be the source of the link.

DTD:	`<!ELEMENT office:a %frame;>`
	`<!ATTLIST office:a xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST office:a xlink:actuate (onRequest) "onRequest">`

The attributes associated with the simple hyperlink element are:

- Link location
- Link target frame
- Name
- Server side image map

Link Location

The `href` attribute specifies the target location of the link.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST office:a xlink:href %url; #REQUIRED>`

Link Target Frame

The `office:target-frame` attribute specifies the target frame of the link.

XML Code:	`office:target-frame`
Rules:	This attribute can have one of the following values: • `_self` : The referenced document replaces the content of the current frame. • `_blank` : The referenced document is displayed in a new frame. • `_parent` : The referenced document is displayed in the parent frame of the current frame. • `_top` : The referenced document is displayed in the topmost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name : The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<office:a>` element. See page 20 for a pointer to the XLink Specification. If the value of the this attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the this attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.

DTD:	`<!ATTLIST office:a office:target-frame CDATA "_blank">`
	`<!ATTLIST office:a xlink:show (new|replace) "replace">`

Name

A simple link can have a name, but it is not essential. The `office:name` attribute specifies the name of the link. The name can serve as a target for other hyperlinks.

XML Code:	`office:name`
Rules:	The name does not have to be unique.
DTD:	`<!ATTLIST office:a office:name CDATA #IMPLIED>`
Notes:	This attribute is specified for compatibility with HTML only, where an `<a>` element may serve as a link source and target simultaneously. We strongly recommend that you do not use this attribute for any purpose other than to represent links that originally came from a HTML document.

Server Side Image Map

A link can be a server side image map. The `office:server-map` attribute is used by the server to determine which link to activate within the image map. If this attribute is present, the mouse coordinates of the click position of the frame are appended to the URL of the link.

XML Code:	`office:server-map`
Rules:	
DTD:	`<!ATTLIST office:a office:server-map %boolean; "FALSE">`

2.8.2 Extended Hyperlinks

Extended hyperlinks are client side image maps. These image maps are represented by the `<office:a-map>` element.

XML Code:	`<office:a-map>`
Rules:	This element is an XLink and it contains the frame to which the image map is assigned, as well as some elements that specify the image map areas.

DTD:	`<!ELEMENT office:a-map`
	`          (office:simple-loc, (office:area-loc|area-noloc)+, %`
	`frame;)>`
	`<!ATTLIST office:a-map xlink:type (extended) #FIXED`
	`"extended">`
	`<!ATTLIST office:a-map xlink:showdefault (replace) #FIXED`
	`"replace">`
	`<!ATTLIST office:a-map xlink:actuatedefault (onRequest) #FIXED`
	`"onRequest">`

2.8.3 Area Locator

The `<office:area-loc>` element specifies an image map area that locates a specific resource, that is, has a hyperlink assigned.

XML Code:	`<office:area-loc>`
Rules:	This element is an XLink.

DTD:	`<!ELEMENT office:area-loc EMPTY>`
	`<!ATTLIST office:area-loc xlink:type (locator) #FIXED "locator">`
	`<!ATTLIST office:area-loc id ID #REQUIRED>`

The attributes associated with the `<office:area-loc>` element are:

- Area shape type
- Area shape coordinates
- Area location
- Area target frame
- Area location title

Area Shape Type

The `office:shape` attribute specifies the shape of the area locator. It corresponds to the `shape` attribute in HTML that can be used with `<area>` elements.

XML Code:	`office:shape`
Rules:	
DTD:	`<!ATTLIST office:area-loc office:shape (rect|circle|poly|default) "rect">`

Area Shape Coordinates

The `office:coords` attribute specifies the coordinates of the area shape. This attribute is very similar to the `shape` and `coords` attributes of HTML, except that the `coords` attribute in HTML contains a comma separated list of pixel values instead of a space separated list of lengths.

XML Code:	`office:coords`
Rules:	The value of this attribute is a space separated list of lengths.
DTD:	`<!ATTLIST office:area-loc office:coords CDATA #IMPLIED>`

Area Location

The link targets of the area are specified by a `href` attribute.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST office:area xlink:href CDATA #REQUIRED>`

Area Target Frame

The `office:target-frame` and `xlink:show` attributes specify the link target frame of the area. See Section 2.8.1 for more detailed information.

Area Location Title

A description of the location specified by an area is represented by a `xlink:title` attribute.

XML Code:	`xlink:title`
Rules:	
DTD:	`<!ATTLIST office:area-loc xlink:title CDATA #IMPLIED>`

2.8.4 Areas without a Location

Some image map areas do not locate a resource. The `<office:area-noloc>` element represents these image map areas. This element corresponds to the `<area>` element in HTML that has a `nohref` attribute assigned.

XML Code:	`<office:area-noloc>`
Rules:	

DTD:	`<!ELEMENT office:area-noloc EMPTY>`
	`<!ATTLIST office:area-noloc office:shape (rect|circle|poly|default) "rect">`
	`<!ATTLIST office:area-noloc office:coords CDATA #IMPLIED>`
	`<!ATTLIST office:area-noloc xlink:title CDATA #IMPLIED>`

The attributes associated with the `<office:area-noloc>` element are:

- Area shape type (see Section 2.8.3)
- Area shape coordinates (see Section 2.8.3)

2.8.5 Simple Locators

A frame with an extended hyperlink or image map can also contain a simple link. If this is the case, the target of the simple link is contained within the extended hyperlink as a `<office:simple-loc>` element.

XML Code:	`<office:simple-loc>`
Rules:	

DTD:	`<!ELEMENT office:simple-loc EMPTY>`
	`<!ATTLIST office:simple-loc xlink:type (locator) #FIXED "locator">`
	`<!ATTLIST office:simple-loc id ID #REQUIRED>`
	`<!ATTLIST office:simple-loc xlink:href CDATA #REQUIRED>`
	`<!ATTLIST office:simple-loc office:target-frame CDATA "_blank">`
	`<!ATTLIST office:simple-loc xlink:show (new|embed|replace) "replace">`
	`<!ATTLIST office:simple-loc office:server-map %boolean; "FALSE">`
	`<!ATTLIST office:simple-loc office:name CDATA #IMPLIED>`

The attributes associated with this element are the same as those associated with the `<office:a>` element. See Section 2.8.1 for more information. The associated attributes are:

- Link location
- Link target frame
- Name
- Server side image map

2.9 Number Format

The StarOffice XML number format consists of three parts:

- Prefix – the text that is displayed before the number
- Display format specification, for example, A, B, C, or 1, 2, 3
- Suffix – the text that is displayed after the number

2.9.1 Prefix and Suffix

The `style:num-prefix` and `style:num-suffix` attributes specify what to display before and after the number.

XML Code:	`style:num-prefix` and `style:num-suffix`
Rules:	If the prefix and suffix do not contain alphanumeric characters, an XSLT `format` attribute can be created from the StarOffice attributes by concatenating the values of the `style:num-prefix`, `style:num-format,` and `style:num-suffix` attributes.
DTD:	`<!ATTIST style:properties style:num-prefix CDATA #IMPLIED>` `<!ATTIST style:properties style:num-prefix CDATA #IMPLIED>`
Note:	Within the XSLT `format` attribute, the prefix and suffix can only be non-alphanumeric characters. This restriction does not apply to StarOffice software.

2.9.2 Format Specification

The `style:num-format` attribute specifies the format of the number.

XML Code:	`style:num-format`
Rules:	The value of this attribute can be "1", "a", "A", "i", or "I".
DTD:	`<!ATTLIST style:properties style:num-format CDATA #IMPLIED>`
Note:	The valid values for this attribute have the same meanings as a single alphanumeric token that appears within an XSLT `format` attribute.

2.9.3 Letter Synchronization in Number Formats

If letters are used in alphabetical order for numbering, there are two ways to process overflows within a digit, as follows:

- You can insert a new digit starting with a value of a or A, that is incremented every time an overflow occurs in the following digit. The numbering sequence in this case is something like a,b,c, ..., z, aa,ab,ac, ...,az, ba, ..., and so on.
- You can insert a new digit that always has the same value as the following digit. The numbering sequence in this case is something like a, b, c, ..., z, aa, bb, cc, ..., zz, aaa, ..., and so on. This is called **letter synchronization**.

XML Code:	`style:num-letter-sync`
Rules:	
DTD:	`<!ATTLIST style:properties style:num-letter-sync %boolean; #IMPLIED>`
Note:	XSLT does not support letter synchronization.

2.10 Scripts

Scripts do not imply a scripting language or an object model. For this reason, a script can operate on the Document Object Model (DOM) of a StarOffice XML document or on the StarOffice API.

Scripts cannot modify a document while the document is loading. However, some events are called immediately after the document is loaded.

Each script is represented by a `<script:script>` element.

XML Code:	`<script:script>`
Rules:	This element contains the source code of the script. It can also optionally contain a compiled version of the script. The compiled version of the script is contained within a `<script:byte-code>` block, which can contain any content. All scripts are contained within the scripting section of a document.
Note:	JavaScript scripts that operate on the HTML DOM are special scripts that have their own representation and they can only be used in text documents.

2.11 Event Tables

Many objects such as controls, images, text boxes, or a document itself support events. These events are represented in a generic way that is based on StarOffice API: For every event that has a script assigned, the following information is stored:

- The event name – the name of the StarOffice API add-listener method without the leading `add` and the trailing `Listener`. For example, the name of an event whose add-listener method is called `addActionListener` is `Action`.
- The script code that is assigned to the event.
- The type of script code assigned to the event, for example, JavaScript, StarBASIC, or StarScript.

The event table element specifies the table of events that are assigned to an object.

XML Code:	`<script:events>`
Rules:	This element contains `<script:event>` elements (one for every event that has script code assigned).
DTD:	`<!ELEMENT script:events (script:event)+>`

2.11.1 Event

The `<script:event>` elements are contained in the `<script:events>` element.

XML Code:	`<script:event>`
Rules:	
DTD:	`<!ELEMENT script:event (#PCDATA)>`

The attributes associated with the `<script:event>` element are:

- Event name
- Script code
- Script type
- Parameter

Event Name

The name of the event is specified by the `script:name` attribute.

XML Code:	`script:name`
Rules:	
DTD:	`<!ATTLIST script:event office:name CDATA #REQUIRED>`

Script Code

The script code is contained in the `<script:event>` element.

Script Type

The type of script code for the event is specified by the `office:type` attribute.

XML Code:	`office:type`
Rules:	
DTD:	`<!ATTLIST script:event script:type CDATA #REQUIRED>`

Parameter

Some events may require or support a parameter to their add-listener method call.

XML Code:	`script:add-listener-param`
Rules:	
DTD:	`<!ATTLIST script:event script:add-listener-param #IMPLIED>`

2.12 Change Tracking

Change tracking content and structure varies depending on the type of document you are tracking. For example, change tracking in text documents is very different from change tracking in spreadsheets. The same applies to the XML file formats of these document types. However, the integration of change tracking information follows the same design principles in both applications and one XML element is used by both document types. For more information on change tracking in a particular type of the document, see the appropriate chapter of this book.

In both text documents and spreadsheets, the default document flow of an XML document reflects the current state of the document. An XSLT stylesheet or any other application that does not acknowledge change tracking information, does for example, process all insertions into the document but does not process any deletions. Insertions are part of the default document flow, while deletions appear outside the default document flow. When an application processes a document in its current state, it does not interpret change tracking information. When an application processes the content of insertions or deletions to the document, it must interpret the change tracking information.

XSLT stylesheets are not intended to process change tracking information and the representation of change tracking information is not optimized to be processed by such scripts. Therefore, it is almost impossible for an XSLT spreadsheet to make deletions visible or insertions invisible. To do this, you need a binary application or an application that processes the DOM of a document.

2.12.1 Change Information

There is some information that all tracked changes have in common. The change information is represented by a `<office:change-info>` element and the common pieces of information are represented by attributes associated with the element.

XML Code:	`<office:change-info>`
Rules:	
DTD:	`<!ELEMENT office:change-info (#PCDATA)>`

The attributes associated with the `<office:change-info>` element are:

- The author who changed the document

- The date and time when the change took place
- A comment from the author about the change (optional)

Author, Change Date, and Change Time

XML Code:	`office:chg-author`, `office:chg-date`, and `office:chg-time`
Rules:	
DTD:	`<!ATTLIST office:change-info office:chg-author CDATA #REQUIRED office:chg-date %date; #REQUIRED office:chg-time %time; #REQUIRED>`

Comment

The comment from the author about a change is contained in the `<office:change-info>` element.

Example: Sample change information

```
<office:change-info office:chg-author="Michael Brauer"
                office:chg-date="6/18/99"
                office:chg-time="17:05:28">
  Section about sections reworked to meed requirements of page styles
</office:changed>
```

2.13 Configurations

2.13.1 Database Connections

A StarOffice XML document can contain a list of databases that are already used or can be used in the document. An `<office:database>` element represents each database connection.

XML Code:	`<office:database>`
Rules:	This element is contained in an `<office:config>` element of class "any"
DTD:	`<!ELEMENT office:database EMPTY>`

2.13.2 Job Setup

A document can contain information about the printer that was used the last time the document was printed and the print settings. The `<office:job-setup>` element contains this information

XML Code:	`<office:job-setup>`
Rules:	This element is contained in the `<office:config>` element of class "any".
DTD:	`<!ELEMENT office:job-setup EMPTY>`

CHAPTER 3

Text Content

This chapter describes the StarOffice XML representation of text content. It contains the following sections:

- Headings and Paragraphs
- Fields
- Sections
- Change Tracking
- Bulleted and Numbered Lists
- Outline Numbering
- Frames in Text Documents
- Footnotes and Endnotes
- Ruby
- Line Numbering
- Text Formatting Properties
- Paragraph Formatting Properties
- Section Formatting Properties
- Optional Information

3.1 Headings and Paragraphs

This section describes the XML elements and attributes that you use to represent heading and paragraph components in a text document.

3.1.1 Primary Heading and Paragraph Components

The XML elements that represent both headings and paragraphs are called paragraph elements.

XML Code:	For headings:
	`<text:h>`
	For paragraphs:
	`<text:p>`

Rules:	The text of the paragraph is contained in the paragraph element. It can be mixed with many other elements.

DTD:	`<!ENTITY % inline-text "text:s|text:tab-stop|text:line-break| text:span|text:a| %frames;|%fields;|text:footnote| text:ref-point|text:bookmark| text:bookmark-start|text:bookmark-end| text:change|text:change-start| text:change-end">`
	`<!ELEMENT text:p (#PCDATA|%inline-text;)*>`
	`<!ELEMENT text:h (#PCDATA|%inline-text;)*>`

If the paragraph element or any of its child elements contains white-space characters, they are **collapsed**, in other words they are processed in the same way that HTML processes them. The following Unicode characters are normalized to a SPACE character:

- HORIZONTAL TABULATION (0x0009)
- CARRIAGE RETURN (0x000D)
- LINE FEED (0x000A)
- SPACE (0x0020)

In addition, these characters are ignored if the preceding character is a white-space character. The preceding character can be contained in the same element, in the parent element, or in the preceding sibling element, as long as it is contained within the same paragraph element and the element in which it is contained processes white-space characters as described above.

White space processing takes place within the following elements:

- `<text:p>`
- `<text:h>`
- `<text:span>`
- `<text:a>`
- `<text:ref-point>`
- `<text:ref-point-start>`
- `<text:ref-point-end>`
- `<text:bookmark>`
- `<text:bookmark-start>`
- `<text:bookmark-end>`

Note: In XSL, you can enable white-space processing of a paragraph of text by attaching an `fo:white-space="collapse"` attribute to the `<fo:block>` element that corresponds to the paragraph element.

The attributes associated with heading and paragraph elements are:

- Heading level
- Style and conditional style
- Paragraph formatting properties

Heading Level

The `text:level` attribute associated with the heading element determines the level of the heading, for example, Heading 1, Heading 2, and so on.

XML Code:	`text:level`
Rules:	This attribute is associated with a `<text:h>` element.
DTD:	`<!ATTLIST text:h text:level %number; "1">`
Note:	You must apply the same style to all headings of the same level.

Style and Conditional Style

The style and conditional style attributes are optional.

XML Code:	`text:style-name` and `text:cond-style-name`
Rules:	The values of these attributes are style names, which is sufficient to identify a style because every style addressed must be a paragraph style. If a conditional style is applied to a paragraph, the `text:style-name` attribute contains the name of the style that is applied under that condition. The `text:style-name` attribute does not contain the name of the conditional style that contains the conditions and maps to other styles. The conditional style name is the value of the `text:cond-style-name` attribute.

DTD:	`<!ATTLIST text:p text:style-name %style-name; #REQUIRED>`
	`<!ATTLIST text:p text:cond-style-name %style-name; #IMPLIED>`
	`<!ATTLIST text:h text:style-name %style-name; #REQUIRED>`
	`<!ATTLIST text:h text:cond-style-name %style-name; #IMPLIED>`

Note:	This XML structure simplifies XSLT transformations because XSLT only has to acknowledge the conditional style if the formatting attributes are relevant. The referenced style can be a common style or an automatic style.

Since most documents use one paragraph style for the majority of paragraphs in the document, there are plans to develop a default style name for paragraphs.

Example: Styles and conditional styles in StarOffice XML

```
<text:p text:style-name="Heading 1">
  „Heading 1" is not a conditional style.
</text:p>

<text:p text:style-name="Numbering 1" text:cond-style-name="Text body">
  "Text body" is a conditional style. If it is contained in a numbered
  paragraph, it maps to "Numbering 1". This is assumed in this example.
</text:p>
```

Paragraph Formatting Properties

The default formatting properties that are assigned to a paragraph are represented by an automatic style. Therefore, when the document is exported, an automatic style is generated with the formatting properties of the paragraph. The parent style of the generated style is the common style that is assigned to the paragraph from the viewpoint of the StarOffice user interface.

If a paragraph has a conditional style assigned and this style is mapped to another style because of a condition, two automatic styles are generated when the document is exported. Both styles have the same formatting properties assigned, but the parent of one style is the conditional style while the parent of the other style is the style that is applied because of the condition.

Example: Paragraph formatting properties in StarOffice XML

```
<office:styles>
  <style:style name="Text Body" ...>
</office:styles>
...
<office:automatic-styles>
  <style:style name="P001" family="paragraph"
                           style:parent-style-name="Text Body">
    <style:properties fo:font-weight="bold"/>
  </style:style>
</office:automatic-styles>
...
<office:body>
  <text:p style:style-name="P001">
    This is a bold paragraph in "Text Body" style.
  </text:p>
</office:body>
```

3.1.2 White Space Characters

In general, consecutive white-space characters in a paragraph are collapsed. For this reason, there is a special XML element used to represent the Unicode character SPACE (0x0020).

XML Code:	`<text:s>`
Rules:	This element uses an attribute called `text:c` to specify the number of SPACE characters that the element represents. This element is required to represent the second and all following SPACE characters in a sequence of SPACE characters. You do not get an error if the character preceding the element is not a white-space character, but it is good practice to use this element for the second and all following SPACE characters in a sequence. This way, an application recognizes a single space character without recognizing this element.

DTD:	`<!ELEMENT text:s EMPTY>`
	`<!ELEMENT text:s text:c %number "1">`

3.1.3 Tab Stops

The `<text:tab-stop>` element represents tab stops in a heading or paragraph.

XML Code:	`<text:tab-stop>`
Rules:	
DTD:	`<!ELEMENT text:tab-stop EMPTY>`

3.1.4 Line Breaks

The `<text:line-break>` element represents a line break in a heading or paragraph.

XML Code:	`<text:line-break>`
Rules:	
DTD:	`<!ELEMENT text:line-break EMPTY>`
Note:	XSL does not support line breaks.

3.1.5 Text Styles

The `<text:span>` element represents portions of text that are formatted using a certain text style.

XML Code:	`<text:span>`
Rules:	The content of this element is the text that uses the text style. The name of the text style is the value of a `text:style-name` attribute attached to the `<text:span>` element. The `text:style-name` can be an automatic style. Since a `<text:span>` element never addresses any other type of style other than text styles, the style name is sufficient to identify the style. You can nest `<text:span>` elements. White-space characters contained in this element are collapsed.

DTD:	`<!ELEMENT text:span (#PCDATA|%inline-text;)*>`
	`<!ATTLIST text:span text:style-name %style-name; #IMPLIED>`

Notes:	The `<text:span>` attribute is similar to the HTML `<span>` attribute.

Example: Text style in StarOffice XML

```
<text:p>
  The last word of this sentence is
  <text:span text:style-name="emphasize">emphasized</text:span>.
</text:p>
```

3.1.6 Text Formatting Properties

Formatting properties that are applied to a portion of text inside a paragraph are represented by an automatic text style, which is attached to the text portion in the same way as common text styles. See Section 3.1.5 for more information. When the document is exported, an automatic text style is generated for all formatting properties that are attached to a text portion. You can assign two formatting properties to the same text portion using nested `<text:span>` elements, and the formatting properties can be represented by one or by two automatic text styles.

In most cases, automatic text styles do not have a parent style. The only situation where an automatic text style might have a parent style is when a text portion has formatting properties and a common text style assigned. The text style can be the parent style of the automatic style, but it is not essential.

Note: In StarOffice software, the text portions that have a certain formatting property applied may overlap but the `<text:span>` elements cannot overlap.

Example: Text formatting properties in StarOffice XML

This example shows the StarOffice XML code required to display the following sentence:

The rain in *Spain* stays mainly in the plain.

```
<office:automatic-styles>
  <style:style name="T001" family="text">
    <style:properties fo:font-style="italic"/>
  </style:style>
  <style:style name="T002" family="text">
    <style:properties style:text-underline="single"/>
  </style:style>
</office:automatic-styles>
...
<office:body>
  <text:p>
    The rain in
    <text:span text:style-name="T001">
      Spain
      <text:span text:style-name="T002">
        stays
      </text:span>
    </text:span>
    <text:span text:style-name="T002">
      mainly in
    </text:span>
    the plain.
  </text:p>
  ...
</office:body>
```

3.1.7 Hyperlinks

Hyperlinks in text documents are represented by a `<text:a>` element.

XML Code:	`<text:a>`
Rules:	If this element contains white-space characters, the characters are collapsed.
DTD:	`<!ELEMENT text:a (script:events?,(%inline-text;)*)>`

This element also contains an event table element, `<script:events>`, which contains the events assigned to the hyperlink. See Section 2.11 for more information on the event table element.

The attributes associated with the `<text:a>` element are:

- Name
- Link locator
- Target frame
- Text styles

Name

A hyperlink can have a name, but it is not essential. The `text:name` attribute specifies the name of the hyperlink if one exists. This name can serve as a target for some other hyperlinks.

XML Code:	`text:name`
Rules:	This name does not have to be unique.
DTD:	`<!ATTLIST text:a text:name CDATA #IMPLIED>`
Notes:	This attribute is specified for compatibility with HTML only, where an `<a>` element may serve as a link source and target simultaneously. Do not use this attribute for any purpose other than to represent links that originally came from a HTML document.

Link Locator

The URL for the link is specified by a `xlink:href` attribute.

XML Code:	`xlink:href`
Rules:	

DTD:	`<!ATTLIST text:a xlink:href %url #REQUIRED>`
	`<!ATTLIST text:a xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST text:a xlink:actuate (onRequest) "onRequest">`

Target Frame

The target frame name of the link is specified by an `office:target-frame-name` attribute.

XML Code:	`office:target-frame-name`
Rules:	This attribute can have one of the following values: • `_self` – The referenced document replaces the content of the current frame. • `_blank` – The referenced document is displayed in a new frame. • `_parent` – The referenced document is displayed in the parent frame of the current frame. • `_top` – The referenced document is displayed in the uppermost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name – The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<text:a>` element. See page 20 for a pointer to the XLink Specification. If the value of the attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.

DTD:	`<!ATTLIST text:a office:target-frame-name CDATA #REQUIRED>`
	`<!ATTLIST text:a xlink:show (new|replace) "replace">`

Text Styles

Every hyperlink has two text styles as follows:

- If the link location of the hyperlink was not visited, one text style is applied to the text of the hyperlink.
- If the link location of the hyperlink was already visited, a different text style is applied to the text of the hyperlink.

XML Code:	`text:style-name` and `text:visited-style-name`
Rules:	

DTD:	`<!ATTLIST text:a text:style-name %style-name; #IMPLIED>`
	`<!ATTLIST text:a text:visited-style-name %style-name; #IMPLIED>`

3.1.8 Footnotes

See Section 3.8.3 for information on footnotes.

3.1.9 Bookmarks

Bookmarks can either mark a text position or a text range. A text range can start at any text position and end at another text position. In particular, a bookmark can start in the middle of one paragraph and end in the middle of another paragraph. The XML element used to represent a bookmark varies depending on the type of bookmark, as follows:

- `<text:bookmark>` – to mark one text position
- `<text:bookmark-start>` – to mark the start position in a text range
- `<text:bookmark-end>` – to mark the end position in a text range

XML Code:	As above
Rules:	

DTD:	`<!ELEMENT text:bookmark EMPTY>`
	`<!ELEMENT text:bookmark-start EMPTY>`
	`<!ELEMENT text:bookmark-end EMPTY>`
	`<!ATTLIST text:bookmark text:name CDATA #REQUIRED>`
	`<!ATTLIST text:bookmark-start text:name CDATA #REQUIRED>`
	`<!ATTLIST text:bookmark-end text:name CDATA #REQUIRED>`

Example: Bookmarks in StarOffice XML

```
<text:p>
  <text:bookmark text:name="Mark 1"/>There is a text mark in front of this
  paragraph.
  <text:bookmark-start text:name="Mark 2"/>In front of this paragraph
  there is
  the start of a bookmark.
</text:p>
<text:p>
  This bookmark ends
  <text:bookmark-end text:name="Mark 2"/>
  amid this sentence.
</text:p>
```

3.1.10 Index Entries

Information to be supplied.

3.1.11 References

The StarOffice XML representation of references is modeled on the XML representation of bookmarks. There are two types of reference marks, as follows:

- A point reference
  A point reference marks a particular position in text and is represented by a single `<text:reference-mark/>` element.
- A range reference
  A range reference marks a range of characters in text and is represented by two elements; `<text:reference-mark-start/>` to mark the start of the range and `<text:reference-mark-end/>` to mark the end of the range.

Every reference is identified by its name, which must be unique. In a range reference, the start and end elements must use the same reference name.

Note: The current version of the StarOffice software does not support range references that span multiple paragraphs. If these types of range references exist, during import the StarOffice software truncates the reference to the paragraph in which the `<text:reference-mark-start/>` element appears.

Point References

Point references are defined in XML as follows:

XML Code:	`<text:reference-mark>`
Rules:	The name must not be reused for any other reference.

DTD:	`<!ELEMENT text:reference-mark EMPTY>`
	`<!ATTLIST text:reference-mark text:name %string; #REQUIRED>`

Range References

Range references are defined in XML as follows:

XML Code:	`<text:reference-mark-start>`
	`<text:reference-mark-end>`

Rules:	The name must not be reused for any other reference.

DTD:	`<!ELEMENT text:reference-mark-start EMPTY>`
	`<!ELEMENT text:reference-mark-end EMPTY>`
	`<!ATTLIST text:reference-mark-start text:name %string; #REQUIRED>`
	`<!ATTLIST text:reference-mark-end text:name %string; #REQUIRED>`

In StarOffice XML, three elements are used to represent references instead of one element because references represented as a single XML element:

- Cannot support overlapping references
- Do not interact well with other elements

Take the following example:

Example: Overlapping range references

```
<text:p>
  <text:reference-mark-start name="first"/>This is an
  <text:reference-mark-start name="second"/>example of a sentence
  <text:reference-mark-end name="first"/>with overlapping references.
  <text:reference-mark-end name="second"/>
</text:p>
```

The example paragraph shows two references that cover the following text:

reference "first"	"This is an example of a sentence"
reference "second"	"example of a sentence with overlapping references."

This overlapping structure cannot be represented using a single reference element to contain the referenced text. Similarly, a reference spanning multiple paragraphs creates the same situation as two overlapping XML elements, as does character formatting either starts or ends, but not both, within the referenced text.

3.1.12 Soft Hyphens, Hyphens, and Non-breaking Blanks

Soft hyphens, hyphens, and non-breaking blanks are represented by UNICODE characters.

The UNICODE character...	**Represents...**
SOFT HYPHEN (00AD)	soft hyphens
NON-BREAKING HYPHEN (2011)	non-breaking hyphens
NO-BREAK SPACE (00A0)	non-breaking blanks

3.2 Fields

StarOffice text documents or StarOffice text content embedded in other types of documents can contain variable text elements called fields. There are several different types of field, each of which implements a different type of variable text element. Fields are most commonly used for:

- Page numbers
  A page number field displays the number of the page it appears on. This field is useful for footers. For every page on which the footer appears, the field assumes the current page number so that all pages are numbered correctly.
- Creation dates
  A creation date field displays the date on which the current document was created. This field is useful for document templates. Every document created using the template contains the date when it was created.
- Number ranges
  A number range field allows the user to number certain elements, for example, images or tables. A number range field displays its own position in relation to the other number range fields for the same range. Therefore, if you move an image and its associated number range field within a document, the fields are automatically updated to reflect the new order.

The rest of this section describes how StarOffice software represents fields in the XML file format.

3.2.1 Common Characteristics of Field Elements

Each field type is represented by a corresponding element type. A field in a document is encoded as a single element of the appropriate type. The content of the element is the textual representation of the current field value as it is displayed in the StarOffice user interface. Therefore, ignoring all field elements and displaying only the textual content of the elements provides an approximate text-only version of the document.

The value of a field is usually stored in an attribute. It is necessary to store the value so that the presentation of the field can be recomputed if necessary, for example, if the user decides to change the formatting style of the field. It is also necessary to store the presentation style of the element content, to facilitate easy processing of the XML document. For example, if complete processing of a field is impossible or undesirable, the application can ignore the field and use only the content in this situation. For string values, if the value is identical to the presentation, the value attribute is omitted to avoid duplicate storage of information.

For fields that can store different types of content, for example, numbers, strings, or dates, a value type is stored in addition to the actual value. The value and value type attributes are explained later in Section 3.2.39. If more information is needed to restore a field, it is stored in additional attributes.

The most common attributes of field elements are:

- Fixed fields
  Many fields have a variant where the content does not change after the initial value is assigned. These fields are generally marked by the attribute `text:fixed`. See Section 3.2.39 for more information on this attribute.
- Formatting style
  Several field types, particularly those representing number, date, or time data, contain a formatting style. In StarOffice XML, this formatting style is represented by a `style:data-style-name` attribute. Since the user can change the presentation style for fields, StarOffice must be able to recompute a new representation of the field content at any time. See Section 3.2.39 for more information on this attribute.

3.2.2 Document Fields

StarOffice Writer fields can display information about the current document or about a specific part of the current document, such as the author, the current page number, or the document creation date. These fields are collectively referred to as document fields.

Document fields are often fixed. A field can be marked fixed to indicate that its content is preserved, rather than reevaluated, when the document is edited. For example, a date field shows the current date. If the date field is marked fixed, the value of the field is preserved during subsequent edits and always reflects the original date on which the field was inserted into the document. If the field is not marked fixed, its value changes whenever the document is edited. Likewise, the author field can show the original author or the last author of a document, depending on whether the field is marked fixed or not.

The group of document fields includes:

- Date and time fields
- Sender and author fields
- Page number fields
- Chapter fields
- File name fields
- Document template fields

- Statistics fields[1]

3.2.3 Date Fields

Date fields display the current date. You can adjust the date to display a date other than the current date. For example, you can change the date on a document that was edited late at night so that it displays the date of the following day or several days later.

XML Code:	`<text:date>`
Rules:	This element contains the presentation of the date field value, depending on the data style specified. The default date is the current date. You can preserve the value of this element using the `text:fixed` attribute described in Section 3.2.39.
DTD:	`<!ELEMENT text:date (#PCDATA)>`

The attributes that you can associate with the `<text:date>` element are:

- Date value
- Date adjustment
- Fixed (see Section 3.2.39)
- Formatting style (see Section 3.2.39). The formatting style must be a date data style, see Section 2.5.4 for more information.

Date Value

The `text:date-value` attribute specifies a particular date value. For example, if the date field is marked fixed, you can use this attribute to specify the date on which the field was marked as fixed. You can also use this attribute to specify a future date.

XML Code:	`text:date-value`
Rules:	The date value must conform with the extended date format described in Section 5.2.1.1 of ISO 8601. See page 20 for a pointer to ISO 8601. If no value is specified, the current date is assumed, even if the field is marked fixed.
DTD:	`<!ATTLIST text:date text:date-value %date; #IMPLIED>`

Date Adjustment

You can adjust the value of a date field by a certain time period, which you specify using the `text:date-adjust` attribute. StarOffice Writer truncates the specified time period to a period of full days and adds it to the value of the date field. If the time period is negative, StarOffice Writer subtracts it from the value of the date field yielding a date before the current date.

XML Code:	`text:date-adjust`
Rules:	The value of this attribute must conform to the time period format described in Section 5.5.3.2 of ISO 8601. See page 20 for a pointer to ISO 8601. The value can be preceded by an optional minus sign to indicate a negative time duration.

1 These fields are not currently part of the StarOffice XML file format.

DTD:	`<!ATTLIST text:date text:date-adjust %c-duration; #IMPLIED>`
Implementation limitation:	Currently, StarOffice does not support month or year durations.

Time Fields

Time fields display the current time. They are very similar to the date fields described in the previous section, supporting the same attributes except that for time fields, they are called `text:time-value` and `text:time-adjust` attributes.

XML Code:	`<text:time>`
Rules:	This element contains the presentation of the time field value, depending on the data style specified. The default time is the current time. You can preserve the value of this element using the `text:fixed` attribute described in Section 3.2.39.
DTD:	`<!ELEMENT text:time (#PCDATA)>`

The attributes that you can associate with the `<text:time>` element are:

- Time value
- Time adjustment
- Fixed (see Section 3.2.39)
- Formatting style (see Section 3.2.39). The formatting style must be a time data style, see Section 2.5.5 for more information.

Time Value

The `text:time-value` attribute records the time at which the document was last edited.

XML Code:	`text:time-value`
Rules:	The value of this attribute must conform with the extended time format described in Section 5.4.1 of ISO 8601. See page 20 for a pointer to ISO 8601. If no value is specified, the current time is assumed, even if the field is marked `fixed`.
DTD:	`<!ATTLIST text:time text:time-value %time; #IMPLIED>`

Time Adjustment

You can adjust the value of a time field by a certain time period, which you specify using the `text:time-adjust` attribute. The StarOffice software truncates the time period to a period of full minutes and adds it to thevalue of the time field. If the time period is negative, the StarOffice software subtracts it from the value of the time field yielding a time in the past.

XML Code:	`text:time-adjust`
Rules:	The value of this attribute must conform to the time period format described in Section 5.5.3.2 of ISO 8601. See page 20 for a pointer to ISO 8601. The value can be preceded by an optional minus sign to indicate a negative time duration. Positive values adjust the time to a time in the future, while negative values adjust the time to a time in the past. The duration is truncated to full minutes.

DTD:	`<!ATTLIST text:time text:time-adjust %c-duration; #IMPLIED>`

Example: Time adjust attributes and their effects

If the attribute `text:time-adjust="PTM15"`, the time field displays a time which is 15 minutes later than the actual time specified by the time field value.

If the attribute `text:time-adjust="-PTH1"`, the time field displays a time which is one hour before the actual time specified by the time field value.

3.2.4 Page Numbers

Page number fields display the current page number. These fields are particularly useful for recurring content, such as headers and footers. If you insert a page number field into a footer, the current page number is displayed on every page on which the footer appears.

XML Code:	`<text:page-number>`
Rules:	If the `text:page-adjust` attribute is used, the page number of the page at the specified offset is displayed, if it exists.
DTD:	`<!ELEMENT text:page-number (#PCDATA)>`

The attributes that you can associate with the `<text:page-number>` element are:

- Page adjustment
- Display previous or following page numbers
- Fixed (see Section 3.2.39)
- Formatting style (see Section 3.2.39)
  Page numbers can be formatted according to the number format described in Section 2.9. If a number style is not specified, the page numbers are formatted according to the number style defined in the current page style.

Page Adjustment

You can adjust the value of a page number field by a specified number, which allows you to display the page numbers of following or preceding pages. You specify the adjustment number using the `text:page-adjust` attribute. When you use this attribute, the application:

1. Adds the value of the attribute to the current page number.
2. Checks to see if the resulting page exists.
3. If the page exists, the number of that page is displayed.
4. If the page does not exist, the value of the page number field remains empty and no number is displayed.

XML Code:	`text:page-adjust`
Rules:	
DTD:	`<!ATTLIST text:page-number text:page-adjust %integer; #IMPLIED>`

Display Previous or Following Page Numbers

The `text:select-page` attribute allows you to display the number of the previous or the following page rather than the number of the current page.

XML Code:	`text:select-page`
Rules:	The value of this attribute can be `previous`, `current`, or `next`.
DTD:	`<!ATTLIST text:page-number text:select-page ( previous | current | next ) "current">`

Note: To display the current page number on all pages except the first or last page, you should use a combination of the `text:select-page` and `text:page-adjust` attributes.

Example: Displaying the current page number on all pages except the first page

```
<text:page-number text:select-page="previous" text:page-adjust="1" text:
num-format="1"/>
```

3.2.5 Page Continuation Text

In some publications, a continuation reminder is printed at the bottom of the page in addition to the page number. To include a continuation reminder, use the `<text:page-continuation>` element.

XML Code:	`<text:page-continuation>`
Rules:	
DTD:	`<!ELEMENT text:page-continuation (#PCDATA)>`

The attributes associated with the `<text:page-continuation>` element are:

- Previous or following page
- String value

Previous or Following Page

This attribute specifies whether to check for a previous or next page and if the page exists, the continuation text is printed.

XML Code:	`text:select-page`
Rules:	The value of this attribute can be `previous` or `next`.
DTD:	`<!ATTLIST text:page-continuation text:select-page ( previous | next ) #REQUIRED>`

String Value

This attribute specifies the continuation text to display.

XML Code:	`text:string-value`
Rules:	If this attribute is omitted, the element content is used.

DTD:	`<!ATTLIST text:page-continuation text:string-value CDATA #IMPLIED>`

3.2.6 Sender Fields

There are several fields which contain information about the sender of the current document, for example, name and email address. The information about the sender is taken from the StarOffice user information dialog. If a sender field is marked fixed using the `text:fixed` attribute, the original sender information in the sender fields is preserved. Otherwise, the information is updated each time the file is edited, causing the fields to change value when the document is edited by a different user.

First Name

This element represents the first name of the sender.

XML Code:	`<text:sender-firstname>`
Rules:	
DTD:	`<!ELEMENT text:sender-firstname (#PCDATA)>` `<!ATTLIST text:sender-firstname text:fixed %boolean "true">`

Last Name

This element represents the last name of the sender.

XML Code:	`<text:sender-lastname>`
Rules:	
DTD:	`<!ELEMENT text:sender-lastname (#PCDATA)>` `<!ATTLIST text:sender-lastname text:fixed %boolean "true">`

Initials

This element represents the initials of the sender.

XML Code:	`<text:sender-initials>`
Rules:	
DTD:	`<!ELEMENT text:sender-initials (#PCDATA)>` `<!ATTLIST text:sender-initials text:fixed %boolean "true">`

Title

This element represents the title of the sender.

XML Code:	`<text:sender-title>`
Rules:	

DTD:	`<!ELEMENT text:sender-title (#PCDATA)>`
	`<!ATTLIST text:sender-title text:fixed %boolean; "true">`

Position

This element represents the position of the sender.

XML Code:	`<text:sender-position>`
Rules:	
DTD:	`<!ELEMENT text:sender-position (#PCDATA)>`
	`<!ATTLIST text:sender-position text:fixed "true">`

Email Address

This element represents the email address of the sender.

XML Code:	`<text:sender-email>`
Rules:	
DTD:	`<!ELEMENT text:sender-email (#PCDATA)>`
	`<!ATTLIST text:sender-email text:fixed "true">`

Private Telephone Number

This element represents the private telephone number of the sender.

XML Code:	`<text:sender-phone-private>`
Rules:	
DTD:	`<!ELEMENT text:sender-phone-private (#PCDATA)>`
	`<!ATTLIST text:sender-phone-private text:fixed "true">`

Fax Number

This element represents the facsimile number of the sender.

XML Code:	`<text:sender-fax>`
Rules:	
DTD:	`<!ELEMENT text:sender-fax (#PCDATA)>`
	`<!ATTLIST text:sender-fax text:fixed "true">`

Company Name

This element represents the name of the company that employs the sender.

XML Code:	`<text:sender-company>`
Rules:	

DTD:	`<!ELEMENT text:sender-company (#PCDATA)>`
	`<!ATTLIST text:sender-company text:fixed "true">`

Office Telephone Number

This element represents the office telephone number of the sender.

XML Code:	`<text:sender-phone-work>`
Rules:	

DTD:	`<!ELEMENT text:sender-phone-work (#PCDATA)>`
	`<!ATTLIST text:sender-phone-work text:fixed "true">`

Street

This element represents the street name of the address of the sender.

XML Code:	`<text:sender-street>`
Rules:	

DTD:	`<!ELEMENT text:sender-street (#PCDATA)>`
	`<!ATTLIST text:sender-street text:fixed "true">`

City

This element represents the city name of the address of the sender.

XML Code:	`<text:sender-city>`
Rules:	

DTD:	`<!ELEMENT text:sender-city (#PCDATA)>`
	`<!ATTLIST text:sender-city text:fixed "true">`

Postal Code

This element represents the postal code of the address of the sender.

XML Code:	`<text:sender-postal-code>`
Rules:	

DTD:	`<!ELEMENT text:sender-postal-code (#PCDATA)>`
	`<!ATTLIST text:sender-postal-code text:fixed "true">`

Country

This element represents the country of the address of the sender.

XML Code:	`<text:sender-country>`
Rules:	

DTD:	`<!ELEMENT text:sender-country (#PCDATA)>`
	`<!ATTLIST text:sender-country text:fixed "true">`

State or Province

This element represents the state or province of the address of the sender, if applicable.

XML Code:	`<text:sender-state-or-province>`
Rules:	

DTD:	`<!ELEMENT text:sender-state-or-province (#PCDATA)>`
	`<!ATTLIST text:sender-state-or-province text:fixed true">`

3.2.7 Author Fields

There are two StarOffice fields that represent the author of a document. One displays the full name of the author and the other displays the initials of the author. Author fields can be fixed using the `text:fixed` attribute. Marking an author field as fixed preserves the original field content. Otherwise, the field content changes each time the document is updated, to reflect the last author of the document.

Name of the Author

This element represents the full name of the author.

XML Code:	`<text:author-name>`
Rules:	

DTD:	`<!ELEMENT text:author-name (#PCDATA)>`
	`<!ATTLIST text:author-name text:fixed "true">`

Initials of the Author

This element represents the initials of the author.

XML Code:	`<text:author-initials>`
Rules:	

DTD:	`<!ELEMENT text:author-initials (#PCDATA)>`
	`<!ATTLIST text:author-initials text:fixed "true">`

3.2.8 Placeholders

StarOffice Writer uses placeholder fields to indicate locations in a document where the user must fill in some information. For example in a letter template, a section of the document is reserved for the address of the recipient. A placeholder field displays text informing the user about the purpose of the placeholder and sometimes includes a description. Placeholder fields can represent different text elements, such as text or tables.

XML Code:	`<text:placeholder>`

Rules:	This element contains some brief text which is displayed with the placeholder.
DTD:	`<!ELEMENT text:placeholder (#PCDATA)>`

The attributes that you can associate with the `<text:placeholder>` element are:

- Placeholder type
- Placeholder description

Placeholder Type

There are five different types of placeholder, representing the five possible types of content: text, tables, text boxes, images, or objects. The placeholder type attribute represents the content type.

XML Code:	`text:placeholder-type`
Rules:	This attribute is mandatory and it indicates which type of text content the placeholder represents. The value of the attribute can be `text`, `text-box`, `image`, `table`, or `object`.
DTD:	`<!ATTLIST text:placeholder text:placeholder-type ( text | table | text-box | image | object ) #REQUIRED>`

Placeholder Description

In addition to the brief text stored in the element content, you can associate a `text:description` attribute with the placeholder element. This attribute is optional. The purpose of the attribute is to contain a more elaborate description of the purpose of the placeholder than the description stored in the element content. See Section 3.2.39 for information on using the `text:description` attribute.

DTD: `<!ATTLIST text:placeholder text:description %string; #IMPLIED>`

3.2.9 Variable Fields

StarOffice Writer documents can contain variables, which are processed or displayed using variable fields. A variable is a name/value pair. The variable name is used throughout the document to identify a particular variable, and therefore variable names cannot be reused for different types of variables. Most variable fields support different value types, such as numbers, dates, strings, and so on. In the StarOffice XML file format, a variable must be declared at the beginning of a document.

There are three types of variables in StarOffice Writer:

- **Simple variables**

  Simple variables, usually called variables, can take different values at different positions throughout a document. They are set using either setter or input fields. Setter fields contain an expression, which is used to compute the new value of the variable. Input fields prompt the user for the new value. Simple variables can be used to display different text in recurring elements, such as headers or footers.

- **User variables**

  User variables have the same value throughout a document. If a user variable is set anywhere within the document, all fields in the document that display the user variable have the same value. In the StarOffice user interface, a user variable can be set at any occurrence of a user field, or using user variable input fields. In the StarOffice XML file format, the value of the user variable can only be set when the variable is declared.

- **Sequence variables**

  Sequence variables are used to number certain items in a StarOffice Writer document, for example, images or tables.

Expression and text input fields are also variable fields, but they are not associated with any particular variables. Since their functionality is closely related to that of the variable fields, they are also described in this section of the manual.

XML Code:	`%variable-fields;`
Rules:	
DTD:	`<!ENTITY % variable-fields "text:variable-set | text:variable-get | text:variable-input | text:user-field-get | text:user-field-input | text:sequence | text:expression | text:text-input" >`

You must declare variables before you can use them. The variable declarations are collected in container elements for the particular variable type. The StarOffice XML code for declaring variables is described in the following table.

XML Code:	`%variable-declarations;`
Rules:	
DTD:	`<!ELEMENT text:variable-decls "text:variable-decl*">` `<!ELEMENT text:user-field-decls "text:user-field-decl*">` `<!ELEMENT text:sequence-decls "text:sequence-decl*">` `<!ENTITY % variable-declarations "text:variable-decl?, text:user-field-decl?, text:sequence-decl?>`

3.2.10 Declaring Simple Variables

You declare simple variables using `<text:variable-decl>` elements. The declaration specifies the name and the value type of the variable.

XML Code:	`<text:variable-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:variable-decl EMPTY>`

To specify the name and value type of the simple variable, you attach the following attributes to the `<text:variable-decl>` element:

➢ `text:name`

  The name of the variable must be unique. The name cannot already be used for any other type of variable. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-decl text:name %variable-name; #REQUIRED>`

- `text:value-type`

  See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-decl %value-type-attlist;>`

3.2.11 Setting Simple Variables

You can set simple variables using variable setter elements.

XML Code:	`<text:variable-set>`
Rules:	This element contains the presentation of the value of the variable, which can be empty if the `text:display` attribute is set to `none`.
DTD:	`<!ELEMENT text:variable-set (#PCDATA)>`

The attributes that you can attach to the `<text:variable-set>` element are:

- `text:name`

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set text:name %variable-name;>`

- `text:formula`

  This attribute contains the formula to compute the value of the variable field. If the formula equals the content of the field element, you can omit this attribute. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set text:formula %formula;>`

- `text:value-type` and the appropriate value attribute

  See Section 3.2.39 for information on using these attributes.

  Note: A simple variable should not contain different value types at different places in a document. However, the current StarOffice software implementation allows the use of different value types for different instances of the same variable. In the case of the numeric value types `float`, `percentage,` and `currency`, the value is automatically converted to the different value type. For value types that are stored internally as numbers, such as `date`, `time,` and `boolean` types, the values are reinterpreted as numbers of the respective types. If a variable is used for both string and non-string types, the behavior is undefined, therefore this practice is not recommended.

DTD:	`<!ATTLIST text:variable-set %value-attlist;>`

- `text:display`

  You can use this attribute to specify whether or not to display the value of the `<text:variable-set>` element. If the `text:display` attribute is set to `value`, the value of the variable is displayed. If the attribute is set to `none`, the value is not displayed. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set text:display ( value | none ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set style:data-style-name %style-name; #IMPLIED>`

3.2.12 Displaying Simple Variables

The `<text:variable-get>` element reads and displays the value of a simple variable.

XML Code:	`<text:variable-get>`
Rules:	The value of this element is the value of the last preceding `<text:variable-set>` element with an identical `text:name` attribute. The element determines how the value of the variable is presented, in accordance with the chosen formatting style.
DTD:	`<!ELEMENT text:variable-get (#PCDATA)>`

The attributes that you can attach to the `<text:variable-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:variable-decl>` element. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get text:name %variable-name; #REQUIRED>`

- `text:display`

  You can use this attribute to specify whether to display the formula for a simple variable or the computed value of the variable. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get text:display ( value | formula ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get style:data-style-name %style-name; #IMPLIED>`

3.2.13 Variable Input Fields

As an alternative to setting simple variables using formulas in variable setter fields, the user can be prompted for variable values. To do this, you use the `<text:variable-input>` element.

XML Code:	`<text:variable-input>`
Rules:	This element contains the presentation of the variable's value according to the chosen formatting style.
Note:	The presentation can be empty if the `text:display` attribute is set to `none`.
DTD:	`<!ELEMENT text:variable-input (#PCDATA)>`

The attributes that you can attach to the `<text:variable-input>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. It must match the name of a variable that was already declared. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:name %variable-name; #REQUIRED>`

- `text:description`

  This optional attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the variable or the use of the value within the document to enable them to choose an appropriate value. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:description %string: #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.2.39 for information on using these attributes.

DTD:	`<!ATTLIST text:variable-input %value-attlist;>`

- `text:display`

  You can use this attribute to specify whether to display or hide the value of the variable through the variable input field. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:display ( value | none ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input style:data-style-name %style-name; #IMPLIED>`

3.2.14 Declaring User Variables

User variables contain values that are displayed using appropriate fields. Unlike simple variables, user variables have the same value throughout a document because the value of the user variable is specified in the variable declaration.

XML Code:	`<text:user-field-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:user-field-decl EMPTY>`

The attributes that you can associate with the `<text:user-field-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and sequence variables. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-decl text:name %variable-name; #REQUIRED>`

- `text:formula`

  This attribute contains the formula to compute the value of the user variable field. If the formula equals the content of the field element, you can omit this attribute. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-decl text:formula %formula; #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.2.39 for information on using these attributes.

DTD:	`<!ATTLIST text:user-field-decl %value-attlist;>`

3.2.15 Displaying User Variables

You can display the content of user variables using `<text:user-field-get>` elements.

XML Code:	`<text:user-field-get>`
Rules:	
DTD:	`<!ELEMENT text:user-field-get (#PCDATA)>`

The attributes that you can attach to the `<text:user-field-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:user-field-decl>` element. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-get text:name %variable-name; #REQUIRED>`

- `text:display`

  You can use this attribute to specify whether to:

  - Display the formula used to compute the value of the user variable.
  - Display the value of the user variable.
  - Hide the user variable fields.

  See Section 3.2.39 for information on using this attribute.

  Note: Since the StarOffice ™ Writer user interface allows users to edit a user field variable by clicking on any user field, a hidden `<text:user-field-get>` element can be used as an anchor to allow easy access to a particular user field variable.

DTD:	`<!ATTLIST text:user-field-get text:display ( value | formula | none ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-get style:data-style-name %style-name;>`

3.2.16 User Variable Input Fields

An alternative way of setting user variables is to use input fields, similar to the input fields for simple variables. You can set a user variable in this way using the `<text:user-field-input>` element. Since the value of a user field variable is stored in the `<text:user-field-decl>` element, the `<text:user-field-input>` element does not contain the value and value type attributes from the `<text:variable-input>` field.

XML Code:	`<text:user-field-input>`
Rules:	This element determines how the value of the user variable is displayed, in accordance with the chosen formatting style.
Note:	The presentation can be empty if the `text:display` attribute is set to `none`.
DTD:	`<!ELEMENT text:user-field-input (#PCDATA)>`

The attributes that you can attach to the `<text:user-field-input>` element are:

- `text:name`

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-input text:name %variable-name; #REQUIRED>`

- `text:description`

  This optional attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the variable or the use of the value within the document, to enable them to choose an appropriate value. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-input text:description %string; #IMPLIED>`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-input style:data-style-name % style-name; #IMPLIED>`

3.2.17 Declaring Sequence Variables

Sequence variables are used to number items within a StarOffice Writer document. The most common use of sequence variables is for sequential numbering. However, you can include expression formulas in sequence fields to support more advanced sequences. See Section 3.2.18 for more information on sequence fields and their uses.

You declare sequence variables using the `<text:sequence-decl>` element.

XML Code:	`<text:sequence-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:sequence-decl EMPTY>`

To facilitate chapter-specific numbering, you can attach attributes for the chapter level and a separation character to a sequence variable. The attributes that you can attach to the `<text:sequence-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and user variables. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence-decl text:name %variable-name; #REQUIRED>`

- `text:display-outline-level`

  See the section *Outline Level* for information about this attribute.

- `text:separation-character`

  See the section *Separation Character* for information about this attribute.

Outline Level

You can number sequences by chapter. To use this feature, use the `text:display-outline-level` attribute to specify an outline level that determines which chapters to reference for the chapter-specific numbering. All chapters that are at or below the specified outline level reset the value of the sequence to zero, the default value. Also, the chapter number of the last chapter at or below the specified outline level is prepended to the sequence number. Choosing an outline level of zero results in a straight sequence of all sequence elements for that sequence variable.

XML Code:	`text:display-outline-level`
Rules:	The value of this attribute must be an integer greater than or equal to zero. StarOffice currently supports ten outline levels.
DTD:	`<!ATTLIST text:sequence-decl text:display-outline-level %integer; "0">`

Separation Character

If you number sequences by chapter, use this attribute to choose a character to separate the chapter number from the sequence number.

XML Code:	`text:separation-character`
Rules:	If the value of the `text:display-outline-level` attribute is a non-zero value, you must specify a separation character. Otherwise, if the value of `text:display-outline-level` is zero, you must omit this attribute.
DTD:	`<!ATTLIST text:sequence-decl text:separation-character %character; ".">`

Example: Sequence variable

The sequence variable 3.2.17#5 with a value of `5` is declared using:

Attribute	**Value**
`text:display-outline-level`	`3`
`text:separation-character`	`#`

3.2.18 Sequence Fields

Once a sequence variable is declared, you can use it in sequence fields throughout the document. Most sequence fields simply increment and display the sequence variable. However, sequence fields can also assume a new start value at any given position in a document. This start value is computed using a formula which is contained in the sequence field. If a sequence field without a start value is added to the StarOffice user interface, the StarOffice software automatically inserts an expression of the type `variable+1`.

Sequence fields are most commonly used for simple counting sequences. However, the ability to provide arbitrary expressions supports more complex sequences. To form a sequence of even numbers, all sequence elements for that particular variable need to contain a formula incrementing the value by two, for example, `variable+2`. A sequence with a starting value of `1` and all subsequent elements using the formula `variable*2` yields all powers of two. Since different sequence elements for the same sequence variable may contain different formulas, complex sequences may be constructed.

XML Code:	`<text:sequence>`
Rules:	
DTD:	`<!ELEMENT text:sequence (#PCDATA)>`

The attributes that you can attach to the `<text:sequence>` element are:

- `text:name`

  This attribute specifies the name of the variable that the field is to display. It must match the name of a sequence variable that was already declared. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence text:name %variable-name; #REQUIRED>`

- `text:formula`

  This optional attribute contains a formula to compute the value of the sequence field. If this attribute is omitted, an expression containing the content of the element is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence text:formula %formula; #IMPLIED>`

- `style:num-format` and `style:num-letter-sync`

  These attributes specify the numbering style to use. If a numbering style is not specified, the numbering style is inherited from the page style. See Section 3.2.39 for information on these attributes.

DTD:	`<!ATTLIST text:page-number %num-format;>`

- `text:ref-name`

See the following section *Reference Name* for more information about this attribute.

Reference Name

Sequence fields can be the target of references, as implemented using reference fields. See Section 3.2.38 for more information about reference fields. To enable a reference field to identify a particular sequence field, the sequence field must contain an additional attribute containing a name. No two sequence fields can have the same reference name.

XML Code:	`text:ref-name`
Rules:	If the sequence field is not the target of a reference, this attribute can be omitted.
DTD:	`<!ATTLIST text:ref-name %string; #IMPLIED>`

3.2.19 Expression Fields

Expression fields contain expressions that are evaluated and the resulting value is displayed. The value of the expression is formatted according to the chosen formatting style.

XML Code:	`<text:expression>`
Rules:	
DTD:	`<!ELEMENT text:expression (#PCDATA)>`

The attributes that you can attach to the `<text:expression>` element are:

- `text:formula`

  This attribute contains the actual expression used to compute the value of the expression field. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:expression text:formula %formula; #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.2.39 for information on using these attributes.

DTD:	`<!ATTLIST text:expression %value-type;>`

- `text:display`

  Use this attribute to specify one of the following:

  - To display the value of the field.
  - To displat the formula used to compute the value.

  See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:expression text:display ( value | formula ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:expression style:data-style-name %style-name; #IMPLIED>`

3.2.20 Text Input Fields

A text input field is a variable field. From the point of view of the StarOffice user interface, a text input field is similar to the <`text:variable-input`> and <`text:user-field-input`> fields. However, the text input field does not change the value of any variables.

XML Code:	`<text:text-input>`
Rules:	
DTD:	`<!ELEMENT text:text-input (#PCDATA)>`

The attribute that you can attach to the `<text:text-input>` element is:

- `text:description`

  This attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the purpose of the field and how it is used within the document, to enable them to choose an appropriate value. See Section 3.2.39 for information on using this attribute.

DTD:	`<!ATTLIST text:text-input text:description %string; #IMPLIED>`

3.2.21 Database Fields

StarOffice Writer documents can connect to StarOffice Base databases and display database information as text content. To display database information, StarOffice Writer uses a group of text fields, collectively called database fields. StarOffice Base can use database tables from SQL servers, therefore you can use database fields to access any SQL database, provided that the appropriate drivers are available.

In StarOffice Base, a database contains:

- Tables which store the actual data
- Queries which extract a subset of data from one or more tables
- Forms which present the data
- Reports which summarize the database content

Database forms and reports are not relevant to XML text content, therefore they are not discussed in this chapter. From the point of view of embedding database information in StarOffice text documents, queries and tables are considered the same. Therefore for the remainder of this section, the phrase **database table** refers to both database tables and database queries.

Every database in StarOffice Base has a name and this name is used by all of the StarOffice components to identify a database. All database fields contain a database name and most database fields also contain the name of a database table, which must be stored in the named database.

The following entity is defined for database fields:

XML Code:	`%database-table;`
DTD:	`<!ENTITY % database-table "text:database-name CDATA #REQUIRED text:table-name CDATA #REQUIRED" >`
Example:	`<!ATTLIST database-element %database-table;>`

Database fields alone do not retrieve information from a database. In addition to the database fields, a set of database rows is also added to the document. When new data is added to the document, all database fields belonging to the added database table are updated. Using the StarOffice user interface, you can add database rows in one of the following ways:

- Manually using the Beamer and the "Data to Fields" function.
- Using the Form Letter option on the File menu. This option mixes each row in the chosen data set into a newly created copy of the form letter.

To display data from a database table use the `<text:database-display>` field. With the `<text:database-select>` and `<text:database-next>` elements, you can determine which row within the current mix-in selection to display. The current row number for a particular table may be displayed using the `<text:database-row-number>` field. Finally, the `<text:database-name>` field displays the name of the most recently used database, which is the address book file database by default.

3.2.22 Displaying Database Content

The database display element displays data from a database. When a new data set is mixed into a document, all fields that display data from that database table update their content.

XML Code:	`<text:database-display>`
Rules:	
DTD:	`<!ELEMENT text:database-display (#PCDATA)>`

The attributes that you can attach to the `<text:database-display>` element are:

- `text:database-name` and `text:table-name`

  These attributes specify the database and database table that this field uses.

DTD:	`<!ATTLIST text:database-display %database-table;>`

- `text:database-column-name`

  See the section *Column Name* for information about this attribute.

- `text:value-type` and the appropriate value attribute

  See Section 3.2.39 for information on using these attributes.

DTD:	`<!ATTLIST text:database-display %value-attlist;>`

- `style:data-style-name`

  If the column specifies a numeric, boolean, date, or time value, the data is formatted according to the appropriate data style. If no data style is specified, the data style assigned to this column in StarOffice Base is used. See Section 3.2.39 for more information about using this attribute.

DTD:	`<!ATTLIST text:database-display style:data-style-name % style-name;>`

Column Name

This attribute specifies the column from which the data is to be displayed.

XML Code:	`text:column-name`
Rules:	The value of this attribute must be a column contained in the specified database.
DTD:	`<!ATTLIST text:database-display text:column-name CDATA #REQUIRED>`

3.2.23 Selecting the Next Database Row

The database next element changes which row from the current mix-in selection is used for display in all following `<text:database-display>` fields. The next row from the current mix-in selection is chosen if it satisfies a given condition. If the next row is wanted regardless of any condition, the condition may be omitted or set to `true`.

XML Code:	`<text:database-next>`
Rules:	
DTD:	`<!ELEMENT text:database-next EMPTY>`

The attributes that you can attach to the `<text:database-next>` are:

- `text:database-name` and `text:table-name`

  These attributes specify the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-next %database-table;>`

- `text:condition`

  See the section *Condition* for information about this attribute.

Condition

The `text:condition` attribute specifies the condition expression. The expression is evaluated and if the result interpreted as a boolean value is true, the next row is used as the new current row. Please note that you can use database field values in the expression by enclosing in square brackets, the database name, the table name, and the column name, separated by dots.

If the `text:condition` attribute is not present, StarOffice assumes the formula `true`, meaning that the next row is selected unconditionally.

XML Code:	`text:condition`
Rules:	
DTD:	`<!ATTLIST text:database-next text:condition %formula; #IMPLIED>`

Example:

```
text:formula='[address book file.address.FIRSTNAME] == "Julie"'
```

This example specifies a condition that is true if the current row from the StarOffice address book is the address for a person named Julie. If the condition shown in this example is used in a `<text:database-next>` element, the following happens:

- The `<text:database-display>` elements display the data from the first row of the current mix-in selection.

- If the `FIRSTNAME` column of the current row reads `Julie`, the current row is changed. Otherwise, nothing happens.
- If the first row is `Julie`, the following `<text:database-display>` elements display data from the second row. Otherwise, they display data from the first row.

See Section 3.2.39 for more information on the formula syntax of a `text:condition` attribute, which is the same as that of the `text:formula` attribute.

3.2.24 Selecting a Row Number

The `<text:database-row-select>` element selects a given row from the current mix-in selection. As with the `<text:database-row-next>` element, you can specify a condition so that the given row is only selected only if the condition is true.

XML Code:	`<text:database-row-select>`
Rules:	
DTD:	`<!ELEMENT text:database-row-select EMPTY>`

The attributes that you can attach to the `<text:database-row-select>` are:

- `text:database-name` and `text:table-name`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-row-select %database-table;>`

- `text:condition`

  This attribute specifies the condition expression. See Section 3.2.23 for a full explanation of how to use this attribute.

DTD:	`<!ATTLIST text:database-row-select text:condition %formula; #IMPLIED>`

- `text:row-number`

  See the section *Selecting the Row Number* for information about this attribute.

Selecting the Row Number

This attribute specifies the row number to select when a condition is true.

XML Code:	`text:row-number`
Rules:	
DTD:	`<!ATTLIST text:database-row-select text:row-number % integer; #REQUIRED>`

3.2.25 Displaying the Row Number

The `<text:database-row-number>` element displays the current row number for a given table. Please note that the element displays the actual row number from the database and not the row number of the current selection that is used as an attribute value in the `<text:database-row-select>` element.

XML Code:	`<text:database-row-number>`
Rules:	
DTD:	`<!ELEMENT text:database-row-number (#PCDATA)>`

The attributes that you can attach to the `<text:database-row-number>` are:

- `text:database-name` and `text:table-name`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-row-number %database-table; #REQUIRED>`

- `text:num-format` and `text:num-letter-sync`

  These attributes determine how the number should be formatted. See Section 3.2.23 for more information on how to use this attribute.

DTD:	`<!ATTLIST text:database-row-number %num-format; #IMPLIED>`

- `text:value`

  This attribute specifies the current row number. The number changes when new data is added to the current document.

DTD:	`<!ATTLIST text:database-row-number text:value %integer; #IMPLIED>`

3.2.26 Display Current Database and Table

StarOffice keeps track of the last database and table that was used in the document. In other words, the table that is used by the last field that was inserted into the document. In the StarOffice user interface, the database is displayed in the Beamer. The `<text:database-name>` element displays the database and table name of the most recently used table.

XML Code:	`<text:database-name>`
Rules:	
DTD:	`<!ELEMENT text:database-name (#PCDATA)>`

The attributes that you can attach to the `<text:database-name>` element are:

- `text:database-name` and `text:table-name`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-name %database-table;>`

3.2.27 Metadata Fields

Metadata fields display meta information about the document, such as, the document creation date or the time at which the document was last printed. The names of the metadata field elements correspond to the metadata elements described in Section 2.1.

All metadata field elements can be marked as fixed using the `text:fixed` attribute.

Several metadata fields display a date or a time. The elements for these fields require an associated `text:date-value` or a `text:time-value` attribute, and optionally, they can also have a `style:data-style-name` attribute. See Section 3.2.39 for more information on these attributes.

Initial Creator

This element represents the name of the author who created the original document.

XML Code:	`<text:initial-creator>`
Rules:	
DTD:	`<!ELEMENT text:initial-creator (#PCDATA)>` `<!ATTLIST text:initial-creator text:fixed %boolean; "false">`

```
<!ELEMENT text:initial-creator (#PCDATA)>
<!ATTLIST text:initial-creator text:fixed %
boolean; "false">
```

Document Creation Date

This element represents the date on which the document was created.

XML Code:	`<text:creation-date>`
Rules:	
DTD:	see below

```
<!ELEMENT text:creation-date (#PCDATA)>
<!ATTLIST text:creation-date
     text:fixed %boolean; "false"
     text:date-value %date; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Document Creation Time

This element represents the time at which the document was created.

XML Code:	`<text:creation-time>`
Rules:	
DTD:	see below

```
<!ELEMENT text:creation-time (#PCDATA)>
<!ATTLIST text:creation-time
     text:fixed %boolean; "false"
     text:time-value %timeInstance; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Document Description

This element contains a brief description of the document.

XML Code: `<text:description>`

Rules:

DTD:

```
<!ELEMENT text:description (#PCDATA)>
<!ATTLIST text:description
     text:fixed %boolean; "false">
```

User-Defined Document Information

This group of elements contains user-defined information about the document. The fields are not used or interpreted by StarOffice, so the user may use these elements for any purpose.

XML Code: `<text:user-defined>`

Rules:

DTD:

```
<!ELEMENT text:user-defined (#PCDATA)>
<!ATTLIST text:user-defined
          text:name %string; #REQUIRED
          text:fixed %boolean; "false">
```

Print Time

This element represents the time at which the document was last printed.

XML Code: `<text:print-time>`

Rules:

DTD:

```
<!ELEMENT text:print-time (#PCDATA)>
<!ATTLIST text:print-time
     text:fixed %boolean; "false"
     text:time-value %timeInstance; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Print Date

This element represents the date on which the document was last printed.

XML Code: `<text:print-date>`

Rules:

DTD:

```
<!ELEMENT text:print-date (#PCDATA)>
<!ATTLIST text:print-date
     text:fixed %boolean; "false
     text:date-value %date; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Printed By

This element represents name of the last person who printed the document.

XML Code: `<text:printed-by>`

Rules:

DTD:

```
<!ELEMENT text:printed-by (#PCDATA)>
<!ATTLIST text:printed-by
    text:fixed %boolean; "false">
```

Document Title

This element represents the title of the document.

XML Code: `<text:title>`

Rules:

DTD:

```
<!ELEMENT text:title (#PCDATA)>
<!ATTLIST text:title
    text:fixed %boolean; "false">
```

Document Subject

This element represents the subject of the document.

XML Code: `<text:subject>`

Rules:

DTD:

```
<!ELEMENT text:subject (#PCDATA)>
<!ATTLIST text:subject
    text:fixed %boolean; "false">
```

Document Keywords

This element contains a list of keywords used to describe the document.

XML Code: `<text:keywords>`

Rules:

DTD:

```
<!ELEMENT text:keywords (#PCDATA)>
<!ATTLIST text:keywords
    text:fixed %boolean; "false">
```

Document Revision Number

This element contains the document revision number. When the document is created, the revision number is set to 1. Each time the document is saved, the document revision number is incremented.

XML Code: `<text:editing-cycles>`

Rules:

DTD:

```
<!ELEMENT text:editing-cycles (#PCDATA)>
<!ATTLIST text:editing-cycles
        text:fixed %boolean; "false">
```

Note: Since the `<text:editing-cycles>` field can not be formatted, the revision number can be read from the element content. Therefore, no extra attribute is needed.

Document Edit Duration

Every time a document is edited, StarOffice records the duration between the time the document is opened and the time the document is closed. It then adds the duration to an internal counter, thereby keeping track of the total time that has been spent editing the document.

XML Code: `<text:editing-duration>`

Rules:

DTD:

```
<!ELEMENT text:editing-duration (#PCDATA)>
<!ATTLIST text:editing-duration
     text:fixed %boolean; "false"
     text:duration %timeDuration; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Document Modification Time

This element represents the time at which the document was last modified.

XML Code: `<text:modification-time>`

Rules:

DTD:

```
<!ELEMENT text:modification-time (#PCDATA)>
<!ATTLIST text:modification-time
      text:fixed %boolean; "false"
      text:time-value %timeInstance; #IMPLIED
      style:data-style-name %style-name; #IMPLIED>
```

Note: This element displays the information from the `<meta:date>` element. The name was chosen to avoid confusion with `<text:date>` fields.

Document Modification Date

This element represents the date on which the document was last modified.

XML Code: `<text:modification-date duration>`

Rules:

DTD:

```
<!ELEMENT text:modification-date (#PCDATA)>
<!ATTLIST text:modification-date
     text:fixed %boolean; "false"
     text:date-value %date; #IMPLIED
     style:data-style-name %style-name; #IMPLIED>
```

Document Modified By

This element represents the name of the person who last modified the document.

XML Code:	`<text:creator>`
Rules:	

DTD:	`<!ELEMENT text:creator (#PCDATA)>`
	`<!ATTLIST text:creator`
	`    text:fixed %boolean; "false">`

3.2.28 Conditional Text Fields

Text fields can be used to display one text or another, depending on the condition. Conditional text fields are given a condition and two text strings. If the condition is true, one of the text strings is displayed. If the condition is false, the other text string is displayed.

XML Code:	`<text:conditional-text>`
Rules:	
DTD:	`<!ELEMENT text:conditional-text (#PCDATA)>`

The attributes that you can associate with the `<text:conditional-text>` element are:

- Condition
- Text to display if the condition is true
- Text to display if the condition is false

Condition

The condition attribute contains a boolean expression. Depending on the result, the value of the `text:display-if-true` or `text:display-if-false` attribute is displayed.

XML Code:	`text:condition`
Rules:	
DTD:	`<!ATTLIST text:conditional-text text:condition %formula; #REQUIRED>`

Text to Display If the Condition is True

This attribute contains the text string to display if the condition is `true`.

XML Code:	`text:string-value-if-true`
Rules:	If the condition is `true`, the value of this attribute is displayed.
DTD:	`<!ATTLIST text:conditional-text text:string-value-if-true %string; #REQUIRED>`

Text to Display If the Condition is False

This attribute contains the text string to display if the condition is `false`.

XML Code:	`text:string-value-if-false`

Rules:	If the condition evaluates to `false`, the value of this attribute is displayed.
DTD:	`<!ATTLIST text:conditional-text text:string-value-if-false %string; #REQUIRED>`

3.2.29 Hidden Text Field

The hidden text field is closely related to the conditional text field. It displays fixed text, except when the condition is `true` when it does not display anything.

XML Code:	`<text:hidden-text>`
Rules:	
DTD:	`<!ELEMENT text:hidden-text (#PCDATA)>`

The attributes that you can associate with the `<text:hidden-text>` element are:

- Condition
- Text

Condition

The `text:condition` attribute contains a boolean expression. If the expression evaluates to `true`, the text is hidden.

XML Code:	`text:condition`
Rules:	
DTD:	`<!ATTLIST text:hidden-text text:condition %formula; #REQUIRED>`

Text

The `text:string-value` attribute specifies the text to display if the condition is `false`.

XML Code:	`text:string-value`
Rules:	The value of this attribute is displayed if the condition evaluates to `false`.
DTD:	`<!ATTLIST text:hidden-text text:string-value %formula; #REQUIRED>`

3.2.30 Hidden Paragraph Fields

The hidden paragraph field has a similar function to the hidden text field. However, the hidden paragraph field does not have any content. It hides the paragraph in which it is contained. This allows you to hide or display a paragraph of formatted text, depending on whether a condition is `true` or `false`.

Hidden paragraph fields are often used together with form letters. For example, if a condition depends on a database field, a hidden paragraph field can be used to selectively include paragraphs in the form letter depending on the database content. Multiple paragraph fields can be contained one paragraph. The paragraph is displayed if the condition associated with at least one hidden paragraph field is `false`. Alternatively, you can combine the conditions associated with several hidden paragraph fields into a single condition for a single field using logical operations on the conditions.

XML Code:	`<text:hide-paragraph>`
Rules:	
DTD:	`<!ELEMENT text:hide-paragraph EMPTY>`
Note:	Unlike most fields, this field does not display text, but it affects the entire paragraph in which it is contained.

The attribute that you can associate with the `<text:hide-paragraph>` element is:

- Condition

Condition

The `text:condition` attribute contains a boolean expression.

XML Code:	`text:condition`
Rules:	If the condition is `true`, the paragraph is hidden. If the condition is `false`, the paragraph is displayed.
DTD:	`<!ATTLIST text:hide-paragraph text:condition %formula; #REQUIRED>`

3.2.31 Chapter Fields

Chapter fields display one of the following:

- The name of the current chapter
- The number of the current chapter
- Both the name and number of the current chapter

If the chapter field is placed inside a header or footer, it displays the current chapter name or number on every page.

XML Code:	`<text:chapter>`
Rules:	
DTD:	`<!ELEMENT text:chapter (#PCDATA)>`

The attributes that you can associate with the `<text:chapter>` element are:

- Display
- Outline level

Display

The `text:display` attribute specifies the information that the chapter field should display.

XML Code:	`text:display`

Rules:	
DTD:	`<!ATTLIST text:chapter text:display ( name | number | number-and-name | plain-number-and-name | plain-number ) "number-and-name">`
Implementation Limitation:	In the current version of the StarOffice Writer user interface, `plain-number-and-name` chapter fields are not supported.

Example: If the current chapter number is 2.4, the chapter title is Working with Tables, the prefix is [, and suffix is], the possible display options and results are as follows:

Value of `text:display` attribute	Field content displayed
`number`	[2.4]
`name`	Working with Tables
`number-and-name`	[2.4] Working with Tables
`plain-number`	2.4
`plain-number-and-name`	2.4 Working with Tables

Outline Level

This attribute allows you to specify the outline level to use. The chapter field displays the chapter number or title up to the specified outline level.

XML Code:	`text:outline-level`
Rules:	
DTD:	`<!ATTLIST text:chapter text:outline-level %integer; "1">`
Note:	StarOffice Writer currently supports up to ten outline levels.

3.2.32 File Name Fields

File name fields display the name of the file that is currently being edited.

XML Code:	`<text:file-name>`
Rules:	
DTD:	`<!ELEMENT text:file-name (#PCDATA)>`

The attributes that you can associate with the `<text:file-name>` element are:

- Display
- Fixed

Display

The `text:display` attribute specifies how much of the file name to display. You can choose whether to display:

- The full file name including the path and the extension

- The file path only
- The file name only
- The file name and the extension

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:file-name text:display ( full | path | name | name-and-extension ) "full">`

Fixed File Name Fields

If a file name field is fixed, its value does not change when the file is edited.

XML Code:	`text:fixed`
Rules:	
DTD:	`<!ATTLIST text:file-name text:fixed %boolean; "false">`

3.2.33 Document Template Name Fields

The document template name field displays information about the document template in use, such as the template title or the file name.

XML Code:	`<text:template-name>`
Rules:	
DTD:	`<!ELEMENT text:template-name (#PCDATA)>`

The attribute that you can associate with the `<text:template-name>` element is:

- Display

Display

This attribute specifies which information about the document template to display. You can choose to display:

- The full file name including the path and the extension
- The file path only
- The file name only
- The file name and the extension
- The title
- The area of the document template

The latter two values are used in the StarOffice Writer user interface document template dialog. The display values are a superset of the display values available for the `<text:file-name>` element.

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:template-name text:display ( full | path | name | name-and-extension | area | title ) "full">`

3.2.34 Page Variable Fields

Page variables allow you to define an alternative page numbering scheme. There is only one page variable, and it is set by any set page variable field in the document. The value of the page variable is increased on each page, in the same way as regular page numbers.

Setting Page Variable Fields

To set a page variable field, you use the `<text:set-page-variable>` element.

XML Code:	`<text:set-page-variable>`
Rules:	
DTD:	`<!ELEMENT text:set-page-variable EMPTY>`

Turning Page Variables On or Off

At the beginning of a document, the page variable is inactive. You can use the `text:active` attribute to disable a page variable after it was used in the document.

XML Code:	`text:active`
Rules:	
DTD:	`<!ATTLIST text:set-page-variable text:active %boolean; "true">`

Page Variable Adjustment

The `text:page-adjust` attribute determines the page adjustment. The value of the active page variable is the current page number plus the closest page adjustment value that was previously set.

XML Code:	`text:page-adjust`
Rules:	
DTD:	`<!ATTLIST text:set-page-variable text:page-adjust %integer; "0">`

Displaying Page Variable Fields

The `<text:get-page-variable>` element displays the value of the page variable. The field can be formatted in the same way as regular page number fields.

XML Code:	`<text:get-page-variable>`
Rules:	
DTD:	`<!ELEMENT text:get-page-variable (#PCDATA)>`

The attributes that you can associate with the `<text:get-page-variable>` element are:

- `text:num-format` and `text:num-letter-sync`

  These attributes determine how the number should be formatted. See Section 3.2.23 for more information on how to use these attributes.

DTD:	`<!ATTLIST text:get-page-variable %num-format; #IMPLIED>`

3.2.35 Macro Field

The macro field contains the name of a macro that is executed when the field is activated. The field also contains a description that is displayed as the field content.

XML Code:	`<text:execute-macro>`
Rules:	
DTD:	`<!ELEMENT text:execute-macro (#PCDATA)>`

The attribute that you can associate with the `<text:execute-macro>` element is:

- Macro name

Macro Name

The `text:name` attribute specifies the macro to invoke when the field is activated.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:execute-macro text:name %string; #REQUIRED>`

3.2.36 DDE Connections

A Dynamic Data Exchange (DDE) connection consists of the parameters for the DDE target application, a file name, and a command string. A DDE connection also takes a parameter that specifies whether it will be updated automatically or only on the user's request. Every DDE connection must be named.

Container for DDE Connection Declarations

The DDE connection declarations are contained in one declarations element.

XML Code:	`<text:dde-connection-decls>`
Rules:	

XML Code:	`<text:dde-connection-decls>`
DTD:	`<!ELEMENT text:dde-connections-decls (text: dde-connection-decl)*>`

Declaring DDE Connections

Every DDE connection is declared using a declaration element. Multiple DDE fields can refer to one DDE connection by using the same name. The declaration element has no content.

XML Code:	`<text:dde-connection-decl>`
Rules:	
DTD:	`<!ELEMENT text:dde-connections-decl  EMPTY>`

The attributes that you can associate with the `<text:dde-connection-decl>` element are:

- Connection name
- DDE target application
- DDE target file name
- DDE command
- Automatic update flag

Connection Name

The `text:name` attribute specifies the name by which the connection will be referred.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:name % string; #REQUIRED>`

Target Application

The `text:dde-application` attribute specifies the name of the target application to use for the DDE connection.

XML Code:	`text:dde-application`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:dde-application %string; #REQUIRED>`
Note:	The target name for StarOffice is `soffice`. Therefore, internal DDE links have the attribute `text:dde-application="soffice"`.

Target Topic

The `text:dde-topic` attribute specifies the name of the topic to use for the DDE connection.

XML Code:	`text:dde-topic`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:dde-topic %string; #REQUIRED>`
Note:	If the target application is StarOffice, it interprets the DDE topic as the name of the file.

Target Item

The `text:dde-item` attribute specifies which information the target application should deliver.

XML Code:	`text:dde-item`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:dde-item %string; #REQUIRED>`
Note:	If the target application for the DDE connection is StarOffice Writer, the item represents the name of a bookmark. StarOffice delivers the current text content to the requesting application.

Automatic Update

StarOffice Writer can automatically update DDE links. If preferred, you can use the `text:automatic-update` attribute to specify that the DDE connection links should only be updated at the request of the user.

XML Code:	`text:automatic-update`
Rules:	If the value of this attribute is `true`, the DDE links are automatically updated. If this value of this attribute is `false`, the DDE links are updated on user request only.
DTD:	`<!ATTLIST text:dde-connection-decl text:automatic-update %boolean; "false">`

3.2.37 DDE Connection Fields

A DDE field allows you to display information from a DDE connection. The only parameter required for the DDE field is the name of the DDE connection that supplies the data to this field. This DDE connection element specifies the actual DDE field that appears in the text body.

XML Code:	`<text:dde-connection>`
Rules:	
DTD:	`<!ELEMENT text:dde-connection (#PCDATA)>`

The attribute that you can associate with the `<text:dde-connection>` element is:

- DDE connection name

DDE Connection Name

The `text:name` attribute specifies the name of the DDE connection to which the field refers.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:dde-connection text:name %string; #REQUIRED>`

3.2.38 Reference Fields

StarOffice Writer uses five types of reference field and each type is represented by its own element. The reference field types are based on the type of element they refer to; footnotes, endnotes, bookmarks, references, and sequences. Every reference contains a reference format which determines what information about the referenced target is displayed. For example, references can display:

- The page number of the referenced target
- The chapter number of the referenced target
- Wording indicating whether the referenced target is above or below the reference field

In addition, each reference field must identify its target which is usually done using a name attribute. Bookmarks and references are identified by the name of the respective bookmark or reference. Footnotes, endnotes, and sequences are identified by a name that is usually generated automatically when a document is exported.

XML Code:	`<text:reference-ref>`
	`<text:sequence-ref>`
	`<text:bookmark-ref>`
	`<text:footnote-ref>`
	`<text:endnote-ref>`

Rules:	

DTD:	`<!ELEMENT text:reference-ref (#PCDATA)>`
	`<!ELEMENT text:sequence-ref (#PCDATA)>`
	`<!ELEMENT text:bookmark-ref (#PCDATA)>`
	`<!ELEMENT text:footnote-ref (#PCDATA)>`
	`<!ELEMENT text:endnote-ref (#PCDATA)>`

The attributes that you can associate with the reference field elements are:

- Reference name
- Reference format

Reference Name

The `text:ref-name` attribute identifies the referenced element. Since bookmarks and references have a name, this name is used by the respective reference fields. Footnotes, endnotes, and sequences are assigned names by the application used to create the StarOffice XML file format when the document is exported.

XML Code:	`text:ref-name`
Rules:	

DTD:	`<!ATTLIST text:reference-ref text:ref-name %string; #REQUIRED>`

Reference Format

The `text:reference-format` attribute determines what information about the reference is displayed.

XML Code:	`text:reference-format`
Rules:	All types of reference fields support the following values for this attribute formats: • `page`, which displays the number of the page on which the referenced item appears. • `chapter`, which displays the number of the chapter in which the referenced item appears. • `direction`, which displays whether the referenced item is above or below the reference field. • `text`, which displays the text of the referenced item. If the reference format is not specified, the page format is used as the default. References to sequence fields support the following three additional values: • `category-and-value`, which displays the name and value of the sequence. • `caption`, which displays the caption in which the sequence is used. • `value`, which displays the value of the sequence.

DTD:	`<!ATTLIST text:reference-ref text:reference-format (page|chapter|text|direction) #IMPLIED>`
	`<!ATTLIST text:footnote-ref text:reference-format (page|chapter|text|direction) #IMPLIED>`
	`<!ATTLIST text:endnote-ref text:reference-format (page|chapter|text|direction) #IMPLIED>`
	`<!ATTLIST text:bookmark-ref text:reference-format (page|chapter|text|direction) #IMPLIED>`
	`<!ATTLIST text:sequence-ref text:reference-format (page|chapter|text|direction|category-and-value|caption|value) #IMPLIED>`

Example:

The following table shows all possible reference formats and the resulting reference display.

Reference format	A reference to the sequence used in the caption of the table using the specified format
`page`	158
`chapter`	3.2.38
`text`	Tabelle 2: Examples of reference formats
`direction`	unten
`category-and-value`	Tabelle 2

Reference format	**A reference to the sequence used in the caption of the table using the specified format**
`caption`	Examples of reference formats
`value`	2

Tabelle 2: Examples of reference formats

3.2.39 Common Field Attributes

You can use the attributes described in this section with several field elements.

Variable Value Types and Values

Variables and most variable fields have a current value. Every variable has a value type that must be specified when the field supports multiple value types. The value type is specified using the `text:value-type` attribute.

XML Code:	`text:value-type`
Rules:	This attribute must specify one of the following value types for the variable: `float`, `percentage`, `currency`, `date`, `time`, `boolean`, or `string`.
DTD:	`<!ENTITY % value-type-attlist "text:value-type ( float | time | date | percentage | currency | boolean | string ) #REQUIRED">`
Note:	This entity should be used within any `<!ATTLIST>` definitions for `text:value-type` attributes.

Example:

```
<!ELEMENT some-element (#PCDATA)>
<!ATTLIST some-element %value-type-attlist>
```

Depending on the value type, the value itself is written to different value attributes. The supported value types, their respective value attributes, and how the values are encoded are described in the following table:

Value of `text:value-type`	**Value Attribute**	**Encoded as...**
`float`, `percentage`	`text:value`	Numeric value
`currency`	`text:value` and `text:currency`	Numeric value and currency symbol
`date`	`text:date-value`	Described in ISO 8601, §5.2.1.1, extended format
`time`	`text:time-value`	Described in ISO 8601, §5.4.1 a), extended format
`boolean`	`text:boolean-value`	`true` or `false` strings
`string`	`text:string-value`	Strings

The StarOffice Writer concept of field values and value types and their encoding in XML is modeled on the corresponding XML for table cell attributes. See Section 4.7.2 for more detailed information on these attributes.

The definition of the entity `%value-attlist;` is as follows:

DTD:	`<!ENTITY % value-attlist`
	`  "%value-type-attlist;`
	`   text:value %float; #IMPLIED`
	`   text:date-value %date; #IMPLIED`
	`   text:time-value %time; #IMPLIED`
	`   text:boolean-value %boolean; #IMPLIED`
	`   text:string-value %string; #IMPLIED`
	`   text:currency CDATA #IMPLIED" >`

Example:	`<!ELEMENT some-element (#PCDATA)>`
	`<!ATTLIST some-element %value-attlist;>`

This entity is useful for defining all value and value type related attributes for any element.

Fixed

The `text:fixed` attribute specifies whether or not the value of a field element is fixed. If the value of a field is fixed, the original value of the field is preserved. If the value of the field is not fixed, the value of the field is replaced by a new value when the document is edited.

This attribute can be used with:

- Date fields
- Time fields
- Page number fields
- All sender fields
- All author fields

XML Code:	`text:fixed`
Rules:	If the value of this attribute is set to `true`, the value of the field element to which this attribute is attached is preserved in all future edits of the document. If the value of this attribute is set to `false`, the value of the field element is not preserved and will be replaced with new values as appropriate.
Sample DTD:	`<!ATTLIST text:author-name text:fixed %boolean; "true">`

Variable Name

Use the `text:name` attribute to specify the name of a variable when you are declaring, setting, or displaying a variable. You can use this attribute with any of the following elements:

- `<text:variable-decl>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-decl>`

- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:sequence-decl>`
- `<text:sequence>`

XML Code:	`text:name`
Rules:	When you are using this attribute to specify the name of a variable to display, a variable of the appropriate type with the same name must already have been declared.
Sample DTD:	`<!ATTLIST text:sequence text:name %variable-name; #REQUIRED>`

Description

The `text:description` attribute contains a brief message that is displayed when users are prompted for input. You can use this attribute with any of the following elements:

- `<text:placeholder>`
- `<text:variable-input>`
- `<text:user-field-input>`
- `<text:text-input>`

XML Code:	`text:description`
Rules:	The optional `text:description` attribute may contain a brief description.
Sample DTD:	`<!ATTLIST text:text-input text:description %string; #IMPLIED>`

Display

The `text:display` attribute supports up to three values as follows:

- `value`
  This value displays the value of the field. Some fields do not support this value. In these cases, the `text:display` attribute only takes the values `value` or `none`, and `value` or `formula`, respectively.
- `formula`
  This value allows you to display the formula rather than the value of the field. Some fields do not support this value. In these cases, the `text:display` attribute only takes the values `value` or `none`, and `value` or `formula`, respectively.
- `none`
  Several variable fields support this value, which hides the field content. This allows you to set variables in one part of the document and display them in another part of the document.

You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`

- `<text:user-field-get>`
- `<text:expression>`

XML Code:	`text:display`
Rules:	If the value of this attribute is `value`, the value of the field is displayed. If the value is `formula`, the formula expression used to compute the value is displayed. Otherwise, the field is not be displayed.
Sample DTD:	`<!ATTLIST text:user-field-get text:display ( value | formula | none ) "value">`

Formula

The `text:formula` attribute contains the formula or expression used to compute the value of the field. You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:user-field-decl>`
- `<text:sequence>`
- `<text:expression>`

XML Code:	`text:formula`
Rules:	
Sample DTD:	`<!ATTLIST text:expression text:formula %formula; #REQUIRED>`

Formatting Style

This attribute refers to the data style used to format the numeric value. For general information about styles, see Section 1.6. For more information about data styles, see Section .

You can use this attribute with any of the following elements:

- `<text:date>`
- `<text:time>`
- `<text:page-number>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:expression>`

XML Code:	`style:data-style-name`

Rules:	For string variables you must omit this attribute. Otherwise, this attribute is required. The name must match the name of a data style.
Sample DTD:	`<!ATTLIST text:expression style:data-style-name %style-name; #IMPLIED>`

Number Formatting Style

You can format numbers that are used for number sequences such as page numbers or sequence fields according to the number styles described in Section 2.9. The number styles supported are as follows:

- Numeric: 1, 2, 3, ...
- Alphabetic: a, b, c, ... or A, B, C, ...
- Roman: i, ii, iii, iv, ... or I, II, III, IV,...

XML Code:	`text:num-format`
Rules:	The value of this attribute can be any of the XSL number format keys `1`, `i`, `I`, `a`, or `A`.
Sample DTD:	`<!ATTLIST some-element text:num-format CDATA #IMPLIED>`

Alphabetic number styles need an additional attribute to determine how to display numbers that cannot be represented by a single letter. The StarOffice software supports

- Synchronized letter numbering, where letters are used multiple times, for example aa, bb, cc, and so on.
- Non-synchronized letter numbering, for example aa, ab, ac, and so on.

See Section 2.9 for more information.

XML Code:	`text:num-letter-sync`
Rules:	
Sample DTD:	`<!ATTLIST some-element text:num-letter-sync %boolean; "false">`

The following entity aids the definition of elements that use number formats:

XML Code:	`%num-format;`
Sample DTD:	`<!ENTITY % num-format 'text:num-format CDATA #IMPLIED text:num-letter-sync %boolean; "false"'>`
Example:	`<!ELEMENT some-element %num-format;>`

3.3 Sections

A text section is a named region of text that can be associated with certain formatting properties. The section starts and ends on paragraph boundaries and can contain any number of paragraphs. Sections can contain regular text content or the text can be contained in another file and linked to the section. Sections can also be write-protected or hidden.

If a section is linked to another document, the link can be through one of the following:

- A resource identified by an Xlink, represented by a `text:section-source` element
- Dynamic Data Exchange (DDE,) represented by a `office:dde-source` element

If these elements are used, they must be the first element in a `<text:section>`.

Sections can have settings for text columns, background color or pattern, footnote and endnote configuration. These settings form the section style, which is represented in a `<style:styles>` element. The formatting properties for sections are explained in Section 3.13.

XML Code:	`<text:section>`
Rules:	
DTD:	`<!ELEMENT text:section ((text:section-source|office:dde-source)?, %text;)>`
Note:	In StarOffice Writer, lists and sections can overlap. Therefore, a text section can start on any paragraph boundary even if the paragraph is part of a numbered or bulleted list. This causes a problem in the XML representation of lists because lists are represented using their own container elements and XML cannot represent overlapping elements. To solve the problem, a section boundary within a list causes the list to split.

Section Style

The `text:style-name` attribute specifies the section style.

XML Code:	`text:style-name`
Rules:	This attribute must refer to a section style.
DTD:	`<!ATTLIST text:section text:style-name %styleName; #IMPLIED>`

Section Name

Every section must have a name that uniquely identifies the section. The `text:name` attribute contains the name of the section.

XML Code:	`text:name`
Rules:	You should not change a section that is marked as protected.
DTD:	`<!ATTLIST text:section text:name %string; #REQUIRED>`

Hidden Sections and Conditional Sections

Sections can be hidden based on a condition or they can be hidden unconditionally.

The `text:display` attribute specifies whether or not the section is hidden. The `text:condition` attribute specifies the condition under which the section is hidden.

XML Code:	`text:display`
Rules:	The value of this attribute can be: • `true`, the section is displayed. This is the default setting. • `none`, the section is hidden unconditionally. • `condition`, the section is hidden under the condition specified in the `text:condition` attribute.
DTD:	`<!ATTLIST text:section text:display (true|none|condition) "true">`

The `text:condition` attribute specifies the condition under which the section is hidden. The condition is encoded as a string.

XML Code:	`text:condition`
Rules:	If the value of `text:display` is `condition`, the `text:condition` attribute must be present.
DTD:	`<!ATTLIST text:section text:condition %formula; #IMPLIED>`

3.3.1 Section Source

The `<text:section-source>` element indicates that the enclosed section is a linked section. If this element is used, it must be the first element in the `<text:section>` element.

XML Code:	`<text:section-source>`
Rules:	This element does not have any contents.
DTD:	`<!ELEMENT text:section-source EMPTY>`

The attributes associated with the `<text:section-source>` attribute are:

- Section source URL
- Name of linked section
- Filter name

Section Source URL

These attributes identify the document or section to which the section is linked.

XML Code:	`xlink:href, xlink:type, xlink:show`
Rules:	The name of the target section is identified by the local part of the URL, following the hash mark.

DTD:	`<!ATTLIST text:section-source`
	`            xlink:href %string; #IMPLIED`
	`            xlink:type (simple) #FIXED "simple"`
	`            xlink:show (embed) #FIXED "embed">`

Note:	The `xlink:href` attribute is implied because `<text:section-source>` elements may also link to internal sections.

Name of Linked Section

If the link targets a section of a document, the attribute `text:section` name contains the name of the target section.

XML Code:	`text:section-name`
Rules:	If the attribute is not present, the link targets the entire document.
DTD:	`<!ATTLIST text:section-source text:section-name %string; #IMPLIED>`

Filter Name

The `text:filter-name` attribute specifies the file type of the link target.

XML Code:	`text:filter-name`
Rules:	The value of this attribute is implementation-dependent.
DTD:	`<!ATTLIST text:section-source text:filter-name %string; #IMPLIED>`

3.3.2 DDE Source

If sections are linked via DDE, they are represented by a `<office:dde-source>` element. It contains attributes that specify the application, topic and item of the DDE connection.

XML Code:	`<office:dde-source>`
Rules:	
DTD:	`<!ELEMENT office:dde-source EMPTY>`

The attributes that you can associate with the `<office:dde-source>` element are:

- Target application
- Target topic
- Target item
- Automatic update

Target Application

The `office:dde-application` attribute specifies the name of the target application to use for the DDE connection.

XML Code:	`office:dde-application`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-application %string; #REQUIRED>`

Target Topic

The `office:dde-topic` attribute specifies the topic to use for the DDE connection.

XML Code:	`office:dde-topic`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-topic %string; #REQUIRED>`

Target Item

The `office:dde-item` attribute specifies the information that the target application will deliver.

XML Code:	`office:dde-item`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-item %string; #REQUIRED>`

Automatic Update

The `office:automatic-update` attribute indicates whether or not the linked section should be automatically updated.

XML Code:	`office:automatic-update`
Rules:	If this attribute is set `true`, the linked sections should be updated automatically.
DTD:	`<!ATTLIST office:dde-source office:automatic-update %boolean; "false">`

3.4 Change Tracking

This section describes how StarOffice tracks changes to text content.

3.4.1 Tracked Changes

All tracked changes to text documents are stored in a list. The list contains an element for each change made to the document.

XML Code:	`<text:tracked-changes>`
Rules:	
DTD:	`<!ELEMENT text:tracked-changes (text:changed-region)+>`

3.4.2 Changed Regions

For every changed region of a document, there is one entry in the list of tracked changes. This entry contains a list of all changes that were applied to the region. The start and end of this region are marked by the start and end elements that are described in the next section.

XML Code:	`<text:changed-region>`
Rules:	Every element has an ID. The elements that mark the start and end of a region use this ID to identify the region to which they belong.

DTD:	`<!ELEMENT text:changed-region (text:insertion|text:deletion|text:format-change)+>`
	`<ATTLIST  text:changed-region text:id ID #REQUIRED>`

3.4.3 Region Start and End

There are three elements that mark the start and the end of a changed region, as follows:

- Change start element – `<text:change-start>`
  This element marks the start of a region with content where text has been inserted or the format has been changed.
- Change end element – `<text:change-end>`
  This element marks the end of a region with content where text has been inserted or the format has been changed.
- Change position element – `<text:change>`
  This element marks a position in an empty region where text has been deleted.

XML Code:	`<text:change-start>`
	`<text:change-end>`
	`<text:change>`

Rules:	All three elements have an attribute that specifies the ID of the region to which they belong.

DTD:	`<!ELEMENT text:change-start EMPTY>`
	`<!ELEMENT text:change-end EMPTY>`
	`<!ELEMENT text:change EMPTY>`
	`<!ATTLIST text:change-start text:region-id IDREF #REQUIRED>`
	`<!ATTLIST text:change-end text:region-id IDREF #REQUIRED>`
	`<!ATTLIST text:chang text:region-id IDREF #REQUIRED>`

3.4.4 Insertion

The `<text:insertion>` element contains the information that is required to identify any insertion of content. This content can be a piece of text within a paragraph, a whole paragraph, or a whole table. The inserted content is part of the text document itself and is marked by a change start and a change end element.

XML Code:	`<text:insertion>`
Rules:	
DTD:	`<!ELEMENT text:insertion (office:change-info)>`

Example: Insertion of text

```
<text:tracked-changes>
  <text:changed-region text:id="c001">
    <text:insertion>
      <office:change-info office:chg-author="Michael Brauer"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04"/>
    </text:insertion>
  </text:changed-region>
</text:tracked-changes>
...
<text:p>
  This is the original text<text:change-start text:region-id="c001"/>,
  but this has been added</text:change-end text:region-id="c001"/>.
</text:p>
```

3.4.5 Deletion

A `<text:deletion>` element contains content that was deleted while change tracking was enabled. The position where the text was deleted is marked by the change position element (`<text:change>`).

XML Code:	`<text:deletion>`
Rules:	If part of a paragraph was deleted, the text that was deleted is contained in this element as a paragraph element. If the deleted text is reinserted into the document, the paragraph is joined with the paragraph where the deletion took place.
DTD:	`<!ELEMENT text:deletion (style:change-info,%text;)>`

Example: Deletion of text

```
<text:tracked-changes>
  <text:changed-region text:id="c002">
    <text:deletion>
      <office:change-info office:chg-author="Michael Brauer"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04">
      <text:p>
        , but this has been deleted
      </text:p>
    </text:deletion>
  </text:changed-region>
</text:tracked-changes>
...
<text:p>
  This is the original text<text:change text:region-id="c002"/>.
</text:p>
```

This example shows:

- Deleted text = `but this has been deleted`
  This text is contained in the `<text:p>` element within the `<text:deletion>` element.
- Current text = `This is the original text.`
  This text is contained in the `<text:p>` element at the end of the example.
- Original text before deletion took place = `This is the original text, but this has been deleted.`

3.4.6 Format Change

A format change element represents any change in formatting attributes. The region where the change took place is marked by a change start and a change end element.

XML Code:	`<text:format-change>`
Rules:	
DTD:	`<!ELEMENT text:format-change (style:change-info)>`
Note:	A format change element does not contain the actual changes that took place.

3.5 Bulleted and Numbered Lists

Bulleted and numbered lists consist of structural and layout information.

Structural information includes the following:

- List type, bulleted or numbered.
- List level, for example main or sublist.
- Information about whether or not a certain paragraph contained in a list has a label, for example number or bullet.
- The number of a paragraph within a numbered list. This information is optional because it can be recalculated.

Layout information includes the following:

- The indentation of paragraphs in a list.
- The label width and the distance between it and the text.
- The bullet character or image for bulleted lists.
- The number format for numbered lists.

The structural information is contained in the document body, with the content. The StarOffice XML representation of structural information is very similar to HTML. The layout information is contained within **list styles**. There are common list styles and automatic list styles.

3.5.1 List Blocks

A list is represented by the one of the following elements:

- `<text:ordered-list>`
  This element specifies an ordered list, that is a list where every list item is preceded by a number that is incremented for each list item.
- `<text:unordered-list>`
  This element specifies an unordered list, that is a list where every list item is preceded by the same bullet character or image.

XML Code:	As above
Rules:	Both elements have the same content; an optional list header followed by any number of list items. Every list has a **list level.** If a list is not contained within another list, the list level is 1. If the list in contained within another list, the list level is the list level of the list in which is it contained incremented by one. If a sublist is contained in a table cell or text box, the list level returns to 1, even though the list elements are nested.

DTD:	`<!ENTITY % list-items "((text:list-header|text:list-item), text:list-item*)">`
	`<!ELEMENT text:ordered-list %list-items;>`
	`<!ELEMENT text:unordered-list %list-items;>`

The attributes that you can associate with the list block elements are:

- Style name
- Continue numbering

Style Name

The `text:style-name` attribute specifies the name of the list style that is applied to the list.

XML Code:	`text:style-name`
Rules:	This attribute is optional and can be used with the `<text:ordered-list>` or `<text: unordered-list>` element. If this attribute is not included and therefore a list style is not specified, one of the following actions is taken: • If the list is contained within another list, the list style defaults to the style of the surrounding list. • If there is no list style specified for the surrounding list but the list contains paragraphs that have paragraph styles attached specifying a list style, this list style is used for any of these paragraphs. A default list style is applied to any other paragraphs.

DTD:	`<!ATTLIST text:ordered-list text:style-name %style-name; #IMPLIED>`
	`<!ATTLIST text:unordered-list text:style-name %style-name; #IMPLIED>`

Note:	To determine which formatting properties are applied to a list, the list level and list style name are taken into account. See Section 3.5.4 for more information on list formatting properties.

Continue Numbering

By default, the first list item in an ordered list starts with the number specified in the list style. If the list follows another ordered list and you want to continue the numbering from the preceding list, you can use the continue numbering attribute.

XML Code:	`text:continue-numbering`
Rules:	This attribute can be used with the `<text:ordered-list>` element and can have a value of `true` or `false`. If the value of the attribute is `true` and the numbering style of the preceding list is the same as the current list, the number of the first list item in the current list is the number of the last item in the preceding list incremented by one.
DTD:	`<!ATTLIST text:ordered-list text:continue-numbering %boolean; "false">`

3.5.2 List Header

A list header contains one or more paragraphs that are displayed before a list. The paragraphs are formatted like list items but they do not have a preceding number or bullet. The list header is represented by the list header element.

XML Code:	`<text:list-header>`
Rules:	This element contains paragraphs or sections. The element cannot contain headings, tables, or lists.
DTD:	`<!ELEMENT text:list-header (text:p|text:section)+>`

3.5.3 List Item

A `<text:list-item>` element can contain paragraphs, sections, or lists.

XML Code:	`<text:list-item>`
Rules:	The first line in a list item is preceded by a bullet or number, depending on the list style assigned to the list. If a list item starts another list immediately and does not contain any text, no bullet or number is displayed. A list item cannot contain headings or tables.

DTD:	`<!ENTITY % list-item-content "(text:p|text:section|`
	`text:ordered-list|text:unordered-list)+"`
	`<!ELEMENT text:list-item %list-item-content;>`

The attributes that you can associate with the `<text:list-item>` element are:

- Restart numbering
- Restart numbering value
- Current number

Restart Numbering

You can restart the numbering of a list and the numbering of the surrounding lists by attaching the `text:restart-numbering` attribute to the `<text:list-item>` element.

XML Code:	`text:restart-numbering`
Rules:	This attribute can have a value of `true` or `false`. It can be used for list items in ordered or unordered lists. If the attribute is applied to a list item in an unordered list, it affects the numbering of all surrounding ordered lists and ordered lists that are contained within the item.
DTD:	`<!ATTLIST text:list-item text:restart-numbering %boolean; "false">`

Restart Numbering Value

You can restart the numbering of the current list at a certain number. Use the `text:start-value` attribute to specify the number with which to restart the list.

XML Code:	`text:start-value`
Rules:	This attribute can only be applied to paragraphs with a numbering list style. Unlike the `text:restart-numbering` attribute, it restarts the numbering of the current list only.
DTD:	`<!ATTLIST text:list-item text:start-value %number; #IMPLIED>`

Current Number

To speed up the conversion or loading of XML documents, the current numbers for a number sequence can be contained in a document. If the numbers are contained in the document, every paragraph must be numbered.

There is also an attribute that can be applied to list styles and that must be specified so that the StarOffice software can recognize the current numbers for lists. If a document is saved using a StarOffice application and is not subsequently changed by another application, the numbers are recognized. This attribute is optional.

XML Code:	`text:current-number` `text:use-current-numbers`
Rules:	This attribute is associated with the paragraph list information element. To enable the recognition of current numbers in lists, the attribute `text:use-current-numbers` must be associated with the list style elements.

DTD:	`<!ATTLIST text:list-info text:current-number %number; #IMPLIED>`
	`<!ATTLIST text:list-style text:use-current-numbers (yes|no) "no">`

Implementation limitation:	Currently, the `text:current-number` and `text:use-current-numbers` attributes are not supported.

Example: Ordered and unordered lists and sublists

```
<text:ordered-list text:style-name="List 1">
  <text:list-item>
    <text:p>This is the first list item</text:p>
    <text:p>This is a continuation of the first list item.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p>This is the second list item.
            It contains an unordered sub list.</text:p>
    <text:unordered-list>
      <text:list-item><text:p>This is a sub list item.</text:p>
      <text:list-item><text:p>This is a sub list item.</text:p>
      <text:list-item><text:p>This is a sub list item.</text:p>
    </text:unordered-list>
  </text:list item>
  <text:list-item>
    <text:p>This is the third list item</text:p>
  </text:list-item>
</text:ordered-list>
```

3.5.4 List Styles

List styles specify the formatting properties for lists. A list style contains a set of specifications, each specification containing a set of properties to apply to a list of a certain list level. These specifications are called **list level styles**. If a list style is applied to a list but it does not contain a list level specification for the level of the list, the list level style of the nearest lower level is used. If a suitable list level style does not exist, a default style is used.

XML Code:	`<text:list-style>`
Rules:	This element contains a set of number or bullet list styles for different list levels. The list styles can be common or automatic styles.
DTD:	`<!ELEMENT text:list-style (text:list-level-style-number| text:list-level-style-bullet| text:list-level-style-image)+>`
Note:	List styles contain different properties than paragraph or text styles. This is why they are represented by a different element.

The attributes that you can associate with the `<text:list-style>` element are:

- Name
- Flag for recognition of current numbers
- Consecutive numbering

Name

The `style:name` attribute specifies the name of the list style.

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST text:list-style style:name %style-name; #REQUIRED>`

Flag for Recognition of Current Numbers

See Section 3.5.3 for more information on the current numbering attributes.

Consecutive Numbering

The `text:consecutive-numbering` attribute specifies whether or not the list style uses consecutive numbering for all list levels or whether each list level restarts the numbering.

XML Code:	`text:consecutive-numbering`
Rules:	This attribute can have a value of `true` or `false`.
DTD:	`<!ATTLIST text:list-style text:consecutive-numbering %boolean; "false">`

3.5.5 Number Level Style

A number level style specifies a list style where the list items are preceded by numbers.

XML Code:	`<text:list-level-style-number>`
Rules:	This element is contained in list style elements (`<text:list-style>`) only.
DTD:	`<!ELEMENT text:list-level-style-number (style:properties?)>`

The attributes that you can associate with the `<text:list-level-style-number>` element are:

- Level
- Start indent
- Minimum label width
- Minimum label distance
- Label alignment
- Text style
- Number format
- Display levels
- Start value

Level

The `text:level` attribute specifies the level of the number list style.

XML Code:	`text:level`
Rules:	The value of this attribute is a number. The number of the highest level is "1".
DTD:	`<!ATTLIST text:list-level-style-numbering text:level %number; #REQUIRED>`

Start Indent

The `text:space-before` attribute specifies the space to include before the number for all paragraphs at this level. If a paragraph has a left margin that is greater than 0, the actual position of the list label box is the left margin width plus the start indent value.

XML Code:	`text:space-before`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element. The value of the attribute is an absolute value. This means that when the position of a label is calculated the start indent value of the current level is only considered. The start indent values for lower levels do not affect the label position.
DTD:	`<!ATTLIST style:properties text:space-before %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Minimum Label Width

The `text:min-label-width` attribute specifies the minimum width of a number.

XML Code:	`text:min-label-width`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element. You can align the label horizontally with the width using an `fo:text-align` property. See the Label Alignment attribute below for more information.
DTD:	`<!ATTLIST style:properties text:min-label-width %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Minimum Label Distance

The `text:min-label-distance` attribute specifies the minimum distance between the number and the text of the list item.

XML Code:	`text:min-label-distance`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.
DTD:	`<!ATTLIST style:properties text:min-label-distance %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Label Alignment

The `fo:text-align` attribute specifies the horizontal alignment of a label (number) within the width specified by the `text:min-label-width` attribute.

XML Code:	`fo:text-align`
Rules:	This attribute is associated with the `text:min-label-width` attribute, which can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.
DTD:	See Section 3.12.4 for a DTD and more information.
Note:	This attribute does not conform with XSL or CSS specifications.

Text Style

The `text:style-name` attribute specifies the name of the style to use to format the number of the list.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:style-name %style-name #IMPLIED>`

Number Format

See Section 2.9 for detailed information on number format attributes. The attributes described in Section 2.9 can also be associated with the `<text:list-level-style-numbering>` element.

DTD:	`<!ATTLIST text:list-level-style-numbering style:num-format CDATA #REQUIRED>` `<!ATTLIST text:list-level-style-numbering style:num-prefix CDATA #IMPLIED>` `<!ATTLIST text:list-level-style-numbering style:num-suffix CDATA #IMPLIED>` `<!ATTLIST text:list-level-style-numbering style:num-letter-sync %boolean; "false">`
Note:	The `style:num-format` attribute can be empty.

Display Levels

The `text:display-levels` attribute specifies the number of levels whose numbers are displayed at the current level. For example, it could specify that you display all three numbers (1.2.1) for a level three heading or that you only display two levels (2.1).

XML Code:	`text:display-levels`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:display-levels %number; "1">`

Start Value

The `text:start-value` attribute specifies the number to display before the list item.

XML Code:	`text:start-value`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:start-value % number; "1">`

3.5.6 Bullet Level Style

A bullet level style element specifies a list style where the list items are preceded by bullets.

XML Code:	`<text:list-level-style-bullet>`
Rules:	This element is contained in list style elements (`<text:list-style>`) only.
DTD:	`<!ELEMENT text:list-level-style-bullet (style:properties?)>`
Note:	The result of including this element in an ordered list is undefined. It can be either an ordered list using a default style or an unordered list using the bullet level style.

The attributes that you can associate with the `<text:list-level-style-bullet>` element are:

- Level, spacing, alignment, and text style
- Font
- Bullet character
- Bullet relative size

Level, Spacing, Alignment, and Text Style

These attributes are the same as those described for use with the number level style, see Section 3.5.5.

Font

The font attributes that can be attached to an item set element are described in Sections 3.11.8 to 3.11.12.

Bullet Character

The bullet character attribute specifies the UNICODE character to use as the bullet in a bullet level style.

XML Code:	`text:bullet-char`
Rules:	The value of this attribute must be *one* UNICODE character.
DTD:	`<!ENTITY % char "CDATA">` `<!ATTLIST text:list-level-style-bullet text:bullet-char %char; #REQUIRED>`

Bullet Relative Size

The `text:bullet-relative-size` attribute specifies a percentage value for the bullet size relative to the font size of the paragraphs in the bullet list. For example, if the value of the `text:bullet-relative-size` attribute is `75`, the bullet used in the list is 75% of the font size for the paragraph.

XML Code:	`text:bullet-relative-size`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-bullet text:bullet-relative-size %percentage; #IMPLIED>`

3.5.7 Image Level Style

An image level style element specifies a list style where the list items are preceded by images.

XML Code:	`<text:list-level-style-image>`
Rules:	This element is an XLink and can only be contained in list style elements.
DTD:	`<!ELEMENT text:list-level-style-image (style:properties?)>` `<!ATTLIST text:list-level-style-image xlink:type (simple) #FIXED "simple">` `<!ATTLIST text:list-level-style-image xlink:show (embed) "embed">` `<!ATTLIST text:list-level-style-image xlink:actuate (onLoad) "onLoad">`
Note:	The result of including this element in an ordered list is undefined. It can be either an ordered list using a default style or an unordered list using the image level style.

The attributes that you can associate with the `<text:list-level-style-image>` element are:

- Level, spacing, and alignment
- Image location
- Image size
- Vertical alignment

Level, Spacing, and Alignment

These attributes are the same as those described for use with the number level style, see Section 3.5.5.

Image Location

The image location is stored in an XLink `href` attribute.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-image xlink:href %url; #REQUIRED>`
Note:	Bullet images can be embedded. Currently, embedded images must be stored in separate files.

Image Size

The size of the image is specified by `fo:width` and `fo:height` attributes that are attached to a `<style:properties>` element that is contained in the `<text:list-level-style-image>` element. See Section

2.6.9 for more information.

Vertical Alignment

The vertical alignment of the image is specified by the `style:vertical-pos` and `style:vertical-rel` attributes that are attached to a `<style:properties>` element that is contained in the `<text:list-level-style-image>` element. See Section 2.6.9 for more information.

Example: Image level style

```
<text:list-style style:name="List 1">
  <text:list-level-style-numbering text:level="1"
    fo:num-format="1"/>
  <text:list-level-style-bullet text:level="2"
    text:bullet-char="-"
    text:style-name="Bullet Char"/>
  <text:list-level-style-image text:level="3" xlink:href="bullet.gif">
    <style:properties fo:width=".7cm" fo:height=".7cm"
                      style:vertical-pos="middle" style:vertical-
rel="line"/>
  </text:list-level-style-image>
</text:list-style>
```

The following is the output from the above example:

```
1.  This is the first list item.

    This is a continuation of the first list item.

2.  This is the second list item. It contains an unordered sub list.

    -   This is a sub list item.

    -   This is a sub list item.

    -   This is a sub list item.

3.  This is the third list item.
```

3.6 Outline Numbering

Outline numbering is linked to paragraph styles.

3.6.1 Outline Style

XML Code:	`<text:outline-style`
Rules:	This element contains elements that specify the style of each outline level. The way StarOffice XML represents outline numbering styles is very similar to the way it represents list styles.
DTD:	`<!ELEMENT text:outline-style (text:outline-level-style)+>`

3.6.2 Outline Level Style

XML Code:	`<text:outline-level-style>`
Rules:	This element is contained in outline numbering style elements only.
DTD:	`<!ELEMENT text:outline-level-style EMPTY>`

The attributes that you can associate with the `<text:outline-level-style>` element are:

- Level
- Spacing and alignment
- Text style
- Number format
- Display levels
- Start value

Level

See Section 3.5.5 for a description of this attribute.

Spacing and Alignment

The `<text:outline-level-style>` element contains a `<style:properties>` element that can contain attributes specifying the spacing and alignment for the outline numbering list. The attributes are the same as the attributes for the numbering level style element, `<text:list-level-style-numbering>`, as follows:

- Start indent (`text:space-before`)
- Minimum label width (`text:min-label-width`)
- Minimum label distance (`text:min-label-distance`)
- Label alignment (`fo:text-align`)

See Section 3.5.5 for detailed information on these attributes.

Text Style

See Section 3.5.5 for information on the text style attribute.

Number Format

See Section 2.9 for detailed information on number format attributes. The attributes described in Section 2.9 can also be associated with the `<text:outline-numbering-level-style>` element.

DTD:	`<!ATTLIST text:outline-level-style style:num-format CDATA #REQUIRED>` `<!ATTLIST text:outline-level-style style:num-prefix CDATA #IMPLIED>` `<!ATTLIST text:outline-level-style style:num-suffix CDATA #IMPLIED>` `<!ATTLIST text:outline-level-style style:num-letter-sync %boolean; "false">`
Note:	The `style:num-format` attribute can be empty.

Display Levels

See Section 3.5.5 for information on the display level element.

Start Value

See Section 3.5.5 for information on the start value attribute.

3.7 Frames in Text Documents

A frame anchor consists of the following two parts:

- Anchor type
  The anchor type specifies how a frame is bound to the text document.
- Anchor position
  The anchor position is the point at which a frame is bound to a text document. For example, if a frame is bound to a page, the anchor position is the page number.

3.7.1 Anchor Type

The anchor type attribute specifies how a frame is bound to the text document.

XML Code:	`text:anchor-type`
Rules:	This attribute has to be attached to every frame element, for example, every `<text:text-box>` that is contained in a text document. It can also be attached to graphic styles, in this case it specifies the default anchor type for every frame that is inserted into a document using the graphic style.
DTD:	`<!ATTLIST style:properties text:anchor-type (page|frame|paragraph|char|as-char) #IMPLIED>`

3.7.2 Anchor Position

The anchor position is the point at which a frame is bound to a text document. The anchor position depends on the anchor type as explained in the following table.

If the value of the `text:anchor-type` attribute is ...	The anchor position is...	The frame element appears ...	Notes
page	The page that has the same physical page number as the value of the `text:anchor-page-number` attribute that is attached to the frame element.	At the start of the document body, outside any paragraph or frame.	The physical page number is the number assigned to the page if all pages in the document are counted starting with one that is the physical page number of the first page of the document. ???
frame	The parent frame that the current frame element is contained in.	In the element representing the frame to which the frame is bound. For example, if an image is bound to a text box, the frame element is located in the text box element.	Currently, frames can only be bound to text boxes.
paragraph	The paragraph that the current frame element is contained in.	At the start of the paragraph element.	
char	The character after the frame element.	Just before the character.	
as-char	There is no anchor position. The frame behaves like a character.	At the position where the character appears in the document.	

Horizontal and Vertical Alignment

The following tables display the possible values of the attributes `style:horizontal-pos`, `style:horizontal-rel`, `style:vertical-pos`, and `style:vertical-rel`, depending on the anchor type of the frame. The possible values of these alignment attributes are listed in the first column on the left, and an alignment attribute value/anchor type value match is indicated by an X.

Value of `style:horizontal-pos`	Value of `text:anchor-type`				
	page	frame	paragraph	char	as-char
any	X	X	X	X	

Value of `style:horizontal-rel`	Value of `text:anchor-type`				
	page	frame	paragraph	char	as-char
page	X		X	X	
page-content	X		X	X	
page-start-margin	X		X	X	
page-end-margin	X		X	X	
frame		X			
frame-content		X			
frame-start-margin		X			
frame-end-margin		X			

Value of `style:horizontal-rel`	**Value of `text:anchor-type`**				
	page	**frame**	**paragraph**	**char**	**as-char**
paragraph			X	X	
paragraph-content			X	X	
paragraph-start-margin			X	X	
paragraph-end-margin			X	X	
char				X	

Value of `style:vertical-pos`	**Value of `text:anchor-type`**				
	page	**frame**	**paragraph**	**char**	**as-char**
any	X	X	X	X	X

Value of `style:vertical-rel`	**Value of `text:anchor-type`**				
	page	**frame**	**paragraph**	**char**	**as-char**
page	X				
page-content	X				
frame		X			
frame-content		X			
paragraph			X	X	
paragraph-content			X	X	
char				X	X
line					X
baseline					X

Note: XSL and HTML support very few of the combinations of anchor type, vertical/horizontal alignment, and wrap mode that StarOffice supports.

3.7.3 Anchor Page Number

The `text:anchor-page-number` attribute specifies the physical page number of an anchor if the frame is bound to a page.

XML Code:	`text:anchor-page-number`
Rules:	
DTD:	`<!ATTLIST style:properties text:anchor-page-number %number; #IMPLIED>`

3.8 Footnotes and Endnotes

3.8.1 Footnotes Configuration

A StarOffice document contains either *none* or *one* footnotes configuration element. If there is no footnote configuration element, a default footnote configuration is used. Therefore, every saved StarOffice document contains a footnote configuration element.

XML Code:	`<text:footnotes-configuration>`
Rules:	
DTD:	`<!ELEMENT text:footnotes-configuration (text:footnote-continuation-notice-forward?, text:footnote-continuation-notice-backward?)>`

The attributes that you can associate with the `<text:footnotes-configuration>` element are:

- Citation text style
- Citation body text style
- Default footnote paragraph style
- Master page
- Offset
- Number format
- Numbering scheme
- Footnote position

You can include the following element in the `<text:footnotes-configuration>` element:

- Footnote continuation notice (forward and backward)

Citation Text Style

The citation text style specifies the text style to use for the footnote citation within the footnote.

XML Code:	`text:citation-style`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:citation-style CDATA #IMPLIED>`

Citation Body Text Style

The citation body text style specifies the text style to use for the footnote citation in the text flow.

XML Code:	`text:citation-body-style-name`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:citation-body-style-name %styleName; #IMPLIED>`

Default Footnote Paragraph Style

The default footnote paragraph style is only used for footnotes that are inserted into an existing document. It is not used for footnotes that already exist.

XML Code:	`text:default-style`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:default-style CDATA #IMPLIED>`

Master Page

If the footnotes in a document should be displayed at the end of the document, the pages that contain the footnotes are instances of this master page.

XML Code:	`text:master-page-name`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:master-page-name %styleName; #IMPLIED>`

Offset

The `text:offset` attribute specifies an offset value that is added to every footnote number. The offset is between the position of the footnote number and the footnote text.

XML Code:	`text:offset`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:offset %number; "0">`

Number Format

See Section 2.9 for information on the number format for footnotes.

Numbering Scheme

The `text:start-numbering-at` attribute specifies if footnote numbers start with a new number at the beginning of the document or at the beginning of each chapter or page.

XML Code:	`text:start-numbering-at`
Rules:	The value of this attribute can be `document`, `chapter`, or `page`.
DTD:	`<!ATTLIST text:footnotes-configuration text:start-numbering-at (document|chapter|page) "document">`
Note:	XSLT does not have the capability to start with new footnote numbers on every page.

Footnotes Position

The `text:footnotes-position` attribute specifies if footnotes are displayed at the bottom of the page where the footnote citation is located or at the end of the document.

XML Code:	`text:footnotes-position`
Rules:	The value of this attribute can be `page` or `document`.
DTD:	`<!ATTLIST text:footnotes-configuration text:footnotes-position (document|page) "page">`
Note:	XSL does have the capability to display footnotes at the end of the document. However, you can use an XSLT stylesheet to generate some other flow objects to display such footnotes.

Footnote Continuation

The footnote continuation elements specify:

- Text displayed at the end of a footnote that is continued on the next page
- Text displayed before the continued text

XML Code:	`<text:footnote-continuation-notice-forward>` and `<text:footnote-continuation-notice-backward>`
Rules:	These elements can be contained in the footnotes configuration element.
DTD:	`<!ELEMENT text:footnote-continuation-notice-forward (#PCDATA)>` `<!ELEMENT text:footnote-continuation-notice-backward (#PCDATA)>`
Note:	XSL and XSLT do not support footnote continuation.

Example: Footnote configuration in StarOffice XML

```
<text:footnotes-configuration text:citation-style="Footnote symbol"
                              text:default-style="Footnote">
  <text:footnote-continuation-notice-forward>" .."
  </text:footnote-continuation-notice-forward>
  <text:footnote-continuation-notice-forward>".. "
  </text:footnote-continuation-notice-forward>
</text:footnotes-configuration>
```

3.8.2 Endnotes Configuration

A StarOffice document contains either *none* or *one* endnotes configuration element.

XML Code:	`<text:endnotes-configuration>`
Rules:	
DTD:	`<!ELEMENT text:endnotes-configuration EMPTY>`

Citation Text Style, Default Endnote Paragraph Style, Page Master, Offset, and Number Format

See Section 3.8.1 for descriptions of these attributes. The application of these attributes to the endnote configuration element is the same as for the footnote configuration element.

DTD:

```
<!ATTLIST text:endnotes-configuration
          text:citation-body-style-name %styleName; #IMPLIED
          text:citation-style-name %string; #IMPLIED
          text:default-style-name %styleName; #IMPLIED
          text:page-master-name %styleName; #IMPLIED
          text:offset %number; "0"
          style:num-format %string; #IMPLIED
          text:num-prefix %string; #IMPLIED
          text:num-suffix %string; #IMPLIED>
```

3.8.3 Footnotes

The footnote element contains the footnote citation element and the elements that make up the footnote content.

XML Code:	`<text:footnote>`
Rules:	
DTD:	`<!ELEMENT text:footnote (text:footnote-citation, text:footnote-body)>`
Note:	StarOffice XML represents footnotes in a similar fashion to XSL. In XSL, the first child of the footnote element contains the citation in the form of an `<fo:inline>` element. StarOffice XML uses the same structure but introduces a `text:footnote-citation` element. The second child contains the footnote body, just as in XSL. Additionally, StarOffice features the `<text:footnotes-configuration>` element. To achieve a similar effect to the footnote configuration in XSL, every footnote and footnote citation element must be appropriately formatted.

Footnote Citation

The footnote citation element specifies the formatted footnote number or characters.

XML Code:	`<text:footnote-citation>`
Rules:	This element is contained in the footnote element (`<text:footnote>`) and it contains the formatted footnote number as text.
DTD:	`<!ELEMENT text:footnote-citation (#PCDATA)>`

Footnote Label

Footnote citation elements can be labeled or numbered. If they are numbered, the number is chosen automatically according to the footnotes configuration element. If they are labeled, the user must supply a label for every footnote he/she inserts into the document. This label is stored in the `text:label` attribute of the `<text:footnote-citation>` element.

XML Code:	`text:label`
Rules:	If this attribute is not present, the footnote is numbered automatically.
DTD:	`<!ATTLIST text:footnote-citation text:label %string; #IMPLIED>`

Footnote Reference ID

The footnote reference ID is used by references to footnotes to identify the footnote that is referenced.

XML Code:	`text:id`
Rules:	This attribute is used by the `<text:footnote>` element. It contains a value of type `ID`, where `ID` is a predefined XML attribute type.
DTD:	`<!ATTLIST text:footnote text:id ID #IMPLIED>`

Footnote Body

This element contains the actual content of the footnote. It does not have any attributes.

XML Code:	`<text:footnote-body>`
Rules:	
DTD:	`<!ELEMENT text:footnote-body %inline-text;>`

Examples: Footnotes

```
<text:p>
  This paragraph contains a footnote
  <text:footnote text:id="ftn001">
    <text:footnote-citation>
      1
    </text:footnote-citation>
    <text:footnote-body>
      <text:p>
        This footnote has a generated sequence number
      </text:p>
    </text:footnote-body>
  </text:footnote>
  .
</text:p>
<text:p>
  This paragraph contains a footnote
  <text:footnote text:id="ftn002">
    <text:footnote-citation text:label="*">
      *
    </text:footnote-citation>
    <text:footnote-body>
      <text:p>
        This footnote has a fixed citation
      </text:p>
    </text:footnote-body>
  </text:footnote>
  , too
</text:p>
```

3.8.4 Endnotes

Endnotes are represented in the same way as footnotes. They contain the endnote citation element and the endnote body element that makes up the endnote content. For a full description of the elements and attributes associated with the `<text:endnote>` element, please refer to the previous section.

XML Code:	`<text:endnote>`
Rules:	

DTD:	`<!ELEMENT text:endnote (text:endnote-citation, text:endnote-body)>`
	`<!ATTLIST text:endnote text:id ID #IMPLIED>`

Limitations:	XSL does not support endnotes but you can use an XSLT stylesheet to generate some other flow objects to display endnotes.

Endnote Citation

XML Code:	`<text:endnote-citation>`
Rules:	

DTD:	`<!ELEMENT text:endnote-citation (#PCDATA)>`
	`<!ATTLIST text:endnote-citation text:label %string; #IMPLIED>`

Endnote Body

The `<text:endnote-body>` element is defined as follows:

XML Code:	`<text:endnote-body>`
Rules:	
DTD:	`<!ELEMENT text:endnote-body %inline-text;>`

3.9 Ruby

Ruby is additional text that is displayed above or below some base text. It's purpose is to annotate the base text or give information about its pronunciation.

XML Code:	`<text:ruby>`
Rules:	This element can be contained anywhere within a paragraph. It contains to sub elements, one for the base and one for the ruby text.
DTD:	`<!ELEMENT text:ruby (text:ruby-base,text:ruby-text)>`

The attributes that you can associate with the `<text:ruby>` element are:

- Ruby style
- Ruby text formatting properties

There are two elements that can be contained in the `<text:ruby>` element:

- Ruby base
- Ruby text

Ruby Style

A ruby style specifies how the ruby text is displayed relative to the base text. It is represented by a `<style:style>` element those family is `ruby`. The ruby style is assigned to the ruby element using a `text:style-name` attribute.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:ruby text:style-name %styleName; #IMPLIED>`

Ruby Text Formatting Properties

All of the ruby text is displayed using the same formatting properties. The `style:style-name` attribute is used to specify these properties.

XML Code:	`style:style-name`
Rules:	This attribute references a text style that is used to display the ruby text.
DTD:	`<!ATTLIST text:ruby-text %styleName;>`

Ruby Position

This property specifies the position of the ruby text relative to the ruby base.

XML Code:	`style:ruby-position`
Rules:	The value of this property can be `above` or `below`.
DTD:	`<!ATTLIST style:properties style:ruby-position (above|below) #IMPLIED>`

Ruby Alignment

This property specifies the alignment of the ruby text relative to the ruby base.

XML Code:	`style:rub-align`
Rules:	The value of this property can be `left`, `center`, `right`, `distribute-letter`, or `distribute-space`.
DTD:	`<!ATTLIST style:properties style:ruby-align (left|center|right|distribute-letter|distribute-space) #IMPLIED>`

3.9.1 Ruby Base

The `<text:ruby-base>` element contains the text that is to be annotated.

XML Code:	`<text:ruby-base >`
Rules:	This element can contain any content.
DTD:	`<!ELEMENT text:ruby-base (%inline-text;|#PCDATA)>`

3.9.2 Ruby Text

The `<text:ruby-text >` element contains the annotation.

XML Code:	`<text:ruby-text >`
Rules:	
DTD:	`<!ELEMENT text:ruby-text (#PCDATA)>`

3.10 Line Numbering

3.10.1 Line Numbering Configuration

A StarOffice document can contain *none* or *one* line numbering configuration element. If the element is not present, a default line numbering configuration is used. The default line numbering may vary depending on the version of StarOffice software but every document saved using the StarOffice software contains a line numbering configuration element.

XML Code:	`<text:linenumbering-configuration>`
Rules:	
DTD:	`<!ELEMENT text:linenumbering-configuration (text: linenumbering-seperator?)>`

The attributes that you can associate with the `<text:linenumbering-configuration>` element are:

- Line numbering enable
- Number format
- Text style
- Increment
- Position
- Offset
- Count empty lines
- Count in floating frames
- Restart numbering on every page

The element that you can associate with the `<text:linenumbering-seperator>` element is:

- Separator and its associated separator offset attribute

Line Numbering Enable

This attribute controls whether or not lines are numbered.

XML Code:	`text:number-lines`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:number-lines % boolean; "true">`

Number Format

See Section 2.9 for detailed information on number formats.

Text Style

The `text:style-name` attribute specifies the text style for all line numbers.

XML Code:	`text:style-name`
Rules:	The value of this attribute is the name of the text style that is applied to all line numbers.
DTD:	`<!ATTLIST text:linenumbering-configuration text:style-name CDATA #IMPLIED>`

Increment

The `text:increment` attribute causes line numbers that are a multiple of the given increment to be numbered. For example, if the increment is 5, only lines number 5, 10, 15, and so on are numbered.

XML Code:	`text:increment`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:increment CDATA #IMPLIED>`

Position

The `text:position` attribute determines whether the line numbers are printed on the left , right, inner, or outer margins.

XML Code:	`text:position`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:position (left|right|inner|outer) "left">`

Offset

The `text:offset` attribute determines the distance between the line number and the margin.

XML Code:	`text:offset`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:offset %nonNegativeLength; #IMPLIED>`

Count Empty Lines

If the value of this attribute is `true`, empty lines are included in the line count.

XML Code:	`text:count-empty-lines`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:count-empty-lines %boolean; "true">`

Count Lines in Floating Frames

If the value of this attribute is `true`, text within floating frames is included in the line count.

XML Code:	`text:count-in-floating-frames`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:count-in-floating-frames %boolean; "false">`

Restart Numbering on Every Page

If the value of this attribute is `true`, the line count is reset to 1 at the beginning of every page, resulting in page - specific numbering of lines. The default value of this attribute is `false`, resulting in document-specific numbering of lines.

XML Code:	`text:restart-on-page`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:restart-on-page %boolean; "false">`

3.10.2 Separator

The `<text:linenumbering-seperator>` element contains the text that is displayed as a separator.

XML Code:	`<text:linenumbering-seperator>`
Rules:	This element is contained in the line numbering configuration element. If the element is not present, no separator is displayed.
DTD:	`<!ELEMENT text:linenumbering-separator (#PCDATA)>`

Separator Offset Attribute

The `text:increment` attribute specifies the separator offset.

XML Code:	`text:increment`
Rules:	This attribute is associated with the line numbering separator element.
DTD:	`<!ATTLIST text:linenumbering-separator text:increment %number; #REQUIRED>`

3.10.3 Line Numbering Properties

Some of the text formatting properties that you can apply to paragraphs and paragraph styles also influence line numbering. These text formatting properties are as follows:

- Line numbering application
- Line number start value

Line Numbering Application

This property controls whether or not paragraph lines are numbered.

XML Code:	`text:number-lines`
Rules:	This attribute can be contained in an item set element that belongs to a paragraph or paragraph style.
DTD:	`<!ATTLIST style:properties text:number-lines %boolean; #IMPLIED>`

Line Number Start Value

This property specifies a new start value for line numbering.

XML Code:	`text:line-number`
Rules:	This attribute can be contained in an item set element that belongs to a paragraph or paragraph style. The attribute is only recognized if there is also a `text:number-lines` attribute with a value of `true` in the same item set element.
DTD:	`<!ATTLIST style:properties text:line-number %number; #IMPLIED>`

3.11 Text Formatting Properties

You can apply text formatting properties to text portions, paragraphs, and paragraph styles.

3.11.1 Font Variant

This property switches the option to display text as small capitalized letters on or off.

XML Code:	`fo:font-variant` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:font-variant (normal|small-caps) #IMPLIED>`
Implementation limitation:	At present, the `fo:font-variant` and `fo:text-transform` properties are mutually exclusive. If both properties are attached to an item set element simultaneously, the result is undefined except that the `fo:text-transform` value is `none` and the `fo:font-variant` value is `normal`.

3.11.2 Text Transformations

This property describes text transformations to uppercase, lowercase, and capitalization.

XML Code:	`fo:text-transform` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:text-transform (none|lowercase|uppercase|capitalize) #IMPLIED>`
Implementation limitation:	At present, the `fo:font-variant` and `fo:text-transform` properties are mutually exclusive. If both properties are attached to an item set element simultaneously, the result is undefined except that the `fo:text-transform` value is `none` and the `fo:font-variant` value is `normal`.

3.11.3 Color

Use this property to specify the foreground color of text.

XML Code:	`fo:color` (XSL property)
Rules:	

DTD:	`<!ENTITY % color "CDATA">`
	`<!ATTLIST style:properties fo:color %color; #IMPLIED>`

3.11.4 Text Outline

Use this property to specify whether to display an outline of text or the text itself.

XML Code:	`style:text-outline`
Rules:	This attribute can have a value of `true` or `false`.

DTD:	`<!ATTLIST style:properties style:text-outline %boolean;`
	`#IMPLIED>`

Note:	XSL does not have a corresponding property.

3.11.5 Crossing Out

Use this property to specify the style to use when crossing out text.

XML Code:	`style:text-crossing-out`
Rules:	The value of this attribute is the crossing out style for the text.

DTD:	`<!ATTLIST style:properties style:text-crossing-out`
	`          (none|single-line|double-line|thick-line|slash|X)`
	`#IMPLIED>`

Note:	XSL does not support this property, but the values `none` and `single-line` correspond to the values `none` and `underline` for the XSL `fo:text-decoration` property.

3.11.6 Text Position

Use this formatting property to specify whether text is positioned above or below the baseline and to specify the relative font height that is used for this text.

XML Code:	`style:text-position`
Rules:	This attribute can have one or two values. The first value must be present and specifies the vertical text position as a percentage that relates to the current font height or it takes one of the values `sub` or `super`. Negative percentages or the `sub` value place the text below the baseline. Positive percentages or the `super` value place the text above the baseline. If `sub` or `super` is specified, the application can choose an appropriate text position. The second value is optional and specifies the font height as a percentage that relates to the current font-height. If this value is not specified, an appropriate font height is used. Although this value may change the font height that is displayed, it never changes the current font height that is used for additional calculations.
DTD:	`<!ATTLIST style:properties style:text-position CDATA #IMPLIED>`
Note:	The effect of using this property is the same as the effect achieved by using the XSL properties `fo:vertical-align` and `fo:font-size`. This representation is not appropriate because the `fo:font-size` property is used to change the font height without changing its position.

3.11.7 Font Name

Use these properties to assign a font to the text.

XML Code:	`style:font-name`
	`style:font-name-asian`
	`style:font-name-complex`

Rules:	The values of these attributes form the name of a font that is declared by a `<style:font-decl>` element within the `<office:font-decls>` element. The `style:font-name-asian` attribute is evaluated for UNICODE characters that are CJK characters. The `style:font-name-complex` attribute is evaluated for UNICODE characters that are complex text layout (CTL) characters. The `style:font-name` attribute is evaluated for any other UNICODE character.

DTD:	`<!ATTLIST style:properties fo:font-name %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-name-asian %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-name-complex %string; #IMPLIED>`

3.11.8 Font Family

Use these properties to specify the font family for the text.

XML Code:	`fo:font-family` (XSL property)
	`style:font-family-asian`
	`style:font-family-complex`

Rules:	You can use these properties instead of the `style:font-name` attributes to specify the properties of a font individually. However, it is advisable touse the `style:font-name` attributes instead. See Section 3.11.7 for information about when asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-family %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-family-asian %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-family-complex %string; #IMPLIED>`

3.11.9 Font Family Generic

Use these properties to specify a generic font family name.

XML Code:	`style:font-family-generic`
	`style:font-family-generic-asian`
	
	`style:font-family-generic-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontFamilyGeneric "(roman|swiss|modern|decorative|script|system)">`
	`<!ATTLIST style:properties style:font-family-generic %fontFamilyGeneric; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-family-generic-asian %fontFamilyGeneric; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-family-generic-complex %fontFamilyGeneric; #IMPLIED>`

3.11.10 Font Style

Use these properties to specify a font style name.

XML Code:	`style:font-style-name`
	`style:font-style-name-asian`
	`style:font-style-name-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties style:font-style-name %string; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-style-name-asian %string; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-style-name-complex %string; #IMPLIED>`

Note:	XSL does not support this property.

3.11.11 Font Pitch

Use these properties to specify whether a font has a fixed or variable width.

XML Code:	`style:font-pitch`
	`style:font-pitchgv`
	`style:font-pitch-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontPitch "(fixed|variable)">`
	`<!ATTLIST style:properties style:font-pitch %fontPitch; #IMPLIED>`
	`<!ATTLIST style:properties style:font-pitch-asian %fontPitch; #IMPLIED>`
	`<!ATTLIST style:properties style:font-pitch-complex %fontPitch; #IMPLIED>`

Note:	XSL does not support this property.

3.11.12 Font Character Set

Use these properties to specify the character set of a font.

XML Code:	`style:font-charset`
	`style:font-charset-asian`
	`style:font-charset-complex`

Rules:	The value of these attributes can be `x-symbol` or the character encoding in the notation described in the XML recommendation (Chapter 4.3.3, Character Encoding and Entities, http://www.w3.org/TR/REC-xml#charencoding). If the value is `x-symbol`, all characters that are displayed using this font must be contained in the UNICODE character range 0xf000 to 0xf0ff. These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties style:font-charset CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:font-charset-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:font-charset-complex CDATA #IMPLIED>`

3.11.13 Font Size

Use these properties to specify the size of font.

XML Code:	`fo:font-size` (XSL property)
	`fo:font-size-asian`
	`fo:font-size-complex`

Rules:	The value of these property is either an absolute length or a percentage. In contrast to XSL, percentage values can be used within styles only and relate to the font height of the parent style rather than to the font height of the attributes neighborhood. Absolute font heights such as `medium`, `large`, `x-large`, and so on, and relative font heights such as `smaller`, and `larger` are not supported. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % length_or_percentage "CDATA">`
	`<!ATTLIST style:properties fo:font-size %length_or_percentage; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-asian %length_or_percentage; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-complex %length_or_percentage; #IMPLIED>`

3.11.14 Relative Font Size

Use these properties to specify a relative font size change.

XML Code:	`style:font-size-rel`
	`style:font-size-rel-asian`
	`style:font-size-rel-complex`

Rules:	These properties specify a relative font size change as a length such as `+1pt`, `-3pt`. It cannot be used within automatic styles. The size changes relates to the font size setting that applies to the parent style of the style. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-size-rel %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-rel-asian %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-rel-complex %length; #IMPLIED>`

3.11.15 Letter Spacing

Use this property to specify the amount of space between letters.

XML Code:	`fo:letter-spacing` (XSL property)
Rules:	The value of this property can be `normal` or it can specify a length.
DTD:	`<!ATTLIST style:properties fo:letter-spacing CDATA #IMPLIED>`

3.11.16 Language

Use this property to specify the language of the text.

XML Code:	`fo:language` (XSL property)
	`fo:language-asian`
	`fo:language-complex`

Rules:	The value of these properties can be any of the ISO 639 language codes (see http://www.oasis-open.org/cover/iso639a.html). At present, these properties are ignored if they are not specified together with the corresponding `fo:country` property. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:language CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:language-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:language-complex CDATA #IMPLIED>`

3.11.17 Country

XML Code:	`fo:country` (XSL property)
	`fo:country-asian`
	`fo:country-complex`

Rules:	The value of these properties can be any of the ISO 3166 country codes (see http://www.sil.org/acpub/catalog/country.html). At present, these properties are ignored if they are not specified together with the corresponding `fo:language` property. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:country CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:country-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:country-complex CDATA #IMPLIED>`

3.11.18 Font Style

Use these properties to specify whether to use a normal or italic font face.

XML Code:	`fo:font-style` (XSL property)
	`fo:font-style-asian`
	`fo:font-style-complex`

Rules:	See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontStyle "(normal|italic|oblique)">`
	`<!ATTLIST style:properties fo:font-style %fontStyle; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-style-asian %fontStyle; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-style-complex %fontStyle; #IMPLIED>`

3.11.19 Text Shadow

Use this property to specify the text shadow style to use.

XML Code:	`fo:text-shadow` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:text-shadow CDATA #IMPLIED>`
Implementation limitation:	At present, StarOffice only supports a default text shadow style. Therefore, any value other than `none` switches on this default shadow style.

3.11.20 Underlining

Use this property to specify if text is underlined.

XML Code:	`style:text-underline`
Rules:	The value of this property is the underlining style for the text, for example, `single`, `dotted`, `dash`.
DTD:	`<!ATTLIST style:properties fo:text-underline (none|single|double|dotted|dash|long-dash|dot-dash|dot-dot-dash|wave|bold|bold-dotted|bold-dash|bold-long-dash|bold-dot-dash|bold-dot-dot-dash|bold-wave|double-wave|small-wave) #IMPLIED>`
Note:	XSL does not support this property but the values `none` and `single` correspond to the values `none` and `underline` in the XSL `fo:text-decoration` property. The `fo:text-decoration` property is also used for crossing out and blinking text.

3.11.21 Underline Color

Use this property to specify the color that is used to underline text.

XML Code:	`style:text-underline-color`
Rules:	The value of this property is either `font-color` or a color. If the value is `font-color`, the current text color is used for underlining.
DTD:	`<!ATTLIST style:properties fo:text-underline-color CDATA #IMPLIED>`
Note:	If you set this property within StarOffice without specifying a `style:text-underline` property for the same style, underlining is switched off.

3.11.22 Font Weight

Use these properties to specify the weight of the font.

XML Code:	`fo:font-weight` (XSL property)
	`fo:font-weight-asian`
	`fo:font-weight-complex`

Rules:	The relative values `lighter` or `bolder` are not supported and only a few distinct numerical values are supported. Unsupported numerical values are rounded off to the next supported value. See Section 3.11.7 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-weight CDATA #IMPLIED>`
	
	`<!ATTLIST style:properties fo:font-weight-asian CDATA #IMPLIED>`
	
	`<!ATTLIST style:properties fo:font-weight-complex CDATA #IMPLIED>`

3.11.23 Text Decoration Word Mode

Use this property to specify whether crossing out and underlining is applied to words only or to text-portions. If crossing out and underlining is applied to text portions, the space between words as well as the word itself is underlined or crossed out.

XML Code:	`fo:score-spaces`
Rules:	
DTD:	`<!ATTLIST style:properties fo:score-spaces %boolean; #IMPLIED>`
Notes:	XSL does not support this property. In StarOffice 5.2, this property was called `style:decorate-words-only`.

3.11.24 Letter Kerning

Use this property to enable or disable kerning between characters.

XML Code:	`style:letter-kerning`
Rules:	
DTD:	`<!ATTLIST style:properties style:letter-kerning %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.11.25 Text Blinking

Use this property to specify whether or not text should blink.

XML Code:	`style:text:blinking`
Rules:	
DTD:	`<!ATTLIST style:properties style:text-blinking %boolean; #IMPLIED>`
Note:	XSL does not support this property but the values `false` and `true` correspond to the values `none` and `blink` of the XSL `fo:text-decoration` property. The XSL `fo:text-decoration` can be used to represent this property but it is also used for underlined an crossing out text.

3.11.26 Text Background Color

Use this property to specify the background color to apply to characters.

XML Code:	`style:text-background-color`
Rules:	The value of this property can be `transparent` or a color. The property has the same values as the `fo:background-color` property.

DTD:	`<!ENTITY % transparent_or_color "CDATA">`
	`<!ATTLIST style:properties style:text-background-color`
	`          %transparent_or_color; #IMPLIED>`

Note:	Unlike StarOffice, XSL does not distinguish between character and paragraph backgrounds. In StarOffice, if a background is applied to a block element (paragraph), it behaves like a paragraph background and if it is applied to an inline element (a piece of text within a paragraph), it behaves like a character background. Therefore, this property can be transformed to an XSL `fo:background-color` but an additional inline element is required.

3.11.27 Text Combine

This property can be used to combine characters so that they are displayed within two lines.

XML Code:	`style:text-combine`
Rules:	The value of this attribute can be `none`, `letters` or `lines`. If the value is `lines`, all characters with this attribute value that immediately follow each other are displayed within two lines of approximately the same length. There can be a line break between any two characters to meet this constraint. If the value of the attribute is `letters`, up to 5 characters are combined within two lines. Any additional character is displayed as normal text.
DTD:	`<!ATTLIST style:properties style:text-combine (none|letters|lines)>`
Note:	See http://www.w3.org/TR/WD-i18n-format/".

3.11.28 Text Combine Start and End Characters

These two properties specify a start and end character that is displayed before and after a portion of text whose `style:text-combine` property has a value of `lines`.

XML Code:	`style:text-combine-start-char` `style:text-combine-end-char`
Rules:	

DTD:	`<!ATTLIST style:properties style:text-combine-start-char %character;>`
	`<!ATTLIST style:properties style:text-combine-end-char %character;>`

3.11.29 Text Emphasis

Use this property to emphasize text in Asian documents.

XML Code:	`style:text-emphasize`
Rules:	The value of this attribute consists of two space-separated values. The first value represents the style to use for emphasis and it can be `none`, `accent`, `dot`, `circle`, or `disc`. The second value represents the position of the emphasis and it can be `above` or `below`. If the first value is `none`, this value can be omitted.
DTD:	`<!ATTLIST style:properties style:text-emphasize CDATA #IMPLIED>`

Note:	StarOffice supports the combined values
	
	`none`
	`dot above`
	`dot below`
	`accent above`
	`circle above`
	
	When the document is imported, `disc above` is changed to `dot above`, while any `below` value is changed to a `dot below` value.

3.11.30 Text Autospace

This property specifies whether to add space between asian, western, and complex text.

XML Code:	`style:text-autospace`
Rules:	StarOffice only supports the values `none` and `ideograph-alpha`.
DTD:	`<!ATTLIST style:properties style:text-autospace (none | ideograph-alpha) #IMPLIED>`

3.11.31 Punctuation Wrap

This property determines whether or not a punctuation mark, if one is present, can be placed in the margin area at the end of a full line of text. This is a common setting in East Asian typography.

XML Code:	`style:punctuation-wrap`
Rules:	
DTD:	`<!ATTLIST style:properties style:punctuation-wrap (simple | hanging) #IMPLIED>`

3.11.32 Line Break

This property selects the set of line breaking rules to use for text.

XML Code:	`style:line-break`
Rules:	
DTD:	`<!ATTLIST style:properties style:line-break (normal | strict) #IMPLIED>`

3.12 Paragraph Formatting Properties

You can apply paragraph formatting properties to paragraphs and paragraph styles.

3.12.1 Fixed Line Height

Use this property to specify a fixed line height either as a length or a percentage that relates to the highest character in a line. A special value of `normal` activates the default line height calculation. It is also used to deactivate the effects of the `style:line-height-at-least` and `style:line-spacing` properties.

XML Code:	`fo:line-height` (XSL property)
Rules:	The value of this property can be a length, a percentage, or a value of `normal`.
DTD:	`<!ATTLIST style:properties fo:line-height (normal|%length;|%percentage;) #IMPLIED>`
Note:	The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set element is undefined. XSL supports this property but it does not support a number property value.

3.12.2 Minimum Line Height

Use this property to specify a minimum line height.

XML Code:	`style:line-height-at-least`
Rules:	The value of this property is a length. There is no `normal` value for the property.
DTD:	`<!ATTLIST style:properties style:line-height-at-least %length; #IMPLIED>`
Note:	XSL does not support this property. The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set elements is undefined.

3.12.3 Line Distance

Use this property to specify a fixed distance between two lines

XML Code:	`style:line-spacing`
Rules:	There is no `normal` value for this property.
DTD:	`<!ATTLIST style:properties style:line-spacing %length #IMPLIED>`
Note:	XSL does not support this property. The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set element is undefined.

3.12.4 Text Align

Use this property to specify how to align text in paragraphs.

XML Code:	`fo:text-align` (XSL property)
Rules:	The value of this property can be `start`, `end`, `center`, or `justify`. If there are no values specified for the `style:text-align-last` and `style:justify-single-word` properties within the same item set element, the values of these properties are set to `left` and `false` respectively.
DTD:	`<!ATTLIST style:properties fo:text-align (start|end|center|justify) #IMPLIED>`
Notes:	At present, the values `page-inside` and `page-outside` are not supported. In StarOffice 5.2, the attribute value `justify` was called `justified`.

3.12.5 Text Align of Last Line

Use this property to specify how to align the last line of a justified paragraph.

XML Code:	`style:text-align-last`
Rules:	The value of this property can be `start`, `center`, or `justify`. This property is ignored if it not accompanied by an `fo:text-align` property. If there are no values specified for the `fo:text-align` and `style:justify-single-word` properties, these values of these properties is set to `left` and `false` respectively.
DTD:	`<!ATTLIST style:properties style:text-align-last (start|center|justify) #IMPLIED>`
Note:	In StarOffice 5.2, the attribute value `justify` was called `justified`.

3.12.6 Justify Single Word

If the last line in a paragraph is justified, use this property to specify whether or not a single word should be justified.

XML Code:	`style:justify-single-word`
Rules:	If there are no values specified for the `fo:text-align` and `style:text-align-last` properties, the values of these properties are set to `left`. This means that specifying a `style:justify-single-word` property without specifying a `style:text-align` and `style:text-align-last` property has no effect.
DTD:	`<!ATTLIST style:properties style:justify-single-word %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.12.7 Break Inside

Use this property to control whether page or column breaks are allowed within a paragraph.

XML Code:	`style:break-inside`
Rules:	
DTD:	`<!ATTLIST style:properties style:break-inside (auto|avoid) #IMPLIED>`
Note:	XSL does not support this property.

3.12.8 Widows

Use this property to specify the minimum number of lines allowed at the top of a page to avoid paragraph **widows**.

XML Code:	`fo:widows` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:widows %number; #IMPLIED>`
Note:	Unlike XSL, this property affects column breaks and page breaks.

3.12.9 Orphans

Use this property to specify the minimum number of lines required at the bottom of a page to avoid paragraph **orphans**.

XML Code:	`fo:orphans` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:orphans %number; #IMPLIED>`
Note:	Unlike XSL, this property affects column breaks and page breaks.

3.12.10 Tab Stops

The tab stop elements specify tab stop definitions.

XML Code:	`<style:tab-stops>` and `<style:tab-stop>`
Rules:	Every tab stop position is represented by a single `<style:tab-stop>` element that is contained in the `<style:tab-stops>` element.

DTD:	`<!ELEMENT style:tab-stops (style:tab-stop)*>`
	`<!ELEMENT style:tab-stop EMPTY>`

Note:	XSL does not support tab stops.

The attribute that you can associate with the `<style:tab-stops>` and `<style:tab-stop>` elements are:

- Tab position
- Tab type
- Delimiter character
- Leader character

Tab Position

The `style:position` attribute specifies the position of a tab stop.

XML Code:	`style:position`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value is a length.
DTD:	`<!ATTLIST style:tab-stop style:position %length; #REQUIRED>`

Tab Type

The `style:type` attribute specifies the type of tab stop.

XML Code:	`style:type`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value can be `left`, `center`, `right`, or `char`.
DTD:	`<!ATTLIST style:tabtype style:type (left|center|right|char) "left">`

Delimiter Character

The `style:char` attribute specifies the delimiter character for tab stops of type `char`.

XML Code:	`style:char`
Rules:	This attribute is associated with the `<style:tab-stop>` element and it *must* be present if the value of the `style:type` attribute is `char`. If the value of `style:type` attribute is not `char,` it is ignored. The value of the attribute must be a single UNICODE character.
DTD:	`<!ATTLIST style:tab-stop style:char %char; #IMPLIED>`

Leader Character

The `style:leader-char` attribute specifies the leader character to use for tab stops.

XML Code:	`style:leader-char`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value must be a single UNICODE character.
DTD:	`<!ATTLIST style:tab-stop style:leader-char %char; " ">`

3.12.11 Hyphenation

Use this property to enable or disable automatic hyphenation.

XML Code:	`fo:hyphenate`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenate %boolean; #IMPLIED>`
Implementation limitation:	At present, if you enable hyphenation, StarOffice XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.12.12 Hyphenation Keep

Use this property to enable or disable the hyphenation of the last word on a page.

XML Code:	`fo:hyphenation-keep`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-keep (none|page|column|spread) #IMPLIED>`
Implementation limitation:	At present, this property is not supported. If you enable this property, StarOffice XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate` to `false` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit` The values `none` and `page` can be set,but they will never be evaluated.

3.12.13 Hyphenation Remain Char Count

Use this property to specify the number of characters that must be present before a hyphenation character.

XML Code:	`fo:hyphenation-remain-char-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-remain-char-count %number; #IMPLIED>`
Implementation limitation:	At present, if you enable this property, StarOffice XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.12.14 Hyphenation Push Char Count

Use this property to specify the minimum number of characters that are moved to the next line.

XML Code:	`fo:hyphenation-push-char-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-push-char-count %number; #IMPLIED>`
Implementation limitation:	At present, if you enable this property, StarOffice XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate property` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.12.15 Maximum Hyphens

Use this property to specify the maximum number of successive lines that can contain a hyphenated word.

XML Code:	`fo:hyphenation-ladder-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-ladder-count (no-limit|%number;) #IMPLIED>`
Note:	In StarOffice 5.2, there was a value `none` instead of `no-limit`.
Implementation limitation:	At present, if you enable this property, StarOffice XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate property` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0`

3.12.16 Drop Caps

This element specifies if the first character(s) of a paragraph are displayed in a larger font.

XML Code:	`<style:drop-cap>`
Rules:	This element can be contained in a `<style:properties>` element.
DTD:	`<!ELEMENT style:drop-cap EMPTY>`

The attributes that you can associate with the `<style:drop-cap>` element are:

- Length
- Lines

- Distance
- Text style

Length

The `style:length` attribute specifies the number of characters that are dropped.

XML Code:	`style:length`
Rules:	The value of this attribute can be a number or `word`, which indicates that the first word should be dropped.
DTD:	`<!ATTLIST style:drop-cap style:length (%number;|word) "1">`
Note:	XSL does not support drop caps but a conversion to XSL may create a formatting object that contains the dropped characters.

Lines

The `style:lines` attribute specifies the number of lines that the dropped characters should encircle.

XML Code:	`style:lines`
Rules:	If the value of this attribute is `1` or `0`, drop caps is disabled.
DTD:	`<!ATTLIST style:drop-cap style:lines %number; "1">`

Distance

The `style:distance` attribute specifies the distance between the last dropped character and the first of the remaining characters of each line.

XML Code:	`style:distance`
Rules:	The value of this attribute is a length.
DTD:	`<!ATTLIST style:drop-cap style:distance %length; "0cm">`

Text Style

The `style:style-name` attribute specifies the text style to apply to the dropped characters.

XML Code:	`style:style-name`
Rules:	
DTD:	`<!ATTLIST style:drop-cap style:style-name CDATA #IMPLIED>`

3.12.17 Register True

The `style:register-true` attribute ensures that when you are using two-sided printing, the printed lines on both sides of a page match. It also ensures that the text in page columns or text box columns is arranged in such a way that the text baselines seem to run from one column to another.

XML Code:	`style:register-true`
Rules:	
DTD:	`<!ATTLIST style:properties style:register-true %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.12.18 Numbering Style

See Section 2.5.1 for information on the number style formatting properties.

3.12.19 Left and Right Margins

Use these properties to specify the left and right margins for a paragraph.

XML Code:	`fo:margin-left` and `fo:margin-right` (XSL properties)
Rules:	These two properties must be attached to an item set element together with the `fo:text-indent` property. If any of the properties is missing, its value is assumed to be 0cm. The value `auto` is not supported.
DTD:	`<!ATTLIST style:properties fo:margin-left (%number;|%percantage;) #IMPLIED>` `<!ATTLIST style:properties fo:margin-right (%number;|%percantage;) #IMPLIED>`
Note:	Unlike XSL, percentage values for these attributes can be used within paragraph styles and relate to the margins of the parent paragraph style of the width of the attribute neighborhood.

3.12.20 Text Indent

Use this property to specify an positive or negative indent for the first line of a paragraph.

XML Code:	`fo:text-indent` (XSL property)
Rules:	This property must be attached to an item set element together with the `fo:margin-left` and `fo:margin-right` properties. If any of these properties is missing, its value is assumed to be 0cm.
DTD:	`<!ATTLIST style:properties fo:text-indent %number; #IMPLIED>`
Note:	Unlike XSL, percentage values can be used within paragraph styles. They percentages relate to the parent paragraph style.

3.12.21 Automatic Text Indent

Use this property to specify that the first line of a paragraph is indented by a value that is based on the current font size.

XML Code:	`style:auto-text-indent`
Rules:	This property must be attached to an item set element together with the `fo:margin-left` and `fo:margin-right` properties. If any of these properties is missing, its value is assumed to be 0cm. If this property is attached to an item set element together with a `fo:text-indent` property that has a value of `true`, the `fo:text-indent` property is ignored.
DTD:	`<!ATTLIST style:properties style:auto-text-indent %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.12.22 Top and Bottom Margins

Use these properties to specify the top and bottom margins for paragraphs.

XML Code:	`fo:margin-top` and `fo:margin-bottom` (XSL properties)
Rules:	These two properties must be attached to an item set element simultaneously. If one of the properties is missing, its value is assumed to be 0cm. The value `auto` is not supported.

DTD:	`<!ATTLIST style:properties fo:margin-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:margin-bottom CDATA #IMPLIED>`

Note:	Unlike XSL, percentage values can used within paragraph styles. The percentages relate to the parent paragraph style instead of the attribute neighborhood.

3.12.23 Page Sequence Entry Point

See Section for detailed information on page sequence entry points.

3.12.24 Break Before and Break After

Use these properties to insert a page or column break before or after a paragraph.

XML Code:	`fo:break-before` and `fo:break-after` (XSL properties)
Rules:	These two properties are mutually exclusive. If they are attached to an item set element simultaneously, the result is undefined. The values `odd-page` and `even-page` behave like a `page` value.

DTD:	`<!ATTLIST style:properties fo:break-before (auto|column|page) #IMPLIED>`
	`<!ATTLIST style:properties fo:break-after (auto|column|page) #IMPLIED>`

3.12.25 Paragraph Background Color

Use this property to specify the background color of a paragraph.

XML Code:	`fo:background-color` (XSL property)
Rules:	The value of this attribute can be either `transparent` or it can be a color. If the value is `transparent`, it switches off any background image that is specified by a `<style:background-image>` element within the same item set element.
DTD:	`<!ATTLIST style:properties fo:background-color % transparent_or_color #IMPLIED>`

3.12.26 Paragraph Background Image

Use this property to specify a background image for a paragraph.

XML Code:	`<style:background-image>`
Rules:	This element is an XLink and is contained within an item set element. If there is no `xlink:href` attribute attached to the element, the background image will not be displayed. If the `<style:background-image>` element is empty and if there is no color specified by an `fo:background-color` element in the same item set element, StarOffice XML sets the background color to transparent.

DTD:	`<!ELEMENT style:background-image EMPTY>`
	`<!ATTLIST style:background-image xlink:type (simple) #IMPLIED>`
	`<!ATTLIST style:background-image xlink:show (embed) #IMPLIED>`
	`<!ATTLIST style:background-image xlink:actuate (onLoad) #IMPLIED>`

Note:	XSL is not suitable for displaying background images because it is not based on XLink.

The attributes that you can associate with the `<style:background-image>` element are:

- Repetition
- Position
- Filter

Repetition

The `style:repeat` attribute specifies whether a background image is repeated or stretched in a paragraph.

XML Code:	`style:repeat`
Rules:	This attribute is attached to the `<style:background-image>` element and its value can be `no-repeat`, `repeat`, or `stretch`.
DTD:	`<!ATTLIST style:background-image style:repeat (no-repeat|repeat|stretch) "repeat">`
Note:	This attribute is similar to the XSL `fo:background-repeat` property, except that XSL does not support the `stretch` value but supports the values `repeat-x` and `repeat-y` instead.

Position

The `style:position` attribute specifies where to position a background image in a paragraph.

XML Code:	`style:position`
Rules:	This attribute is attached to the `<style:background-image>` element and its value can be a space separated combination of `top`, `center`, or `bottom` for the vertical position and `left`, `center`, or `right` for the horizontal position. The vertical and horizontal positions can be specified in any order and if you wish, you can specify just one position in which case the other position defaults to `center`.
DTD:	`<!ATTLIST style:background-image style:repeat CDATA "center">`
Note:	This attribute is similar to the XSL fo:background-position property except that XSL supports a wider range of values.

Filter

The `style:filter-name` attribute specifies the internal StarOffice filter name that is used to load the image into the document.

XML Code:	`style:filter-name`
Rules:	This attribute is attached to the `<style:background-image>` element.
DTD:	`<!ATTLIST style:background-image style:filter CDATA #IMPLIED>`

3.12.27 Border

The border attributes specify the border properties for paragraphs.

XML Code:	`fo:border`
	`fo:border-top`
	`fo:border-bottom`
	`fo:border-left`
	`fo:border-right`

Rules:	The `fo:border` property applies to all four sides of a paragraph while the other properties apply to one side only.

DTD:	`<!ATTLIST style:properties fo:border CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-right CDATA #IMPLIED>`

Implementation limitations:	At present, all four borders must be set simultaneously by using either the `fo:border` property or by attaching all four of the other border properties to an item set element. In the latter case, if one or more of the properties is missing their values are assumed to be `none`. The only border styles supported are `none` or `hidden`, `solid`, and `double`. Any other border style specified is displayed as `solid`. Transparent borders are not supported and the border widths `thin`, `medium`, and `thick` are mapped to lengths. In addition, only some distinct border widths are supported. Unsupported widths are rounded up to the next supported width. If there are no padding properties specified within the same item set element, a default padding is used for sides that have a border. A value of 0cm is used for sides without a border.

3.12.28 Border Line Width

If the line style for a border is double, use the border line attributes to individually specify the width of the inner and outer lines and the distance between them.

XML Code:	`style:border-line-width`
	`style:border-line-width-top`
	`style:border-line-width-bottom`
	`style:border-line-width-left`
	`style:border-line-width-right`

Rules:	The `style:border-line-width` specifies the line widths of all four sides, while the other attributes specify the line widths of one side only. The value of the attributes can be a list of three space-separated lengths, as follows: • The first value specifies the width of the inner line • The second value specified the distance between the two lines • The third value specifies the width of the outer line

DTD:	`<!ATTLIST style:properties style:border-line-width CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-right CDATA #IMPLIED>`

Implementation limitations:	Only a few distinct width triples are supported. Unsupported width triples are rounded to a supported width triple. The result of specifying a border line width without specifying a border width style of `double` for the same border is undefined.
Note:	XSL does not support these border line width properties.

3.12.29 Padding

XML Code:	`fo:padding`
	`fo:padding-top`
	`fo:padding-bottom`
	`fo:padding-left`
	`fo:padding-right`

Rules:	

DTD:	`<!ATTLIST style:properties fo:padding CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-right CDATA #IMPLIED>`

Implementation limitations:	At present, the value of these properties can be a non-zero value if there is a border at the same side and the border is specified within the same item set element. The value can be zero if there is no border at the same side. If an item set element contains a padding specification for one but not all four sides, a zero or a default padding is assigned to these sides depending on whether or not there is a border at that side. If you specify a padding for one or more sides without specifying borders within the same item set element, StarOffice XML switches off all borders that are not set.

3.12.30 Shadow

XML Code:	`style:shadow`
Rules:	The valid values for this attribute are the same as the values for the `fo:text-shadow` property. See Section 3.11.19 for information.
DTD:	`<!ATTLIST style:properties style:shadow CDATA #IMPLIED>`
Implementation limitations:	At present, only one shadow effect is supported at a time. The absolute values of the vertical and horizontal shadow positions must be the same and there is no blur radius.
Note:	XSL does not support this property.

3.12.31 Keep with Next

XML Code:	`style:keep-with-next`
Rules:	
DTD:	`<!ATTLIST style:properties style:keep-with-next %boolean; #IMPLIED>`
Note:	In StarOffice 5.2, this attribute was called `fo:keep-with-next`.

3.12.32 Line Numbering

See Section 3.10 for detailed information on line numbering formatting properties.

3.12.33 Text Autospace, Punctuation Wrap, Line Break

See Section 3.11.30, 3.11.31 and 3.11.32 for information about these properties.

3.13 Section Formatting Properties

You can apply section formatting properties to section descriptions.

3.13.1 Section Background

The background formatting properties for sections are the same as the background properties for paragraphs. See Section 3.12.25 and 3.12.26 for information on background formatting properties for paragraphs.

3.13.2 Columns

The `<style:columns>` element contains the column elements for a section.

XML Code:	`<style:columns>`
Rules:	This element can contain `<style:column>` elements that specify each column individually (see Section 3.13.3). If these elements are not present, all columns are assigned the same width. The `<style:columns>` can contain a `<style:column-sep>` element that describes the separator line between columns. See Section 3.13.4 for information on this element.
DTD:	`<!ELEMENT style:columns (style:column-sep?,(style:column, style:column+)?>`

The attributes that you can associate with the `<style:columns>` element are:

- Column count
- Column gap

Column Count

The `fo:columns-count` attribute specifies the number of columns in a section.

XML Code:	`fo:columns-count`
Rules:	This attribute is essential.
DTD:	`<!ATTLIST style:columns fo:column-count %number #REQUIRED>`
Note:	This attribute has the same name as an XSL property but it is attached to a different element.

Column Gap

If the `<style:columns>` element does not contain individual `<style:column>` elements, you can specify the gap between columns using the `fo:column-gap` attribute.

XML Code:	`fo:column-gap`
Rules:	If there are individual column elements, this attribute is ignored.
DTD:	`<!ATTLIST style:columns fo:column-gap %length #IMPLIED>`
Note:	This attribute has the same name as an XSL property but it is attached to a different element.

3.13.3 Column Specification

The `<style:column>` element can be contained in a `<style:columns>` element, to specify details of an individual column.

XML Code:	`<style:column>`
Rules:	This element is contained in the `<styles:columns>` element. There can be either no column elements or there can be the same number of column elements as specified by the `fo:column-count` attribute.
DTD:	`<!ELEMENT style:column EMPTY>`
Note:	In XSL, it is not possible to specify columns individually.

The attributes that you can associate with the `<style:column>` element are:

- Column width
- Column left, right, upper, and lower space

Column Width

Use the `style:rel-width` attribute to specify the width of a column.

XML Code:	`style:rel-width`
Rules:	The column widths are specified as numbers instead of lengths. To get the absolute column width, the space that is available for a columned area is distributed among the columns proportional to these numbers.
DTD:	`<!ATTLIST style:column style:rel-width %number; #REQUIRED>`

Column Left, Right, Upper, and Lower Space

For each column, you can specify the left, right, upper and lower space. The right space of a column together with the left space of the next column corresponds to the gap between two columns. If a columned area contains a separator line between columns, the space that is occupied by the line is contained within the left and right spaces and therefore is not added to them.

XML Code:	**For left and right spaces:**
	`fo:start-indent`
	`fo:end-indent`
	For upper and lower spaces:
	`fo:space-before`
	`fo:space-after`

Rules:	

DTD:	`<!ATTLIST style:column fo:start-indent %length: "0cm">`
	`<!ATTLIST style:column fo:end-indent %length: "0cm">`
	`<!ATTLIST style:column fo:space-before %length: "0cm">`
	`<!ATTLIST style:column fo:space-after %length: "0cm">`

3.13.4 Column Separator

The `<style:column-sep>` element specifies the separator line to use between columns.

XML Code:	`<style:column-sep>`
Rules:	This element can be contained in a `<style:columns>` element to specify the type of separator line to use between columns.
DTD:	`<!ELEMENT style:column-sep EMPTY>`
Note:	XSL does not support column separators.

The attributes that you can associate with the `<style:column-sep>` element are:

- Line style
- Line width
- Line height
- Vertical line alignment
- Line color

Line Style

Use the `style:style` attribute to specify the line style of the column separator line.

XML Code:	`style:style`
Rules:	

DTD:	`<!ATTLIST style:column-sep style:style`
	`          (none|solid|dotted|dashed|dot-dashed) "solid">`

Line Width

Use the `style:width` attribute to specify the width of the column separator line.

XML Code:	`style:width`
Rules:	
DTD:	`<!ATTLIST style:column-sep style:width %length; #REQUIRED>`

Line Height

Use the `style:height` to specify the height of the column separator line.

XML Code:	`style:height`
Rules:	The value of this attribute is a percentage that relates to the height of the columned area.
DTD:	`<!ATTLIST style:column-sep style:height %percentage; "100%">`

Vertical Line Alignment

Use the `style:vertical-align` attribute to specify how to vertically align a line that is less than 100% of its height within the columned area.

XML Code:	`style:vertical-align`
Rules:	The value of this attribute can be either `top`, `middle`, or `bottom`.
DTD:	`<!ATTLIST style:column-sep style:vertical-align (top|middle|bottom) "top">`

Line Color

Use the `style:color` attribute to specify the color of the column separator line.

XML Code:	`style:color`
Rules:	
DTD:	`<!ATTLIST style:column-sep style:color %color; "#000000">`

3.13.5 Protect

Sections marked with the `style:protect` attribute should not be changed. The user interface should prevent the user from manually making any changes.

XML Code:	`style:protect`
Rules:	The `style:protect` attribute is set by default for linked sections or indexes. Removing the protection makes these sections accessible to the user, but updating the links or the index will not preserve the changes.
DTD:	`<!ATTLIST style:properties style:protect %boolean; "false">`

3.14 Optional Information

The information described in this section can be contained in an XML document to improve performance, but it is not essential.

3.14.1 Wrong List

The wrong list contains a list of all of the words in the document that are spelled incorrectly. This list can be contained in a paragraph element. An additional flag (dirty flag) specifies whether or not the list is valid.

3.14.2 Spelling Configuration

The spelling configuration contains the names of all of the dictionaries that were used to check the spelling in a document and some related information. The information is used to determine whether or not a document should be checked again for spelling. This information is only required if the document contains wrong lists.

3.14.3 Document Statistics

The document statistics contain information about the number of paragraphs, words, tables, and so on, that are contained in a document. This information is only used if the document was saved by a StarOffice application and was not changed by another application afterwards.

3.14.4 Current Number

See Section 3.5.3 for information on the optional current number attribute.

CHAPTER 4

Table Content

This chapter describes the StarOffice XML representation of table and spreadsheet content. It contains the following sections:

- General Introduction to StarOffice Tables
- Calculation Settings
- Change Tracking
- Tables
- Columns
- Rows
- Cells
- Subtables
- Label Ranges
- Named Expressions
- Filters
- Database Ranges
- Data Pilot Tables
- Consolidation
- DDE Links
- Table Formatting Properties
- Column Formatting Properties
- Table Row Formatting Properties
- Table Cell Formatting Properties

4.1 General Introduction to StarOffice Tables

Both StarOffice Writer and StarOffice Calc documents can include tables, but the internal structure of the tables in these applications is quite different. The structure of StarOffice Calc tables is similar to the structure of un-nested HTML and XSL tables. The structure of StarOffice Writer tables is similar to the structure of nested HTML or XSL tables that do not have any vertically merged cells.

Therefore, the StarOffice XML representation of tables is similar to nested HTML and XSL tables:

- A StarOffice Calc XML document does not contain any **subtables** (nested tables).
- A StarOffice Writer XML document does not contain vertically merged cells.

If a document that contains either a subtable or vertically merged cells, or both, is converted to XML, the structure of the table may change. This does not affect how the table appears when the document is displayed.

There are several reasons why you need to preserve the internal StarOffice Writer table structure:

1. The StarOffice API and formulas access table cells using names that are derived from the internal table structure.
2. Within a StarOffice Writer table, rows may have a fixed height or background. If the internal StarOffice Writer table structure is not preserved, there could be rows that do not have a corresponding row in the HTML or XSL representation of the table. This could lead to a loss of information.
3. The internal column widths of a StarOffice Writer table do not have to be the same as the displayed column widths.

The representation of tables is based on a grid of rows and columns. Rows take precedence over columns. That means, the table is divided into rows and the rows are divided into cells. Also, each column includes a column description, but this description does not contain any cells.

Rows and columns appear in **row groups** and **column groups**. These groups specify whether or not to repeat a row or column on the next page.

4.2 Calculation Settings

In spreadsheet documents, there are settings which effect the calculation of formulas, for example the null date or iteration settings. These settings must be saved in the document in the `<table:calculation-settings>` element.

XML Code:	`<table:calculation-settings>`
Rules:	
DTD:	`<!ELEMENT table:calculation-settings (table:null-date?, table:iteration?)>`

The attributes associated with this element are:

- Case Sensitive
- Precision as Shown
- Search criteria must apply to whole cell
- automatic find labels

Case Sensitive

This attribute specifies whether or not to distinguish between upper and lower case in text when comparing cell contents.

XML Code:	`table:case-sensitive`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:case-sensitive %boolean; "true">`

Precision as Shown

This attribute specifies whether to perform a calculation using the rounded values displayed in the spreadsheet, or perform the calculation using all of the digits in the number but display a rounded number.

XML Code:	`<table:precision-as-shown>`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:precision-as-shown %boolean; "false">`

Search Criteria Must Apply to Whole Cell

This attribute specifies whether or not the specified search criteria, according to the regular expression used, must apply to the entire cell contents.

XML Code:	`<table:search-criteria-must-apply-to-whole-cell>`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:search-criteria-must-apply-to-whole-cell %boolean; "true">`

Automatic Find Labels

This attribute specifies whether or not to automatically find the labels of rows and columns.

XML Code:	`<table:automatic-find-labels>`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:automatic-find-labels %boolean; "true">`

4.2.1 Null Date

This element specifies the null date.

XML Code:	`<table:null-date>`
Rules:	
DTD:	`<!ELEMENT table:null-date EMPTY>`
Implementation limitation:	In Star Office are only three possible null dates ; `12/30/1899`, `01/01/1900`, and `01/01/1904`.

The attributes associated with this element are:

- Value Type and Date Value

Value Type and Date Value

XML Code:	`table:value-type` `table:date-value`
Rules:	

DTD:	`<!ATTLIST table:null-date table:value-type %valueType; #FIXED "date"`
	`table:date-value %date; "1899-12-30">`

4.2.2 Iteration

If the `<table:iteration>` element is enabled, formulas with iterative references (formulas that are repeated until the problem is solved) are calculated after a specific number of iterations. If this element is not enabled an iterative reference in the table causes an error message.

XML Code:	`<table:iteration>`
Rules:	
DTD:	`<!ELEMENT table:iteration EMPTY>`

The attributes associated with this element are:

- Status
- Steps
- Minimum Difference

Status

This attribute specifies whether or not the iteration is enabled.

XML Code:	`table:status`
Rules:	
DTD:	`<!ATTLIST table:iteration table:status (enable|disable) "disable">`

Steps

This attribute specifies the maximum number of iteration steps.

XML Code:	`table:steps`
Rules:	
DTD:	`<!ATTLIST table:iteration table:steps %positiveInteger; "100">`

Maximum Difference

This attribute specifies the maximum difference between two calculation results. The iteration is finished if the result is less than the value specified in this attribute.

XML Code:	`<table:maximum-difference>`
Rules:	
DTD:	`<!ATTLIST table:iteration table:maximum-difference %float; "0.001">`

4.3 Change Tracking

In StarOffice Writer documents, you cannot track changes in tables. In StarOffice Calc documents, you can track changes in tables.

This section describes how StarOffice tracks changes to table content in the spreadsheet application.

All tracked changes to spreadsheet documents are stored in a list. The list contains an element for each change made to the document.

XML Code:	`<table:tracked-changes>`
Rules:	
DTD:	`<!ELEMENT table:tracked-changes (table:changed-region)+>`

4.3.1 Changed Regions

For every changed region of a document, there is one entry in the list of tracked changes. This entry contains a list of all changes that were applied to the region. The start and end of this region are marked by the start and end elements that are described in the next section.

XML Code:	`<table:changed-region>`
Rules:	Every element has an ID. The elements that mark the start and end of a region use this ID to identify the region to which they belong.
DTD:	`<!ELEMENT table:changed-region (table:insertion|table:deletion|table:moving)+>` `<ATTLIST  table:changed-region table:id ID #REQUIRED>`

4.3.2 Region Start and End

There are three elements that mark the start and the end of a changed region, as follows:

- Change start element – `<table:change-start>`
  This element marks the start of a region with content where text, rows or columns has been inserted or where cells has been moved.
- Change end element – `<table:change-end>`
  This element marks the end of a region with content where text, rows or columns has been inserted or where cells has been moved.
- Change position element – `<table:change>`

This element marks a position in an empty region where text, rows or columns has been deleted.

XML Code:	`<table:change-start>`
	`<table:change-end>`
	`<table:change>`

Rules:	All three elements have an attribute that specifies the ID of the region to which they belong.

DTD:	`<!ELEMENT table:change-start EMPTY>`
	`<!ELEMENT table:change-end EMPTY>`
	`<!ELEMENT table:change EMPTY>`
	`<!ATTLIST table:change-start table:region-id IDREF #REQUIRED>`
	`<!ATTLIST table:change-end table:region-id IDREF #REQUIRED>`
	`<!ATTLIST table:chang table:region-id IDREF #REQUIRED>`

4.3.3 Insertion

The `<table:insertion>` element contains the information that is required to identify any insertion of content. This content can be a cell content, one or more rows, one or more columns, or a table. The inserted content is part of the table document itself and is marked by a change start and a change end element.

XML Code:	`<table:insertion>`
Rules:	
DTD:	`<!ELEMENT table:insertion (office:change-info)>`

Example: Insertion of text in a cell

```
<table:tracked-changes>
  <table:changed-region table:id="c001">
    <table:insertion>
      <office:change-info office:chg-author="Sascha Ballach"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04"/>
    </table:insertion>
  </table:changed-region>
</table:tracked-changes>
...
<table:table-cell ...>
  <table:change-start table:region-id="c001"/>
  <text:p>
    This is the original text, but this has been added.
  </text:p>
  <table:change-end table:region-id="c001"/>
</table:table-cell>
```

4.3.4 Deletion

A `<table:deletion>` element contains content that was deleted while change tracking was enabled. The position where the cell content, the row, or the column was deleted is marked by the change position element (`<table:change>`).

XML Code:	`<table:deletion>`
Rules:	If the content of a cell was deleted, the deleted paragraphs are contained in this element. If one or more rows were deleted, the deleted row with the deleted cells are contained in this element. If one or more columns were deleted, the deleted columns with the deleted cells are contained in this element. Every row in this columns is represented by a `<table:table-row>` element like in a table.
DTD:	`<!ELEMENT table:deletion (style:change-info,(%text;|%table-rows;)>`

Example: Deletion of text of a cell

```
<table:tracked-changes>
  <table:changed-region text:id="c002">
    <table:deletion>
      <office:change-info office:chg-author="Sascha Ballach"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04">
      <text:p>
        , but this has been deleted
      </text:p>
    </table:deletion>
  </table:changed-region>
</table:tracked-changes>
...
<table:table-cell ...>
  <text:change text:region-id="c002"/>
</table:table-cell>
```

4.3.5 Moving

A `<table:moving>` element contains the information that is required to identify any movement of content. This content can be a cell content or a cell range content. The moved content is part of the table document itself and is marked by:

- A change start and a change end element on the position where the content is moved to
- A change element on the position where the content is moved from

All three elements contains the same ID. If the range goes over more than one row, the correct elements are located in the right position in every row.

XML Code:	`<table:moving>`
Rules:	
DTD:	`<!ELEMENT table:moving (office:change-info)>`

Example: Moving a cell

```
<table:tracked-changes>
  <table:changed-region table:id="c003">
    <table:moving>
      <office:change-info office:chg-author="Sascha Ballach"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04"/>
    </table:moving>
  </table:changed-region>
</table:tracked-changes>
...
<table:table-cell ...>
  <table:change-start table:region-id="c003"/>
  <text:p>
    This is the original text, but this has been added.
  </text:p>
  <table:change-end table:region-id="c003"/>
</table:table-cell>
...
<table:table-cell ...>
  <table:change table:region-id="c003"/>
</table:table-cell>
```

4.4 Tables

4.4.1 Table

The table element describes a table.

XML Code:	`<table:table>`
Rules:	The content of a table element is one or more groups of columns and rows.
DTD:	`<!ENTITY % table-columns "( table:table-columns | ( table:table-column | table:table-column-group )+ )">` `<!ENTITY % table-header-columns "table:table–header-columns">` `<!ENTITY % table-rows "( table:table-rows | ( table:table-row+ | table:table-row-group ) )">` `<!ENTITY % table-header-rows "table:table-header-rows">` `<!ENTITY % table-column-groups "( (%table-header-columns;?, %table-columns;) |(%table-columns;, %table-header-columns;, %table-columns;?) )">` `<!ENTITY % table-row-groups "( (%table-header-rows;?, %table-rows;) | (%table-rows;, %table-header-rows;, %table-rows;?) )">` `<!ELEMENT table:table (table:view-settings, table:table-source?, table:scenario?, table:shapes?, %table-column-groups;,%table-row-groups;)>`

Table Name

A `table:name` attribute specifies the name of a table.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:table table:name CDATA #REQUIRED>`

Table Style

A table style attribute describes the formatting properties of a table, such as width and background color. The table style may be either an automatic or a common style.

XML Code:	`table:style-name`
Rules:	You define a table style using a `<style:style>` element and a family attribute value of `table`. The `<table:table>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:table table:style-name %style-name #REQUIRED>`

Example: Table Style

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  ...
</table:table>
```

DDE Connection

Information to be supplied.

XML Code:	`office:dde-source`
	`office:dde-command`
	`office:dde-mode`

Rules:	

DTD:	`<!ATTLIST table:table office:dde-source CDATA #IMPLIED>`
	`<!ATTLIST table:table office:dde-command CDATA #IMPLIED>`
	`<!ATTLIST table:table office:dde-mode CDATA #IMPLIED>`

Use Cell Protection

This attribute specifies whether or not a table is protected and if it is protected, another attribute specifies the password. If a table is protected, all of the table elements and the cell elements with a `style:cell-protect` attribute set to `true` are protected.

XML Code:	`table:use-cell-protection` and `table:cell-protection-key`
Rules:	These attributes can be attached to the `<table:table>` element.

DTD:	`<!ATTLIST table:table table:use-cell-protection %boolean; "false">`
	`<!ATTLIST table:table table:cell-protection-key CDATA #IMPLIED>`

Print Ranges

This attribute specifies the print ranges of the table. It contains a list of cell addresses or cell range addresses.

XML Code:	`table:print-ranges`
Rules:	This attribute can be attached to the `<table:table>` element.
DTD:	`<!ATTLIST table:table table:print-ranges %cell-range-address-list; #IMPLIED>`

4.4.2 View Settings

Spreadsheet documents contain settings to specify how the table is viewed. For example, whether the table is the active table, the current cursor position, the current position of the left top edge of the table, whether the table is split into separate parts, and the location of the left top edges of the parts.

XML Code:	`<table:view-settings>`
Rules:	
DTD:	`<!ELEMENT table:view-settings (table:split?)>`

The attributes associated with this element are:

- Is Active
- Cursor position
- Left Top Edge Position
- Horizontal Scroll Bar Position

Is Active

This attribute specifies whether or not the current table is active.

XML Code:	`table:active`
Rules:	Only one table in the document can be active.
DTD:	`<!ATTLIST table:view-settings table:active %boolean; "false">`

Cursor Position

This attribute specifies the cursor position in the current table.

XML Code:	`table:cursor-position`
Rules:	
DTD:	`<!ATTLIST table:view-settings table:cursor-position %cell-address; #REQUIRED>`

Left Top Edge Position

This attribute specifies the left top visible edge of the table.

XML Code:	`table:left-top-edge-position`
Rules:	
DTD:	`<!ATTLIST table:view-settings table:left-top-edge-position %cell-address; #REQUIRED>`

Horizontal Scroll Bar Position

This attribute specifies the position of the left end of the horizontal scroll bar. The base position is the left side of the table.

XML Code:	`table:horizontal-scroll-bar-position`
Rules:	
DTD:	`<!ATTLIST table:view-settings table:horizontal-scroll-bar-position %nonNegativeLength; #REQUIRED>`

4.4.3 Table Split

This element specifies whether or not a table is split and and how it is split. The table can be split by a row, a column, or both.

XML Code:	`table:split`
Rules:	
DTD:	`<!ELEMENT table:split (table:split-column | table:split-row | (table:split-column, table:split-row))>`

The attributes associated with this element are:

- Freeze

Freeze

This attribute specifies whether or not to freeze the split. Freezing a worksheet is especially useful when working with large spreadsheet documents, so that you can view the column or row headers even when you are working in another part of the document. A horizontal line and a vertical line are displayed, extending from the active cell. These lines divide the screen into left, right, upper, and lower sections. If the split is not frozen, all four sections contain a scroll bar which you can use to view the entries in the individual sections. The lines can be moved between the rows and columns or the lines cannot be moved.

XML Code:	`table:freeze`
Rules:	
DTD:	`<!ATTLIST table:split table:freeze %boolean; #REQUIRED>`

4.4.4 Split Field

This element specifies the position of the column split. If the split is frozen, the position is a column/row. The split is after the given column/row. If the split is not frozen, the position is a length from the left/top of the table.

XML Code:	`table:split-column` `table:split-row`
Rules:	

DTD:	`<!ELEMENT table:split-column EMPTY>`
	`<!ATTLIST table:split-column table:split-position % nonNegativeLength; #IMPLIED>`
	`<!ATTLIST table:split-column table:freeze-position % nonNegativeInteger; #IMPLIED>`
	`<!ELEMENT table:split-row EMPTY>`
	`<!ATTLIST table:split-row table:split-position % nonNegativeLength; #IMPLIED>`
	`<!ATTLIST table:split-row table:freeze-position % nonNegativeInteger; #IMPLIED>`

The attributes associated with this element are:

- Split Position
- Freeze Position
- left top edge position

Split Position

This attribute specifies the position of the split line, which is either a split position or a freeze position.

XML Code:	`table:split-position` `table:freeze-position`
Rules:	

DTD:	`<!ATTLIST table:split-column table:split-position % nonNegativeLength; #IMPLIED>`
	`<!ATTLIST table:split-column table:freeze-position % nonNegativeInteger; #IMPLIED>`
	`<!ATTLIST table:split-row table:split-position % nonNegativeLength; #IMPLIED>`
	`<!ATTLIST table:split-row table:freeze-position % nonNegativeInteger; #IMPLIED>`

Left Top Edge Position

This attribute specifies the left top visible edge of the split table range. The value have to the same column like the value of the `table:left-top-edge-position` attribute of the `<table:view-setting>` element if it is a row split or the same row if it is a column split.

XML Code:	`table:left-top-edge-position`
Rules:	
DTD:	`<!ATTLIST table:split-column table:left-top-edge-position %cell-address; #REQUIRED>` `<!ATTLIST table:split-row table:left-top-edge-position %cell-address; #REQUIRED>`

4.4.5 Table Source

If a table is linked to an original table, the original table is represented by a table source element.

XML Code:	`<table:table-source>`
Rules:	
DTD:	`<!ELEMENT table:table-source EMPTY>`

The attributes associated with this element are:

- Mode
- URL
- Filter name
- Table name
- Filter options

Mode

This attribute specifies if the table is a reference to another table and how the data should be copied.

XML Code:	`table:mode`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST table:table-source table:mode ( "copy-all" | "copy-results-only" ) "copy-all">`

URL

The XLink attributes specify the URL of the linked table document.

XML Code:	`xlink:type`, `xlink:actuate`, and `xlink:href`
Rules:	
DTD:	`<!ATTLIST table:table-source xlink:type (simple) #FIXED "simple">` `<!ATTLIST table:table-source xlink:actuate (onRequest) "onRequest">` `<!ATTLIST table:table-source xlink:href %url; #REQUIRED>`

Filter Name

This attribute specifies the file type of the linked table document.

XML Code:	`table:filter-name`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:table-source table:filter-name CDATA #IMPLIED>`

Table Name

This attribute specifies the name of the table in the linked table document.

XML Code:	`table:table-name`
Rules:	
DTD:	`<!ATTLIST table:table-source table:table-name CDATA #REQUIRED>`

Filter Options

This attribute specifies optional settings about the file type.

XML Code:	`table:filter-options`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:table-source table:filter-options CDATA #IMPLIED>`

4.4.6 Scenario Table

The `<table:scenario>` element represents a scenario table. The name of the table and the name of the scenario are the same. The scenario is displayed in the regular table preceeding the scenario table. Only one scenario table can be active at one time.

XML Code:	`<table:scenario>`
Rules:	
DTD:	`<!ELEMENT table:scenario EMPTY>`

The attributes that you can associate with this element are:

- Display Border
- Border Color
- Copy Back
- Copy Styles
- Copy Formulas
- Is Active
- Scenario Ranges

- Comment

Display Border

The `table:display-border` attribute specifies whether or not to display the border of the scenario.

XML Code:	`table:display-border`
Rules:	
DTD:	`<!ATTLIST table:scenario table:display-border %boolean; "true">`

Border Color

The `table:border-color` attribute specifies the color of the border.

XML Code:	`table:border-color`
Rules:	
DTD:	`<!ATTLIST table:scenario table:border-color %color; #IMPLIED>`

Copy Back

The `table:copy-back` attribute specifies whether or not data is copied back into the active scenario table if another scenario is activated.

XML Code:	`table:copy-back`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, you can directly edit the data for each scenario in the scenario table.
DTD:	`<!ATTLIST table:scenario table:copy-back %boolean; "true">`

Copy Styles

The `table:copy-styles` attribute specifies whether or not to copy the cell styles with the data.

XML Code:	`table:copy-styles`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST table:scenario table:copy-styles %boolean; "true">`

Copy Formulas

The `table:copy-formulas` attribute specifies whether or not to copy the formulas.

XML Code:	`table:copy-formulas`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the formulas are copied. If the value is `false`, only the values resulting from the formulas are copied.
DTD:	`<!ATTLIST table:scenario table:copy-formulas %boolean; "true">`

Is Active

The `table:is-active` attribute specifies whether or not the current scenario is active.

XML Code:	`table:is-active`
Rules:	
DTD:	`<!ATTLIST table:scenario table:is-active %boolean; #REQUIRED>`

Scenario Ranges

The `table:scenario-ranges` attribute specifies the range of this scenario.

XML Code:	`table:scenario-ranges`
Rules:	The value of this attribute is a list of cell range addresses.
DTD:	`<!ATTLIST table:scenario table:scenario-ranges %cell-range-address-list; #REQUIRED>`

Comment

The `table:comment` attribute contains a comment about the scenario.

XML Code:	`table:comment`
Rules:	
DTD:	`<!ATTLIST table:scenario table:comment CDATA #IMPLIED>`

4.4.7 Shapes

This element contains all shapes with an anchor on a table. This is a container element and does not have any associated attributes.

XML Code:	`<table:shapes>`
Rules:	
DTD:	`<!ELEMENT table:shapes ANY>`

4.5 Columns

4.5.1 Grouping

Columns can be grouped. Every group can contain a new group, columns, and columnheaders. Every group can be visible or hidden.

The `table:table-column-headers` should only be separated by `<table:table-column-group>` elements, so that if the `table:table-column-group` does not exist there is only one `table:table-header-columns` element.

XML Code:	`<table:table-column-group>`
Rules:	There can only be one `<table:table-header-columns>` element in this element.
DTD:	`<!ELEMENT table:table-column-group "( table:table-header-columns | table:table-column | table:table-column-group)+" >`

The attributes associated with this element are:

- Display

Display

This attribute specifies whether or not the group is visible.

XML Code:	`<table:display>`
Rules:	
DTD:	`<!ATTLIST table:table-column-group table:display %boolean; "true">`

4.5.2 Column Groups

There are two types of column groups, as follows:

- **Header groups**

  A header group is a group of columns that repeat on each page if the table extends over several pages.

- **Body groups**

  A body group is a group of columns that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.

XML Code:	`<table:table-header-columns>`
	
	`<table:table-columns>`

Rules:	The table header column element represents a header column. The table column element represents a body column. You can omit the `<table:table-columns>` element, in the same way that you can omit the`<TBODY>` tag in HTML. A table must contain at least one column group, but only one header group. A body group must not follow another body group.

DTD:	`<!ELEMENT table:table-header-columns ( table:table-column | table:table-column-group )+>`
	`<!ELEMENT table:table-columns ( table:table-column | table:table-column-group )+>`

Notes:	Applications may support column header groups, but this is not essential. If a user agent does not support header groups, it must process header groups as body groups. There are no column groups in XSL.

4.5.3 Column Description

Every column in a table has a column description element. If two or more columns are adjoining, and have the same properties, you can describe them using a single `<table:table-column>` element.

XML Code:	`<table:table-column>`
Rules:	
DTD:	`<!ELEMENT table:table-column EMPTY>`
Notes:	The `<table:table-column>` element is similar to the XSL `<fo:table-column>` element.

Number of Columns Repeated

The number of columns repeated attribute specifies the number of columns a column description applies to.

XML Code:	`table:number-columns-repeated`
Rules:	If two or more columns are adjoining, and have the same properties, you can use a single `<table:table-column>` element to describe them. In this case, you use a `table:number-columns-repeated` attribute to specify the number of successive columns that the description applies to. You specify this attribute with the `<table:table-column>` element.
DTD:	`<!ATTLIST table:table-column table:number-columns-repeated %number; "1">`

Column Style

A table style stores the formatting properties of a table column, such as width and background color. The table style may be either an automatic or a common style. You specify the style of a column using a table style.

XML Code:	`table:style-name`
Rules:	To define the style of the column, you use a `<style:style>` element and a family attribute value of `table`. The `<table:table-column>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:style-name %style-name; #REQUIRED;>`

Visibility

This attribute specifies whether the column is visible, filtered, or collapsed.

XML Code:	`table:visibility`
Rules:	This attribute is associated with the `<table:table-column>` element. If the value of this attribute is `filter`, the column is also collapsed.
DTD:	`<!ATTLIST table:table-column table:visibility ( visible | collapse | filter ) "visible">`

Example: Table with three columns

This example shows the StarOffice XML for a table with three columns.

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  ...
</table:table>
```

4.6 Rows

4.6.1 Grouping

Rows can be grouped. Every group can contain a new group, rows, and row-headers. Every group can be visible or hidden.

The `table:table-row-headers` element can only be separated by `table:table-row-group` elements, so that if `table:table-row-group` does not exist there is only one `table:table-header-rows` element.

XML Code:	`<table:table-row-group>`
Rules:	There can only be one `table:table-header-rows` element in this element.
DTD:	`<!ELEMENT table:table-row-group "( table:table-header-rows | table:table-row | table:table-row-group)+" >`
Notes:	

The attributes associated with this element are:

- Display

Display

This attribute specifies whether or not the group is visible.

XML Code:	`<table:display>`
Rules:	
DTD:	`<!ATTLIST table:table-row-group table:display %boolean; "true">`

4.6.2 Row Groups

There are two types of row groups, as follows:

- **Header group**

  A header group is a group of rows that repeat on each page if the table extends over several pages.

- **Body group**

  A body group is a group of rows that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.

XML Code:	`<table:table-header-rows>`
	`<table:table-rows>`

Rules:	The table header rows element represents a header row. The table row element represents a body row. You can omit the `<table:table-rows>` element, in the same way that you can omit the`<TBODY>` tag in HTML. A table must contain at least one row group, but only one header group. A body group must not follow another body group.

DTD:	`<!ELEMENT table:table-header-rows ( table:table-row | table:table-row-group )+>`
	`<!ELEMENT table:table-rows ( table:table-row | table:table-row-group )+>`

Notes:	Applications may support row header groups, but this is not essential. If a user agent does not support header groups, it must process header groups as body groups.

4.6.3 Row

The table row element includes other elements that specify the content of a table row.

XML Code:	`<table:table-row>`
Rules:	

DTD:	`<!ENTITY % table-cell "(table:table-cell|table:covered-table-cell)">`
	`<!ELEMENT table:table-row %table-cell;+>`

Notes:	The `<table:table-row>` element is similar to the XSL `<fo:table-row>` element.

Number of Rows Repeated

The number of rows repeated attribute specifies the number of rows a row element applies to. If two or more rows are adjoining, and have the same content and properties, you can use a single `<table:table-row>` element to describe them. This attribute specifies the number of successive rows to which a table row element applies.

XML Code:	`table:number-rows-repeated`
Rules:	You specify this attribute with the `<table:table-row>` element.
DTD:	`<!ATTLIST table:table-row table:number-rows-repeated %number; "1">`
Notes:	A row that contains vertically merged cell cannot repeat.

Row Style

A table style stores the formatting properties of a table row, such as height and background color. The table style may be either an automatic or a common style. You specify the style of a row using a table style.

XML Code:	`table:style-name`
Rules:	To define the style of the row, you use a `<style:style>` element and a family attribute value of `table`. The `<table:table-row>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:table-row table:style-name %style-name; #IMPLIED;>`
Notes:	The table row style attribute is similar to the XSL `<fo:table-row>` element.

Visibility

This attribute specifies whether the row is visible, filtered, or collapsed.

XML Code:	`table:visibility`
Rules:	This attribute is associated with the `<table:table-row>` element. If the value of this attribute is `filter`, the row also collapsed.
DTD:	`<!ATTLIST table:table-row table:visibility ( visible | collapse | filter ) "visible">`

Example: Table with three rows and three columns

This example shows the StarOffice XML for a table with three rows and three columns. The first two rows of the table have a blue background.

```
<style:style style:name="Table 1" style:family="table">

  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="blue"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-rows>
    <table:table-row table:style-name="Row1">
      ...
    </table:table-row>
    <table:table-row table:style-name="Row1">
      ...
    </table:table-row>
    <table:table-row>
      ...
    </table:table-row>
  <table:table-rows>
</table:table>
```

4.7 Cells

4.7.1 Table Cell

The table cell element specifies a table cell. Table row elements contain table cell elements.

XML Code:	`<table:table-cell>`
	
	`<table:covered-table-cell>`

Rules:	When table cells merge horizontally and vertically, the `<table:covered-table-cell>` element represents cells that are covered by other cells. The `<table:table-cell>` element represents all other cells. A table cell may either contain either paragraphs and other text content or one subtable.

DTD:	`<!ENTITY % cell-content "(table:cell-range-source?, office: annotation?,(table:subtable|%text-wo-table;))">`
	
	`<!ELEMENT table:table-cell %cell-content;>`
	
	`<!ELEMENT table:covered-table-cell %cell-content;>`

Notes:	There is a difference between the representation of cells that span several rows or columns in StarOffice XML, and their representation in HTML and XSL. When a cell merges with other cells, the cells that the first cell covers do not appear in the HTML or XSL representation of the table. StarOffice XML represents these covered cells as `<table: covered-table-cell>` elements. The reasons for this are as follows: • In spreadsheets, there may be some content in the covered cells. • If a row does not include covered cells, it is very difficult to identify the column that contains a particular cell. It means that you must process all preceding cells to identify the column. Identifying the column that contains a particular cell is very important for transformations to other XML languages, because this information is essential to calculate the width of a cell. Apart from this, the table cell element is similar to the XSL `<fo:table-cell>` element and the HTML `<td>` and `<th>` tags.

Number of Cells Repeated

The number of cells repeated attribute specifies the number of times a cell repeats.

XML Code:	`table:number-columns-repeated`
Rules:	You can use a single `<table:table-cell>` element to describe two or more adjoining cells, if they meet the following conditions: • The cells contain the same content and properties. • The cells are not merged horizontally or vertically. In this case, you use a `table:number-columns-repeated` attribute to specify the number of successive columns that the cell repeats in. You specify this attribute with either the `<table:table-cell>` element or the `<table:covered-table-cell>` element.

DTD:	`<!ATTLIST table:table-cell`
	`table:number-columns-repeated %number; "1">`
	`<!ATTLIST table:covered-table-cell`
	`table:number-columns-repeated %number; "1">`

4.7.2 Cell Content

The content of a cell element is the paragraphs or subtable that appears in the cell. Additional is a Cell Range Source element and a Annotation element is possible.

Number of Rows and Columns Spanned

The number of rows spanned and number of columns spanned attributes specify the number of rows and columns that a cell spans. You specify these attributes with the cell element.

XML Code:	`table:number-rows-spanned` `table:number-columns-spanned`
Rules:	

DTD:	`<!ATTLIST table:table-cell table:number-rows-spanned %number; "1">`
	`<!ATTLIST table:table-cell table:number-columns-spanned %number; "1">`

Notes:	When a cell covers another cell, a `<table:covered-table-cell>` element must appear in the table to represent the covered cell.

Cell Style

A table style stores the formatting properties of a cell, such as the following:

- Background color
- Number format
- Vertical alignment
- Borders

The table style may be either an automatic or a common style. You specify the style of a cell using a table style.

XML Code:	`table:style-name`
Rules:	The value of this attribute must be the name of a `<style:style>` element that belongs to the `table-cell` style family.

DTD:	`<!ATTLIST table:table-cell table:style-name %style-name; #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:style-name %style-name; #IMPLIED>`

Cell Content Validation

A cell can contain a validity check.

XML Code:	`table:content-validation-name`
Rules:	The value of this attribute is the name of a `<table:cell-content-validation>` element.

DTD:	`<!ATTLIST table:table-cell table:content-validation-name CDATA #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:content-validation-name CDATA #IMPLIED>`

See Section for more information on cell content validation and the `<table:cell-content-validation>` element.

Formula

Formulas allow you to make calculations within table cells. Every formula begins with an equal (=) sign. Formulas can include:

- Numbers.
- Text.
- Named ranges.
- Operators.
- Logical operators.
- Function calls.
- Addresses of cells containing numbers. The references can be relative or absolute, see Section 4.7.5. Addresses in formulas start with a "[" and end with a "]". See Sections 4.7.5and 4.7.9 for information on how to address a cell or cell range.

The following is an example of a simple formula:

```
=sum([.A1:.A5])
```

This formula calculates the sum of the values of all cells in the range "`.A1:.A5`". "`sum`" is a function. The parameters are marked by a "`(`" at the start, and a "`)`" at the end. If a function contains more than one parameter, the parameters are separated by a "`;`".

The following is a variation of the same formula shown above:

```
=sum([.A1];[.A2];[.A3];[.A4];[.A5])
```

The result of this formula is the same. The components that you use in the formula depend on the the application you are using.

The formula attribute contains a formula for a table cell.

XML Code:	`table:formula`
Rules:	See above.

DTD:	`<!ATTLIST table:table-cell table:formula CDATA #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:formula CDATA #IMPLIED>`

Notes:	One of the following attributes represents the current value in the cell: • `table:value` • `table:date-value` • `table:time-value` • `table:boolean-value` • `table:string-value`

Matrix

When carrying out spreadsheet calculations, a connected range of cells containing values is designated as a matrix . If the cell range contains *m* rows and *n* columns, the matrix is called an *m x n* matrix. The smallest possible matrix is a *1 x 2* or *2 x 1* matrix with two adjacent cells. If you want to use a matrix in a formula, you must include the cell range address of the matrix in the formula. In a matrix formula, only some special matrix operations are possible.

The number of rows and columns that a matrix spans are represented by the `table:number-matrix-rows-`

`spanned` and `table:number-matrix-columns-spanned` attributes, which are attached to the cell elements.

XML Code:	`table:number-matrix-rows-spanned`
	
	`table:number-matrix-columns-spanned`

Rules:	These attributes are attached to the cell element in the first row and the first column of the matrix.

DTD:	`<!ATTLIST table:covered-table-cell table:number-matrix-rows-spanned %number; #IMPLIED>`
	`<!ATTLIST table:table-cell table:number-matrix-rows-spanned %number; #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:number-matrix-columns-spanned %number; #IMPLIED>`
	`<!ATTLIST table:table-cell table:number-matrix-columns-spanned %number; #IMPLIED>`

Value Type

A value type attribute describes the type of value that can appear in a cell.

XML Code:	`table:value-type`
Rules:	The `table:cell-type` may contain one of the following values: • `float` • `time` • `date` • `percentage` • `currency` • `boolean` • `string`

DTD:	`<!ATTLIST table:table-cell table:value-type ("float"|"time"|"date"|"percentage"|"currency"|"boolean"|"string") "string">`
	`<!ATTLIST table:covered-table-cell table:value-type ("float"|"time"|"date"|"percentage"|"currency"|"boolean"|"string") "string">`

Cell Current Numeric Value

A cell current numeric value attribute specifies the numeric value of a cell.

XML Code:	`table:value`
Rules:	This attribute evaluates only for cells of the following data types: • `float` • `percentage` • `currency`

DTD:	`<!ENTITY % float CDATA>`
	`<!ATTLIST table:table-cell table:value %float; #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:value %float; #IMPLIED>`

Cell Current Date Value

A current date value attribute specifies the date value of a cell.

XML Code:	`table:date-value`
Rules:	This attribute is only evaluated for cells whose data type is `date`.
DTD:	`<!ATTLIST table:table-cell table:date-value %date; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:date-value %date; #IMPLIED>`

Cell Current Time Value

A cell current time value attribute specifies the time value of a cell.

XML Code:	`table:time-value`
Rules:	This attribute is only evaluated for cells whose data type is `time`.
DTD:	`<!ATTLIST table:table-cell table:time-value %time; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:time-value %time; #IMPLIED>`

Cell Current Boolean Value

A cell current Boolean value attribute specifies the Boolean value of a cell.

XML Code:	`table:boolean-value`
Rules:	This attribute is only evaluated for cells whose data type is `boolean`.
DTD:	`<!ATTLIST table:table-cell table:boolean-value %boolean; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:boolean-value %boolean; #IMPLIED>`

Cell Current String Value

A cell current string value attribute specifies the string value of a cell.

XML Code:	`table:string-value`
Rules:	This attribute is only evaluated for cells whose data type is `string`.
DTD:	`<!ATTLIST table:table-cell table:string-value CDATA #IMPLIED>` `<!ATTLIST table:covered-table-cell table:string-value CDATA #IMPLIED>`
Notes:	If the `table:string-value` attribute does not exist, the value of the cell is the text content of the cell.

Cell Current Currency Value

A cell current currency value attribute specifies the currency information for a cell. The value of this attribute is typically currency information such as DEM or EUR.

XML Code:	`table:currency`
Rules:	This attribute is only evaluated for cells whose data type is `currency`.
DTD:	`<!ATTLIST table:table-cell table:currency CDATA #IMPLIED>` `<!ATTLIST table:covered-table-cell table:currency CDATA #IMPLIED>`

4.7.3 Cell Range Source

If a cell is linked to a database range or named range of another file, the original database range or named range is represented by a cell range source element.

XML Code:	`<table:cell-range-source>`
Rules:	
DTD:	`<!ELEMENT table:cell-range-source EMPTY>`

The attributes associated with this element are:

- Name
- URL
- Filter name
- Filter options
- Last size

Name

This attribute specifies the name of the database range or named range.

XML Code:	`table:name`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST table:cell-range-source table:name CDATA #REQUIRED>`

URL

The XLink attributes specify the URL of the linked table document.

XML Code:	`xlink:type`, `xlink:actuate`, and `xlink:href`
Rules:	
DTD:	`<!ATTLIST table:cell-range-source xlink:type (simple) #FIXED "simple">` `<!ATTLIST table:cell-range-source xlink:actuate (onRequest) #FIXED "onRequest">` `<!ATTLIST table:cell-range-source xlink:href %url; #REQUIRED>`

Filter Name

This attribute specifies the file type of the linked table document.

XML Code:	`table:filter-name`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:cell-range-source table:filter-name CDATA #REQUIRED>`

Filter Options

This attribute specifies optional settings about the file type.

XML Code:	`table:filter-options`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:cell-range-source table:filter-options CDATA #IMPLIED>`

Last Size

This attribute specifies the size of the range. This is the last known size of the range. If the size of the range is changed after the last actualisation, this attribute is incorrect.

XML Code:	`table:last-column-spanned` `table:last-row-spanned`
Rules:	This attributes are mandatory.

DTD:	`<!ATTLIST table:cell-range-source table:last-column-spanned %positiveInteger; #REQUIRED>`
	`<!ATTLIST table:cell-range-source table:last-row-spanned %positiveInteger; #REQUIRED>`

4.7.4 Annotation

An annotation element specifies a StarOffice annotation.

XML Code:	`<office:annotation>`
Rules:	
DTD:	`<!ELEMENT office:annotation #PCDATA>`

The attributes associated with the annotation element are:

- Author
- Creation date
- Display

Author

This author attribute specifies the author of the annotation.

XML Code:	`office:author`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST office:author CDATA; #REQUIRED>`

Creation Date

This attribute specifies the creation date and time of the annotation. If the application only has a date string and cannot parse this string, it must write the string to the `office:create-date-string` attribute.

XML Code:	`office:create-date`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST office:create-date %date-time; #IMPLIED>` `<!ATTLIST office:create-date-string %string; #IMPLIED>`

Display

This attribute specifies whether or not the annotation is visible.

XML Code:	`office:display`
Rules:	This attribute is optional and can have a `true` or `false` value.
DTD:	`<!ATTLIST office:display %boolean; "true">`

4.7.5 Detective

The `<table:detective>` element allows you to display links between the current formula cells and the cells in the spreadsheet document. This makes is easy to check formula definitions and make any necessary corrections.

XML Code:	`<table:detective>`
Rules:	
DTD:	`<!ELEMENT table:detective (table:highlighted-range*, table: operation*)>`

The elements that can be contained in the `<table:detective>` element are:

- Highlighted range
- Operation

4.7.6 Highlighted Range

The highlighted range element specifies a range cell address for the range and whether or not the range contains an error.

XML Code:	`table:highlighted-range`
Rules:	
DTD:	`<!ELEMENT table:highlighted-range EMPTY>`

The attributes associated with the `table:highlighted-range` element are:

- Cell Range Address
- Direction
- Contains Error

Cell Range Address

This attribute specifies the cell range address of the highlighted range.

XML Code:	`table:cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:cell-range-address %cell-range-address; #IMPLIED>`

Direction

This attribute specifies the direction of the arrow between this cell and the highlighted range.

XML Code:	`table:direction`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:direction (from-another-table | to-another-table | from-same-table) #REQUIRED>`

Contains Error

This attribute specifies whether or not the cell range contains an error.

XML Code:	`table:contains-error`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:contains-error %boolean; "false">`

4.7.7 Operation

The operation element contains the operation and its index. Using the index, the application can restore the order of the operations of all cells.

XML Code:	`table:operation`
Rules:	
DTD:	`<!ELEMENT table:operation EMPTY>`

The attributes associated with the `table:operation` element are:

- Name
- Index

Name

This attribute specifies the name of the operation. Possible names are `trace-dependents` , `remove-dependents`, `trace-precedents`, `remove-precedents`, and `trace-errors`.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:operation table:name (trace-dependents|remove-dependents|trace-precedents|remove-precedents|trace-errors) #REQUIRED>`

Index

This attribute specifies the index in the order of the operations of all cells.

XML Code:	`table:index`
Rules:	
DTD:	`<!ATTLIST table:operation table:index %nonNegativeInteger; #REQUIRED>`

4.7.8 Cell Address Entity

A special data type exists for the address of a cell. A cell address entity describes a cell address.

The structure of a cell address is as follows:

1. The name of the table.
2. A dot (.).
3. An alphabetic value representing the column. The letter A represents column 1, B represents column 2, and so on. AA represents column 27, AB represents column 28, and so on.
4. A numeric value representing the row. The number 1 represents the first row, the number 2 represents the second row, and so on.

   A1 represents the cell in column 1 and row 1. B1 represents the cell in column 2 and row 1. A2 represents the cell in column 1 and row 2.

For example, if you have a table with the name `SampleTable` and you want to address the cell in column 34 and row 16, the address is `SampleTable.AH16`. In some cases it is not necessary to provide the name of the table. However, the dot must be present. When the table name is not required, the address in the previous example is `.AH16`.

The structure of the address of a cell in a subtable is as follows:

1. The address of the cell that contains the subtable.

2. A dot (.).
3. The address of the cell in the subtable.

For example, to reference the cell in column 1 and row 1 in a subtable that is called `Subtable`, and that is in column 34 and row 16 of the table `SampleTable`, the address is `SampleTable.AH16.A1`. If the name of the table contains a blank the name should be between two quotation marks. This quotation marks should be quoted.

Absolute and relative cell addressing

You can reference cells in two ways, using absolute addresses or relative addresses. When you perform an operation on a table cell, for example when you copy a formula, absolute cell references do not change; relative cell references do change. The previous example uses relative addressing.

To create an absolute address, place a dollar sign ($) before each table name, column reference, and row reference. For example, the absolute address of the previous example is `$SampleTable.$AH$16`. You can combine absolute and relative references in a cell address. For example, you can use `SampleTable.AH$16` to refer to a relative table and column, and an absolute row. Absolute addresses must contain a table name. The discrimination between absolute and relative addressing is only necessary in some special cases. Otherwise, a cell reference without the dollar signs is used.

XML Code:	`cell-address`
Rules:	
DTD:	`<!ENTITY % cell-address CDATA>`

4.7.9 Cell Range Address Entity

A cell range is a number of adjacent cells forming a rectangular shape. The rectangle stretches from the cell on the top left to the cell on the bottom right.

The range address of cells has a special data type. To specify a cell range address, you use an entity. To reference a range of adjacent cells, construct the address as follows, in the specified order:

1. The address of the cell at the top left of the range you want to reference.
2. A colon (:).
3. The address of the cell at the bottom right of the range you want to reference.

For example, you use the address `.A1:.B2` to reference the range of cells from column 1 and row 1 to column 2 and row 2. The smallest range you can specify is a single cell. In this case, the range address is the same as the cell address.

XML Code:	`cell-range-address`
Rules:	
DTD:	`<!ENTITY % cell-range-address CDATA>`

4.7.10 Cell Range Address List Entity

A cell range address list is a list of cell ranges or cell addresses , or both. Each item in the list is separated by a space.

XML Code:	`cell-range-address-list`
Rules:	
DTD:	`<!ENTITY % cell-range-address-list CDATA>`

4.8 Table Cell Content Validations

The `<table:content-validations>` element contains all of the cell content validations.

XML Code:	`<table:content-validations>`
Rules:	
DTD:	`<!ELEMENT table:content-validations (table:content-validation) *>`

4.8.1 Table Cell Content Validation

The `<table:content-validation>` element specifies the validation of the content of the cell with this style.

XML Code:	`<table:content-validation>`
Rules:	If this element is not present, the cell can contain any content.
DTD:	`<!ELEMENT table:content-validation (table:help-message?, (table:error-message | table:error-macro)?)>`

The attributes that you can associate with the `<table:content-validation>` element are:

- Name
- Condition
- Base Cell Address
- allow empty cell

Name

The `table:name` attribute specifies the name of the content validation. The name is created automatically by the application.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:content-validation table:name CDATA #REQUIRED>`

Condition

The condition attribute specifies the condition which cell content is allowed.

XML Code:	`table:condition`

Rules:	The value of this attribute is a Boolean expression. The syntax of the expression is similar to the XPath syntax. The following are valid conditions: • `Condition ::= ExtendedTrueCondition | TrueFunction 'and' TrueCondition` • `TrueFunction ::= cell-content-is-whole-number() | cell-content-is-decimal-number() | cell-content-is-date() | cell-content-is-time()` • `ExtendedTrueCondition ::= ExtendedGetFunction | cell-content-text-length() Operator Value` • `TrueCondition ::= GetFunction | cell-content() Operator Value` • `GetFunction ::= cell-content-is-between(Value, Value) | cell-content-is-not-between(Value, Value)` • `ExtendedGetFunction ::= cell-content-text-length-is-between (Value, Value) | cell-content-text-length-is-not-between (Value, Value)` • `Operator ::= '<' | '>' | '<=' | '>=' | '=' | '!='` • `Value ::= NumberValue | String | Formula` A `Formula` is a formula without an equals (=) sign at the beginning. See Section 4.7.2 for more information. A `String` comprises one or more characters surrounded by quotation marks. A `NumberValue` is a whole or decimal number. You must include an `Operator`. The number in a `NumberValue` or `Formula` cannot contain comma separators for numbers of 1000 or greater.
DTD:	`<!ATTLIST table:content-validation table:condition CDATA #IMPLIED>`

Base Cell Address

The `table:base-cell-address` attribute specifies the address of the base cell for relative addresses in formulas.

XML Code:	`table:base-cell-address`
Rules:	This attribute is only necessary when the condition contains a formula. The value of this attribute must be an absolute cell address with a table name.
DTD:	`<!ATTLIST table:content-validation table:base-cell-address %cell-address; #IMPLIED>`

Allow Empty Cell

The `table:allow-empty-cell` attribute specifies whether or not a cell can be empty.

XML Code:	`table:allow-empty-cell`
Rules:	
DTD:	`<!ATTLIST table:content-validation table:allow-empty-cell % boolean; #IMPLIED>`

4.8.2 Help Message

The `<table:help-message>` element specifies a message to display if a user selects the cell.

XML Code:	`<table:help-message>`
Rules:	
DTD:	`<!ELEMENT table:help-message (text:p*)>`

The attributes that you can associate with the `<table:help-message>` element are:

- Title
- Display

Title

The `table:title` attribute specifies the title of the help message.

XML Code:	`table:title`
Rules:	
DTD:	`<!ATTLIST table:help-message table:title CDATA #IMPLIED>`

Display

The `table:display` attribute specifies whether or not to display the message.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:help-message table:display %boolean; #IMPLIED>`

4.8.3 Error Message

The `<table:error-message>` element specifies a message to display if a user tries to include unacceptable content in a cell.

XML Code:	`<table:error-message>`
Rules:	
DTD:	`<!ELEMENT table:error-message (text:p*)>`

The attributes that you can associate with the `<table:error-message>` element are:

- Title
- Message Type
- Display

Title

The `table:title` attribute specifies the title of the error message.

XML Code:	`table:title`
Rules:	
DTD:	`<!ATTLIST table:error-message table:title CDATA #IMPLIED>`

Message Type

The `table:message-type` attribute specifies the type of error message.

XML Code:	`table:message-type`
Rules:	The value of this attribute can be `stop`, `warning`, or `information`.
DTD:	`<!ATTLIST table:error-message table:message-type ( stop | warning | information ) #IMPLIED>`

Display

The `table:display` attribute specifies whether or not to display the message.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:error-message table:display %boolean; #IMPLIED>`

4.8.4 Error Macro

The `<table:error-macro>` element specifies a macro that is executed when a user tries to included unacceptable content in a cell.

XML Code:	`<table:error-macro>`
Rules:	
DTD:	`<!ELEMENT table:error-macro EMPTY>`

The attributes that you can associate with the `<table:error-macro>` element are:

- Name
- Execute

Name

The `table:name` attribute specifies the name of the macro.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:error-macro table:name CDATA #IMPLIED>`

Execute

The `table:execute` attribute specifies whether or not to execute the macro.

XML Code:	`table:execute`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST table:error-macro table:execute %boolean; #IMPLIED>`

4.9 Subtables

A subtable is a table within another table. It occupies one cell and no other content can appear in this cell. If a table cell contains a subtable, it cannot contain any paragraphs.

XML Code:	`<table:sub-table>`
Rules:	The borders of a subtable merge with the borders of the cell that it resides in. A subtable does not contain any formatting properties. A subtable is essentially a container for some additional table rows that integrate seamlessly with the parent table.
DTD:	`<!ELEMENT table:sub-table (%table-column-groups;,%table-row-groups;)>`
Notes:	There is a difference between a subtable and a HTML table that is nested within another HTML table. A nested HTML table appears as a table within a table, that is, it has borders distinct from those of the parent cell and respects the padding of the parent cell.

Example of Representation of Subtable

StarOffice XML can represent this table in either of the ways detailed in Sample 1 and Sample 2.

A1	**B1**	**C1**
A2	B2.1.1	B2.2.1
	B2.1.2	

Sample 1

StarOffice XML can describe the preceding table as follows:

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm" fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
  <style:properties fo:background-color="grey"/>
</style:style>
<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-header-rows>
    <table:table-row table:style-name="Row1">
      <table:table-cell>
        <text:p text:style="Table Caption">
          A1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          B1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          C1
        </text:p>
      </table:table-cell>
    </table:table-row>
  </table:table-header-rows>
  <table:table-rows>
    <table:table-row>
      <table:table-cell table:number-rows-spanned="2" table:style-name="Cell1">
        <text:p text:style="Table Body">
          A2
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Body">
          B2.1.1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Body">
          B2.2.1
        </text:p>
      </table:table-cell>
    </table:table-row>
    <table:table-row>
      <table:covered-table-cell/>
      <table:table-cell table:number-columns-spanned="2">
        <text:p text:style="Table Body">
          B2.1.2
        </text:p>
      </table:table-cell>
      <table:covered-table-cell/>
    </table:table-row>
  </table:table-rows>
</table:table>
```

Sample 2

This sample ignores the borders of the table. StarOffice XML can describe the preceding table as follows:

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm" fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
  <style:properties fo:background-color="grey"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-header-rows>
    <table:table-row table:style-name="Row1">
      <table:table-cell>
        <text:p text:style="Table Caption">
          A1
        </text:p>
      </table:table.cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          B1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          C1
        </text:p>
      </table:table-cell>
    </table:table-row>
  </table:table-header-rows>
  <table:table-rows>
    <table:table-row>
      <table:table-cell table:style-name="Cell1">
        <text:p text:style="Table Body">
          A2
        </text:p>
      </table:table-cell>
      <table:table-cell table:number-columns-spanned="2">
```

```
        <table:subtable>
          <table:table-columns>
            <table:table-column table:style-name="Col2"/>
            <table:table-column table:style-name="Col3"/>
          </table:table-columns>
              <table:table-cell>
                <text:p text:style="Table Body">
                  B2.1.1
                </text:p>
              </table:table-cell>
              <table:table-cell>
                <text:p text:style="Table Body">
                  B2.2.1
                </text:p>
              </table:table-cell>
            </table:table-row>
            <table:table-row>
              <table:table-cell table:number-columns-spanned="2">
                <text:p text:style="Table Body">
                  B2.1.2
                </text:p>
              </table:table-cell>
              <table:covered-table-cell/>
            </table:table-row>
          </table:table-rows>
        </table:subtable>
      </table:table-cell>
      <table:covered-table-cell/>
    </table:table-row>
  </table:table-rows>
</table:table>
```

4.10 Label Ranges

The label ranges element contains a collection of label ranges.

XML Code:	`<table:label-ranges>`
Rules:	
DTD:	`<!ELEMENT table:label-ranges (table:label-range)* >`

4.10.1 Label Range

The label range element specifies a cell range which contain the labels and a cell range, which contains the data. There are two types of label ranges.

- One for columns
- One for rows.

The range of the data should have either the same height and vertical position like the label range of rows or the same width and horizontal position like the label range of columns. For information on defining a cell range, see Section 4.7.5.

XML Code:	`<table:label-range>`
Rules:	
DTD:	`<!ELEMENT table:label-range EMPTY>`

The attributes associated with the label range element are:

- Label cell range address
- Data cell range address
- Orientation

Label Cell Range Address

This attribute specifies the cell range address of the labels.

XML Code:	`table:label-cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:label-range table:label-cell-range-address %cell-range-address; #REQUIRED>`

Data Cell Range Address

This attribute specifies the cell range address of the data.

XML Code:	`table:data-cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:label-range table:data-cell-range-address %cell-range-address; #REQUIRED>`

Orientation

This attribute specifies the orientation of the label range.

XML Code:	`table:orientation`
Rules:	This attribute can have a value of `column` or `row`.
DTD:	`<!ATTLIST table:range table:orientation (column|row) #REQUIRED>`

4.11 Named Expressions

The named expressions element contains a collection of expressions with names, which you can use to refer to the expression.

XML Code:	`<table:named-expressions>`
Rules:	
DTD:	`<!ELEMENT table:named-expressions (table:named-range|table:named-expression)* >`

An expression element can be used to represent:

- A named cell range.
- Other expressions, for example, part of a formula.

4.11.1 Named Range

The named range element specifies a cell range with a name. For information on defining a cell range, see Section 4.7.5.

XML Code:	`<table:named-range>`
Rules:	If the cell range address is relative, the `table:base-cell-address` attribute must be attached to this element. If the named expression is a cell range a attribute that contains the cell area is necessary.
DTD:	`<!ELEMENT table:named-range EMPTY>`
Note:	A Named Range is one case where a discrimination between absolute and relative addresses is possible.

The attributes associated with the named range element are:

- Name
- Cell range address
- Base cell address
- Range usable as

Name

XML Code:	`table:name`
Rules:	This attribute can be attached to the `<table:named-range>` and `<table:named-expression>` elements.
DTD:	`<!ATTLIST table:named-range table:name CDATA #REQUIRED>`

Cell Range Address

This attribute specifies the cell range address.

XML Code:	`table:cell-range-address`
Rules:	This attribute can be attached to the `<table:named-range>` element.
DTD:	`<!ATTLIST table:named-range table:cell-range-address %cell-range-address; #REQUIRED>`

Base Cell Address

This attribute specifies the base cell address if the cell address or the cell range address in the named range is relative.

XML Code:	`table:base-cell-address`
Rules:	This attribute can be attached to the `<table:named-range>` element.
DTD:	`<!ATTLIST table:named-range table:base-cell-address %cell-address; #IMPLIED>`
Note:	Discrimination between absolute and relative addressese is not possible. Therefore a table name in the address will needed and dollar signs will ignored.

Range usable as

This attribute specifies the possible usage of the named range. The named range can be used as a Print Range, a Filter, a Repeat Row, or a Repeat Column.

XML Code:	`table:range-usable-as`
Rules:	This attribute can be attached to the `<table:named-range>` element. The value of this attribute can be either: • `none` or • a space-separated list that consists of any of the values `print-range`, `filter`, `repeat-row` or `repeat-column`.
DTD:	`<!ATTLIST table:named-range table:range-usable-as CDATA "none">`

4.11.2 Named Expression

The named expression element contains an expression with a name, for example, part of a formula.

XML Code:	`<table:named-expression>`
Rules:	Expressions do not support the equal (=) sign as the first character. If the element contains a named range or another named expression, the named range or named expression must be specified first, before the containing expression.
DTD:	`<!ELEMENT table:named-expression EMPTY>`

The attributes associated with the named expression element are:

- Name (the usage of this attribute is the same as for the `<table:named-range>` element, see Section 4.11.1.)
- Expression
- Base cell address (the usage of this attribute is the same as for the `<table:named-range>` element, see Section 4.11.1.)

Expression

XML Code:	`table:expression`
Rules:	This attribute can be attached to the `<table:named-expression>` element.
DTD:	`<!ATTLIST table:named-expression table:expression CDATA #REQUIRED>`

Example: Named expressions element with a named range and a named expression

```
<table:named-expressions>
  <table:named-range table:name="sample1" table:cell-range-address=".C4"
table:base-cell-address="sampletable.F1" table:area-type="none"/>
  <table:named-range table:name="sample2" table:cell-range-address=".$D$3:.
$K$8" table:area-type="print-range filter"/>
  <table:named-expression table:name="sample3" table:expression="sum([.A1:.
B3])"/>
</table:named-expressions>
```

4.12 Filters

4.12.1 Table Filter

The table filter element describes how to filter the data in a database range or datapilot tables.

XML Code:	`<table:filter>`
Rules:	
DTD:	`<!ELEMENT table:filter ( table:filter-condition | table:filter-and | table:filter-or ) >`

Target Range Address

This attribute specifies where the result of the filter is output.

XML Code:	`table:target-range-address`
Rules:	This attribute can be attached to the `<table:filter>` and `<table:sort>` elements. If the attribute is not present, the data is output in the source range.
DTD:	`<!ATTLIST table:filter table:target-range-address %cell-range-address; #IMPLIED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

Condition Source Range Address

This attribute specifies the cell range from which the filter condition can get its data.

XML Code:	`table:condition-source-range-address`
Rules:	This attribute can be attached to the `<table:filter>` element.
DTD:	`<!ATTLIST table:filter table:condition-source-range-address %cell-range-address; #IMPLIED>`

Condition Source

This attribute specifies the source location from where the application gets the filter condition.

XML Code:	`<table:condition-source>`
Rules:	This attribute can have one of the following values: • `self` The application gets the filter condition from the `<table:filter>` element. • `cell-range` The application gets the filter condition from the cell range specified by the `table:condition-source-range-address` attribute.
DTD:	`<!ATTLIST table:filter table:condition-source ("self" | "cell-range") "self")`

Display Duplicates

A display duplicates attribute specifies whether or not to display duplicate matches in the result.

XML Code:	`table:display-duplicates`
Rules:	
DTD:	`<!ATTLIST table:filter table:display-duplicates %boolean; "true">`

4.12.2 Filter And

The `filter-and` element specifies whether the logical operator AND is used in a filter.

XML Code:	`<table:filter-and>`
Rules:	
DTD:	`<!ELEMENT table:filter-and ( table:filter-or | table:filter-condition )+ >`

4.12.3 Filter Or

The `filter-or` element specifies whether the logical operator OR is used in a filter.

XML Code:	`<table:filter-or>`
Rules:	
DTD:	`<!ELEMENT table:filter-or ( table:filter-and | table:filter-condition )+ >`

4.12.4 Filter Condition

The table filter condition element describes a condition to apply in a filter operation.

XML Code:	`<table:filter-condition>`
Rules:	
DTD:	`<!ELEMENT table:filter-condition EMPTY >`

Field Number

A field number attribute specifies which field to use for the condition. An example of a field number is a row or column number, the condition being the orientation of the table.

XML Code:	`table:field-number`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:field-number CDATA #REQUIRED>`

Case Sensitive

A case sensitive attribute determines whether a filter condition is case sensitive.

XML Code:	`table:case-sensitive`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:case-sensitive %boolean; "false">`

Data Type

A data type attribute specifies what data type to use for the filter condition.

XML Code:	`table:data-type`
Rules:	

DTD:	`<!ATTLIST table:filter-condition`
	`table:data-type ("text" | "number") "text">`

Value

A value attribute specifies a value for the filter condition.

XML Code:	`table:value`
Rules:	

DTD:	`<!ATTLIST table:filter-condition`
	`table:value CDATA #REQUIRED>`

Operator

An operator attribute specifies what operator to use in the filter condition.

XML Code:	`table:operator`
Rules:	The possible values for the operator attribute depend on whether the condition uses a regular expression. If the condition includes a regular expression there are only two possible values: • match (matches) • !match (does not match) If the condition does not include a regular expression, the possible values are the following conventional relational operators: • = (Equal to) • != (Not equal to) • < (Less than) • > (Greater than) • <= (Less than or equal to) • >= (Greater than or equal to) In addition, you can use non-conventional operators such as "empty" and "!empty", "bottom values", "top values", "bottom percent", and "top percent". For example, you can use the latter two operators to filter the lowest and highest percentage of entries.

DTD:	`<!ATTLIST table:filter-condition`
	`table:operator CDATA #REQUIRED>`

Example:Representation of a filter

```
<filter>
 <filter-or>
  <filter-and>
   <filter-condition table:field-number=1 table:operator="=" table:
value="Doe"/>
   <filter-condition table:field-number=2 table:operator="=" table:
value="John"/>
  </filter-and>
  <filter-and>
   <filter-condition table:field-number=1 table:operator="=" table:
value="Burns"/>
   <filter-condition table:field-number=2 table:operator="=" table:
value="Michael"/>
  </filter-and>
 </filter-or>
</filter>
```

4.13 Database Ranges

The Database Ranges element contains a collection of Database Ranges.

XML Code:	`<table:database-ranges>`
Rules:	
DTD:	`<!ELEMENT table:database-ranges (table:database-range)* >`

Database Range

A database range is a named area in a table where you can perform database operations.

XML Code:	`<table:database-range>`
Rules:	
DTD:	`<!ELEMENT table:database-range ((table:database-source-sql| table:database-source-table| table:database-source-query)?, table:filter?, table:sort?, table:subtotal-rules?)>`

Database Range Name

A database range name attribute specifies the name of the database range to perform operations on. Only one database range can be without a name. This database range is usually created by the application and is used to filter or sort data in a cell range without the user creating a database range.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:database-range table:name CDATA #IMPLIED>`

Is Selection

An is selection attribute specifies whether the database range includes the complete database, or a selection of records from the database.

XML Code:	`table:is-selection`
Rules:	

DTD:	`<!ATTLIST table:database-range`
	`table:is-selection %boolean; "false">`

On Update Keep Styles

An on update keep styles attribute specifies the behavior of the cell styles if the data in the data source changes.

XML Code:	`table:on-update-keep-styles`
Rules:	If the attribute value is true, the style of the new cells in the database range is the same as the other cells in the column. If the attribute value is false, the style of the new cells in the database range is the standard style of the document.

DTD:	`<!ATTLIST table:database-range`
	`table:on-update-keep-styles %boolean; "false">`

On Update Keep Size

An on update keep size attribute specifies the behavior of the database range if the size of the data in the data source changes.

XML Code:	`table:on-update-keep-size`
Rules:	If the attribute value is true, the range retains its size. If the attribute value is false, the range does not retain its size.

DTD:	`<!ATTLIST table:database-range`
	`table:on-update-keep-size %boolean; "true">`

Has Persistent Data

A persistent data attribute specifies whether or not to store the data in a database range.

XML Code:	`has-persistent-data`
Rules:	

DTD:	`<!ATTLIST table:database-range`
	`table:has-persistent-data %boolean; "true">`

Orientation

An orientation attribute specifies the orientation of the database range.

XML Code:	`table:orientation`
Rules:	The orientation of a database range can be by row or by column. The only possible values for this attribute are `row` and `column`.

DTD:	`<!ATTLIST table:database-range`
	`table:orientation ( "row" | "column" ) "row">`

Contains Header

A contains header attribute specifies whether or not the database range contains a header.

XML Code:	`table:contains-header`
Rules:	
DTD:	`<!ATTLIST table:database-range table:contains-header %boolean; "true">`

Display Filter Buttons

A display filter buttons attribute specifies whether or not to display filter buttons.

XML Code:	`table:display-filter-buttons`
Rules:	
DTD:	`<!ATTLIST table:database-range table:display-filter-buttons %boolean; "false">`

Target Range Address

This attribute specifies the cell range address of the database range.

XML Code:	`table:target-range-address`
Rules:	This attribute can be attached to the `<table:database-range>` element.
DTD:	`<!ATTLIST table:database-range table:target-range-address %cell-range-address; #REQUIERED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, a table name must be specified in the address and dollar signs are ignored.

4.13.1 Database Source SQL

This element describes an SQL database that integrates with the table.

XML Code:	`<table:database-source-sql>`
Rules:	
DTD:	`<!ELEMENT table:database-source-sql EMPTY?>`

Database Name

A database name attribute specifies the name of the SQL database that the data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:database-name CDATA #REQUIRED>`

SQL Statement

An SQL statement attribute specifies the SQL statement to use when importing data from an SQL database.

XML Code:	`table:sql-statement`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:sql-statement CDATA #REQUIRED>`

Parse SQL Statement

A parse SQL statement attribute specifies whether or not the application will parse SQL statements.

XML Code:	`table:parse-sql-statement`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:parse-sql-statement %boolean; "false">`

4.13.2 Database Source Table

The database source table element contains the information about the included database and the table.

XML Code:	`<table:database-source-table>`
Rules:	
DTD:	`<!ELEMENT table:database-source-table EMPTY?>`

Database Name

A database name attribute specifies the name of the database that data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-table table:database-name CDATA #REQUIRED>`

Table Name

A table name attribute specifies the table that data is imported from.

XML Code:	`table:table-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-table table:table-name CDATA #REQUIRED>`

4.13.3 Database Source Query

The database source query element contains the information about the included database and the query.

XML Code:	`<table:database-source-query>`
Rules:	
DTD:	`<!ELEMENT table:database-source-query EMPTY?>`

Database Name

A database name attribute specifies a database that data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-query table:database-name CDATA #REQUIRED>`

Query Name

A query name attribute specifies the query to perform on the database whose data is being imported.

XML Code:	`table:query-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-query table:query-name CDATA #REQUIRED>`

4.13.4 Sort

The sort element describes the sort keys in a database range.

XML Code:	`<table:sort>`
Rules:	
DTD:	`<!ELEMENT table:sort (table:sort-by)+ >`

Bind Styles to Content

A bind styles to content attribute specifies whether or not cells retain their style features after a sort operation.

XML Code:	`table:bind-styles-to-content`
Rules:	
DTD:	`<!ATTLIST table:sort table:bind-styles-to-content %boolean; "true">`

Target Range Address

This attribute specifies where the result of the sort is put. The attribute is used with the `<table:sort>` element in the same way as it is used with the `<table:filter>` element. Please see Section 4.12.1 for information on using this attribute.

Case Sensitive

A case sensitive attribute specifies whether or not sorting is case sensitive.

XML Code:	`table:case-sensitive`
Rules:	
DTD:	`<!ATTLIST table:sort table:case-sensitive %boolean; "false">`

4.13.5 Sort By

The sort by element specifies which field to sort, the data type of this field, and how to sort it.

XML Code:	`<table:sort-by>`
Rules:	
DTD:	`<!ELEMENT table:sort-by EMPTY >`

Field Number

A field number attribute specifies the row or column number to sort by.

XML Code:	`table:field-number`
Rules:	
DTD:	`<!ATTLIST table:sort-by table:field-number CDATA #REQUIRED>`

Data Type

A data type attribute specifies the data type of the field to be sorted.

XML Code:	`table:data-type`
Rules:	If the attribute value is `automatic`, the application must determine what type of data is in the field. If the field contains a user-defined data type, the attribute value is the name of this data type.

DTD:	`<!ATTLIST table:sort-by table:data-type`
	`("text" | "number" | "automatic" | qname-but-not-ncname )`
	`"automatic" >`

Order

An order attribute specifies whether to sort the data in ascending or descending order.

XML Code:	`table:order`
Rules:	

DTD:	`<!ATTLIST table:sort-by`
	`table:order ( "ascending" | "descending" ) "ascending" >`

4.13.6 Subtotal Rules

The subtotal rules element contains the following information:

- The provisional result of a field in a database range, for example, a column.
- The function used to calculate the provisional result.

The element consists of generated groups of fields in the database range. For example, all cells with the same content in the same field form a group.

XML Code:	`<table:subtotal-rules>`
Rules:	

DTD:	`<!ELEMENT table:subtotal-rules`
	`( table:sort-groups? | table:subtotal-rule*) >`

Bind Styles To Content

A bind style to content attribute specifies whether or not cells retain their style features after a subtotal operation.

XML Code:	`table:bind-styles-to-content`
Rules:	This attribute is only evaluated if the `table:sort-groups` element is present.

DTD:	`<!ATTLIST table:subtotal-rules`
	`table:bind-styles-to-content %boolean; "true">`

Case Sensitive

A case sensitive attribute specifies whether or not the case of characters is important when comparing entries, for example, when sorting groups.

XML Code:	`table:case-sensitive`
Rules:	This attribute only evaluates if the `table:sort-groups` element is present.

DTD:	`<!ATTLIST table:subtotal-rules`
	`table:case-sensitive %boolean; "false">`

Page Breaks On Group Change

A page breaks on group change attribute specifies whether or not to insert a page break after the subtotal for each group.

XML Code:	`table:page-breaks-on-group-change`
Rules:	This attribute only evaluates if the `table:sort-groups` element is present.

DTD:	`<!ATTLIST table:subtotal-rules`
	`table:page-breaks-on-group-change %boolean; "false">`

4.13.7 Sort Groups

The sort groups element specifies whether to sort column groups or row groups, and how to sort them. It belongs to the subtotal rules element, see the previous section.

XML Code:	`<table:sort-groups>`
Rules:	
DTD:	`<!ELEMENT table:sort-groups EMPTY >`

Data Type

A data type attribute specifies the data type of the column group or row group to sort.

XML Code:	`table:data-type`
Rules:	If the attribute value is `automatic`, the application must determine what type of data is in the group. If the group contains a user-defined data type, the attribute value is the name of this data type.

DTD:	`<!ATTLIST table:sort-groups table:data-type`
	`("text" | "number" | "automatic" | qname-but-not-ncname )`
	`"automatic" >`

Order

An order attribute specifies whether to sort the group data in ascending or descending order.

XML Code:	`table:order`
Rules:	

DTD:	`<!ATTLIST table:sort-groups`
	`table:order ( "ascending" | "descending" ) "ascending" >`

4.13.8 Subtotal Rule

The subtotal rule element contains a rule for one field, for example, a column. The rule contains the group field number, which specifies the column group for which the rule is used, and one or more subtotal fields, which specify a field and the function of the field. In summary, the rule describes how to calculate the subtotal.

XML Code:	`<table:subtotal-rule>`
Rules:	
DTD:	`<!ELEMENT table:subtotal-rule (table:subtotal-field)* >`

Group By Field Number

A group by field number attribute specifies the field, for example, a column, that is to be grouped.

XML Code:	`table:group-by-field-number`
Rules:	This attribute can be used with the `<table:subtotal-rule>` element.

DTD:	`<!ATTLIST table:subtotal-rule`
	`table:group-by-field-number CDATA #REQUIRED >`

4.13.9 Subtotal Field

The subtotal field element contains the field number and the function that is used to calculate a provisional result. An example of a field is a column.

XML Code:	`<table:subtotal-field>`
Rules:	
DTD:	`<!ELEMENT table:subtotal-field EMPTY >`

Field Number

A field number attribute specifies the index number of the field.

XML Code:	`table:field-number`
Rules:	

DTD:	`<!ATTLIST table:subtotal-field`
	`table:field-number CDATA #REQUIRED>`

Function

A function attribute specifies what kind of subtotals to calculate. The following are possible values for this attribute: "`auto`", "`average`", "`count`", "`countnums`", "`max`", "`min`", "`product`", "`stdev`", "`stdevp`", "`sum`", "`var`" and "`varp.`"

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:subtotal-field table:function CDATA #REQUIRED>`

Example: Subtotal field

```
<table:database-range table:range-position="sampletable.A1:sampletable.G20"
table:name="sample">
  <table:database-source-table table:database-name="sampleDB" table:table-
name="sampleTable"/>
  <table:filter ...>

  </table:filter>
  <table:sort>
    <table:sort-by table:field-number=1>
  </table:sort>
  <table:subtotal-rules>
    <table:sort-groups/>
    <table:subtotal-rule table:column-group "3">
      <table:subtotal-field table:field-number="1" table:function="sum"/>
    </table:subtotal-rule>
  </table:subtotal-rules>
</table:database-range>
```

4.14 Data Pilot Tables

Data pilot tables allow you to analyze and evaluate your data. The data pilot tables element can contain several data pilot tables.

XML Code:	`<table:data-pilot-tables>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-tables (table:data-pilot-table)* >`

4.14.1 Data Pilot Table

XML Code:	`<table:data-pilot-table>`
Rules:	This element can contain *one* of the following elements: • `<table:database-source-sql>` (see Section 4.13.1) • `<table:database-source-table>` (see Section 4.13.2) • `<table:database-source-query>` (see Section 4.13.3) • `<table:source-service>` (see Section 4.14.2) • `<table:source-cell-range>` (see Section 4.14.3)
DTD:	`<!ELEMENT table:data-pilot-table ((table:database-source-sql | table:database-source-table | table:database-source-query | table:source-service | table:source-cell-range), table:data-pilot-field+)>`

The attributes associated with the data pilot table element are:

- Data pilot table name
- Application data

- Grand total
- Ignore empty rows
- Identify categories
- Target range address

Data Pilot Table Name

This attribute specifies the name of the data pilot table.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:name CDATA #REQUIRED>`

Application Data

This attribute specifies extra information about the data pilot table, which can be used by the application.

XML Code:	`table:application-data`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:application-data CDATA #IMPLIED>`

Grand Total

This attribute specifies if a column, row, or both have a grand total or if there is a grand total at all.

XML Code:	`table:grand-total`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:grand-total ( "no" | "row" | "column" | "both" ) "both" >`

Ignore Empty Rows

This attribute specifies whether or not empty rows in the source range should be ignored.

XML Code:	`table:ignore-empty-rows`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:ignore-empty-rows % boolean; "false">`

Identify Categories

This attribute specifies whether or not the application orders rows without labels to the next higher category specified by a row label.

XML Code:	`table:identify-categories`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:identify-categories %boolean; "false">`

Target Range Address

This attribute specifies where the target range of the data pilot table output.

XML Code:	`table:target-range-address`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:target-range-address %cell-range-address; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

Buttons

This attribute specifies all cells which are a button. This is a list of cell-addresses.

XML Code:	`table:buttons`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:buttons %cell-range-address-list; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

4.14.2 Source Service

A source service element contains information about the service which is used to create the data pilot table.

XML Code:	`<table:source-service>`
Rules:	
DTD:	`<!ELEMENT table:source-service EMPTY >`

The attributes associated with this element are:

- Service name
- Source name
- Object name
- Source username
- Source password

Service Name

This attribute specifies the name of the service.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:name CDATA #REQUIRED >`

Source Name

This attribute specifies the source of the service.

XML Code:	`table:source-name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:source-name CDATA #REQUIRED >`

Object Name

This attribute specifies the name of the object in the source which contains the data.

XML Code:	`table:object-name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:object-name CDATA #REQUIRED >`

Source Username

This attribute specifies the username required to access the source.

XML Code:	`table:username`
Rules:	
DTD:	`<!ATTLIST table:source-service table:username CDATA #IMPLIED >`

Source Password

This attribute specifies the password required to access the source.

XML Code:	`table:password`
Rules:	
DTD:	`<!ATTLIST table:source-service table:password CDATA #IMPLIED >`

4.14.3 Source Cell Range

Asource cell range element contains information about the cell range and how the data pilot table gets the data from the range. You can acquire the data with or without a query.

XML Code:	`<table:source-cell-range>`
Rules:	
DTD:	`<!ELEMENT table:source-cell-range (table:filter)? >`

The attributes associated with the source cell range element is:

- Cell range address

Cell Range Address

This attribute specifies the cell range containing the source data.

XML Code:	`table:cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:source-cell-range table:cell-range-address % cell-range-address; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

4.14.4 Filter

See Section 4.12.1 for information on filtering in tables.

4.14.5 Data Pilot Field

The data pilot field element..... *information to be supplied.*

XML Code:	`<table:data-pilot-field>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-field ( table:data-pilot-level ) ? >`

The attributes associated with the data pilot field element are:

- Source field name
- Is data layout field
- Function
- Orientation
- Used hierarchy

Source Field Name

This attribute specifies the name of the source field. There can be multiple `<table:data-pilot-field>` elements with the same value for this attribute.

XML Code:	`table:source-field-name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:source-field-name CDATA #REQUIRED >`

Is Data Layout Field

This attribute specifies whether or not the source field is a data layout field.

XML Code:	`table:is-data-layout-field`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:is-data-layout-field %boolean; "false" >`

Function

This attribute specifies the function which is used for the source field. Possible values for this attribute are: "`average`", "`count`", "`countnums`", "`max`", "`min`", "`product`", "`stdev`", "`stdevp`", "`sum`", "`var`" and "`varp`".

XML Code:	`table:function`
Rules:	This attribute is only evaluated if the value of the `table:orientation` attribute is "`data`".
DTD:	`<!ATTLIST table:data-pilot-field table:function CDATA #REQUIRED >`

Orientation

This attribute specifies the orientation of the source field. The orientation can be by row, by column, by data, by page, or hidden.

XML Code:	`table:orientation`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:orientation ( "row" | "column" | "data" | "page" | "hidden" ) #REQUIRED>`

Used Hierarchy

This attribute specifies the used hierarchy of the source field.

XML Code:	`table:used-hierarchy`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:used-hierarchy CDATA "1" >`

4.14.6 Data Pilot Level

The data pilot level element contains information about the level of a data pilot table.

XML Code:	`<table:data-pilot-level>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-level (table:data-pilot-subtotals?, table:data-pilot-members?) >`

The attribute associated with the data pilot level element is:

- Show empty

Show Empty

This attribute specifies whether or not empty fields should be displayed. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:show-empty`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-level table:show-empty %boolean; #IMPLIED >`

4.14.7 Data Pilot Subtotals

The data pilot subtotals element contains information about the provisional result of a field in a data pilot table and the function used to calculate the result. If the element is not present, the application can determine the subtotals with the help of the source.

XML Code:	`<table:data-pilot-subtotals>`
Rules:	This element can contain the data pilot subtotal elements.
DTD:	`<!ELEMENT table:data-pilot-subtotals ( table:data-pilot-subtotal )* >`

4.14.8 Data Pilot Subtotal

Thedata pilot subtotal element contains the function which is used to calculate the subtotal.

XML Code:	`<table:data-pilot-subtotal>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-subtotal EMPTY >`

The attribute associated with the data pilot subtotal element is:

- Function

Function

This attribute specifies the function used for the subtotal. Possible functions are "`auto`", "`average`", "`count`", "`countnums`", "`max`", "`min`", "`product`", "`stdev`", "`stdevp`", "`sum`", "`var`" and "`varp`".

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-subtotal table:function CDATA #REQUIRED >`

4.14.9 Data Pilot Members

The data pilot members element contains information about the members of the data pilot source. This element can contain data pilot member elements.

XML Code:	`<table:data-pilot-members>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-members ( table:data-pilot-member ) * >`

4.14.10 Data Pilot Member

The data pilot member element contains information about a member of the data pilot table.

XML Code:	`<table:data-pilot-member>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-member EMPTY >`

The attributes associated with the data pilot member element are:

- Member name
- Display
- Show details

Member Name

This attribute specifies the name of the data pilot member.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:name CDATA #REQUIRED>`

Display

This attribute specifies whether or not a data pilot member is visible. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:display %boolean; #IMPLIED >`

Show Details

This attribute specifies whether or not the details about a data pilot member are displayed. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:show-details`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:show-details %boolean; #IMPLIED >`

4.15 Consolidation

Use this function to combine data from several independent table areas. A new area is calculated via a selected mathematical function and based on those areas.

XML Code:	`<table:consolidation>`
Rules:	
DTD:	`<!ELEMENT table:consolidation EMPTY >`

The attributes associated with this element are:

- Function
- Source cell range addresses
- Target cell address
- Use label
- Link to source data

Function

This attribute contains the function which is used to consolidate the data. Possible functions are `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:function CDATA #REQUIRED >`

Source Cell Range Addresses

This attribute contains a list of cell range addresses. These are the source cell ranges.

XML Code:	`table:source-cell-range-addresses`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:source-cell-range-addresses %cell-range-address-list; #REQUIRED >`

Target Cell Address

This attribute contains the target cell address.

XML Code:	`table:target-cell-address`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:target-cell-address %cell-address; #REQUIRED >`

Use Label

This attribute specifies whether or not labels should be used by the consolidation and if used, which ones. Possible values are `none`, `column`, `row` and `both`.

XML Code:	`table:use-label`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:use-label (none | column | row | both) "none" >`

Link to Source Data

This attribute specifies whether to link the data in the consolidation area to the source data, and whether or not to automatically update the results of the consolidation if any changes are made to the original data.

XML Code:	`table:link-to-source-data`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:link-to-source-data %boolean; "false" >`

4.16 DDE Links

The `table:dde-links` container element stores all DDE links. Every link contains the DDE Source and the data of the last connection.

XML Code:	`table:dde-links`
Rules:	
DTD:	`<!ELEMENT table:dde-links (table:dde-link)+ >`

4.16.1 DDE Link

This `table:dde-link` element contains the DDE source and a simple table element.

XML Code:	`table:dde-link`
Rules:	The table does not need a name.
	The elements in the table do not have styles or other information. Only the data in the cell attributes is used.
	The cell does not contain paragraphs.
	The data is stored in the value attributes.
	The table must contain at least one cell.
DTD:	`<!ELEMENT table:dde-link (table:dde-source, table:table) >`

4.16.2 DDE Source

The DDE source of the `<table:dde-link>` element only supports the value `true` for the `office:automatic-update` attribute.

Additionally, the DDE source has a new attribute `table:conversion-mode`.

Conversion Mode

This attribute specifies the method by which the DDE server converts its data into numbers. There are three possible values.

XML Code:	`table:conversion-mode`
Rules:	
DTD:	`<!ATTLIST table:dde-link table:conversion-mode (into-default-style-data-style|into-english-number|let-text) "into-default-style-data-style" >`

4.17 Table Formatting Properties

The following sections detail the formatting properties that can be applied to tables.

4.17.1 Table Width

Every table must have a fixed width. You specify this width as a fixed length.

You can also specify the width of a table relative to the width of the area that the table is in. In this case, you specify the width as a percentage. User agents that support specifying the relative width of a table can specify widths in this way, but it is not essential.

XML Code:	`style:width`
	
	`style:rel-width`

Rules:	

DTD:	`<!ATTLIST style:properties style:width %length; #IMPLIED>`
	`<!ATTLIST style:properties`
	`style:rel-width %percentage; #IMPLIED>`

Notes:	The reasons why every table must have a fixed width and relative widths are only an option are as follows: • Specifying the width of a table by a percentage is useful for current web browsers and other applications where the percentage is relative to the width of a window. But it may cause problems if the percentage relates to a fixed paper width. • Relative widths can also cause problems for applications such as spreadsheet applications, where there is no requirement for a table to fit on a page. However, if an application supports relative widths, it is relatively easy to program the application to calculate a fixed table width, based on a percentage.

4.17.2 Table Alignment

A table alignment property specifies the horizontal alignment of a table.

XML Code:	`table:align`
Rules:	The options for a table alignment property are as follows: • `left` – The table aligns to the left. • `center` – The table aligns to the center. • `right` – The table aligns to the right. • `margins` – The table fills all the space between the left and right margins.

DTD:	`<!ATTLIST style:properties`
	`table:align (left|center|right|margins) #REQUIRED>`

Notes:	User agents that do not support the `margins` value, may treat this value as `left`.

4.17.3 Table Left and Right Margin

These properties specify the distance of the table from the left and right margins. See Section 3.12.19 for a full explanation of left and right margin properties.

XML Code:	`fo:margin-left`
	
	`fo:margin-right`

Rules:	An application may recognize table margins, but this is not essential. Tables that align to the left or to the center ignore right margins, and tables align to the right or to the center ignore left margins.

4.17.4 Table Top and Bottom Margin

These properties specify the distance of the table from the top and bottom margins, `fo:margin-top` and `fo:margin-bottom`. See Section 3.12.22 for a full explanation of top and bottom margin properties.

4.17.5 Page Sequence Entry Point

See Section for information on this attribute `style:page-sequence-name`.

4.17.6 Break Before and Break After

These properties insert a page or column break before or after a table, `fo:break-before` and `fo:break-after`. See Section 3.12.24 for a full explanation of these properties.

4.17.7 Table Background and Background Image

These properties specify the background color and image of a table using the attribute `fo:background-color` and the element `<style:background-image>`. See Section 3.12.25 and 3.12.26 for a full explanation of these properties.

4.17.8 Table Shadow

The table shadow property specifies that a shadow visual effect appears on a table, using the attribute `style:shadow`. See Section 3.12.30 for a full explanation of this property.

4.17.9 Keep with Next

The keep with next property specifies that a table stays with the paragraph that follows it, using the attribute `fo:keep-with-next`. See Section 3.12.31 for a full explanation of this property.

4.17.10 May Break Between Rows

This property specifies that a page break may occur inside a table.

XML Code:	`style:may-break-between-rows`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`style:may-break-between-rows %boolean; #IMPLIED>`

4.17.11 Border Model Property

The `table:border-model` property specifies what border model to use when creating a table with a border.

There are two types of border model, as follows:

- **Collapsing border model**

  When two adjacent cells have different borders, the wider border appears as the border between the cells. Each cell receives half of the width of the border.

- **Separating border model**

  Borders appear within the cell that specifies the border.

Both border models are very similar to the collapsing and separating border models of XSL and **CSS2**. They differ in how border widths relate to row and column widths.

In StarOffice, a row height or column width includes any space required to display borders or padding. This means that, while the width and height of the content area is less than the column width and row height, the sum of the widths of all columns is equal to the total width of the table.

In XSL and CSS2, a column width or row height specifies the width or height of the content area of a cell. This means that the sum of the widths of all columns is less than the width of the table.

XML Code:	`table:border-model`
Rules:	
DTD:	`<!ATTLIST style:properties table:border-model (collapsing|separating) #IMPLIED>`

4.17.12 Page Style

This attribute specifies the name of the page style.

XML Code:	`table:page-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties table:page-style-name %styleName; #IMPLIED>`

4.17.13 Display

This attribute specifies whether or not a table is displayed.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST style:properties table:display %boolean; #IMPLIED>`

4.18 Column Formatting Properties

The following sections detail the formatting properties that you can apply to table columns.

4.18.1 Column Width

Every table column must have a fixed width. You specify this width as a fixed length.

You can also specify the width of a column relative to the width of the area that the column is in. In this case, you specify the width as a percentage. Applications that support specifying the relative width of a column can specify widths in this way, but it is not essential.

New information to be supplied.

XML Code:	`style:column-width` `style:rel-column-width`
Rules:	To specify a fixed width, use the `style:column-width` property. To specify a relative width, use the `style:rel-column-width` property, followed by a number signifying the width you require. Place a percentage sign (%) before this number.

DTD:	`<!ENTITY % rel-number "CDATA">`
	`<!ATTLIST style:properties style:column-width %length; #IMPLIED>`
	`<!ATTLIST style:properties style:rel-column-width %rel-number; #IMPLIED>`

Notes:	The relative width of a table is the sum of all of the relative widths of the columns in the table. A percentage width relates to the table width.
Note:	In XSL and CSS, a column width does not include any border widths or padding.

Break Before and Break After

The break before (`fo:break-before`) and break after (`fo:break-after`) attributes can be used to format tables in a similar way to the way they are used to format paragraphs. For tables, the only values you can set for these attributes are "`auto`" or "`page`". See Section 3.12.24 for more information about using these attributes.

4.19 Table Row Formatting Properties

The following sections detail the formatting properties that you can apply to table rows.

4.19.1 Row Height

This property specifies the height of a table row. By default, row height is the height of the tallest item in the row. You can also specify a minimum height or a fixed height.

XML Code:	`style:row-height` `style:min-row-height`
Rules:	To specify a fixed height, use the `style:row-height` property. To specify a minimum height, use the `style:min-row-height` property.

DTD:	`<!ATTLIST style:properties style:row-height %length; #IMPLIED>`
	`<!ATTLIST style:properties style:min-row-height %length; #IMPLIED>`

Note:	In XSL and CSS, a row height does not include border widths or padding.

Row Background

To apply a background color or a background image to a table row, use the background and background color paragraph formatting properties. See Sections 3.12.25 and 3.12.26 for a full explanation of the background and background color properties.

4.19.2 Break Before and Break After

The break before (`fo:break-before`) and break after (`fo:break-after`) attributes can be used to format table rows in a similar way to the way they are used to format paragraphs. For table rows, the only values you can set for these attributes are "`auto`" or "`page`". See Section 3.12.24 for more information about using these attributes.

4.20 Table Cell Formatting Properties

The following sections detail the formatting properties that you can apply to table cells.

4.20.1 Vertical Alignment

The vertical alignment property allows you to specify the vertical alignment of text in a table cell.

XML Code:	`fo:vertical-align`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`fo:vertical-align (top|middle|bottom|automatic) #IMPLIED>`

Notes:	The options for the vertical alignment property are as follows: • `top` – Aligns text vertically with the top of the cell. • `middle` – Aligns text vertically with the middle of the cell. • `bottom` – Aligns text vertically with the bottom of the cell. • Automatic The application decide how to align the text.
Limitations:	There is no XSL property that specifies the vertical alignment of a table cell. This appears to be an oversight rather than deliberate.

4.20.2 Text Align

See Section 3.12.4 for information on using this property to format table cells.

4.20.3 Text Align Source

This property specifies the source of the text-align property.

XML Code:	`style:text-align-source`
Rules:	If the value of this attribute is "`fix`", only the `fo:text-align` property is used. If the value is "`value-type`", the text alignment is dependent on the `value-type` of the cell.
DTD:	`<!ATTLIST style:properties style:text-align-source (fix|value-type) #IMPLIED>`

4.20.4 Text Outline

See Section 3.11.4 for information on using this property to format table cells.

4.20.5 Direction

This property specifies the direction of characters in a cell. The most common direction is left to right (`ltr`). The other direction is top to bottom (`ttb`), where the characters in the cell are stacked but not rotated.

XML Code:	`fo:direction`
Rules:	
DTD:	`<!ATTLIST style:properties fo:direction ( "ltr" | "ttb" ) #IMPLIED>`

4.20.6 Text Shadow

To specify a text shadow within a table cell, use the `fo:text-shadow` formatting property. See Section 3.11.19 for a full explanation of the text shadow property.

4.20.7 Cell Shadow

To specify a cell shadow in a table cell, use the `style:shadow` formatting property. See Section 3.12.30 for a full explanation of the shadow property.

4.20.8 Cell Background

To apply a background color or a background image to a table cell, use the background and background color paragraph formatting properties. See Sections 3.12.25 and 3.12.26 for a full explanation of the background and background color properties.

4.20.9 Cell Borders and Border Line Width

To apply a border to a table cell and specify the width of the border, use the border and border line width paragraph formatting properties. See Sections 3.12.27 and 3.12.28 for a full explanation of the border and border line width properties.

4.20.10 Padding

To apply padding to a table cell, use the padding paragraph formatting property. See Section 3.12.29 for a full explanation of the padding property.

4.20.11 Left Margin

To specify a left margin in a table cell, use the left margin paragraph formatting property. See Section 3.12.19 for a full explanation of the left margin property.

4.20.12 Wrap Option

This property is like specified in XSL. In addition the XSL property `fo:overflow` is necessary. The default is not wrap like in XSL, but the default is no-wrap.

XML Code:	`fo:wrap-option, fo:overflow`
Rules:	

DTD:	`<!ATTLIST style:properties fo:wrap-option (no-wrap | wrap) #IMPLIED>`
	 `<!ATTLIST style:properties fo:overflow (auto) #FIXED>`

4.20.13 Rotation Angle

This property specifies the value of a rotation angle in degrees.

XML Code:	`style:rotation-angle`
Rules:	
DTD:	`<!ATTLIST style:properties style:rotation-angle %number #IMPLIED>`

4.20.14 Rotation Align

This property specifies how the edge of the text in a cell is aligned after a rotation. There are four alignment options:

Alignment	**Text is...**	**Borders and background are...**
None.	Rotated.	Unchanged.
Bottom of the cell.	Rotated and may overlap with other cells if the text is longer than the length of the cell.	Positioned parallel to the text, whereby the upper or lower edge is drawn at the original position of the cell.
Top of the cell.		
Center of the cell.		

The StarOffice XML code for the rotation align attribute is described in the following table.

XML Code:	`style:rotation-align`
Rules:	The value of this attribute can be "`none`", "`bottom`", "`top`", or "`center`".
DTD:	`<!ATTLIST style:properties style:rotation-align ( "none" | "bottom" | "top" | "center") #IMPLIED>`

4.20.15 Cell Protect

This property specifies how a cell is protected.

XML Code:	`style:cell-protect`
Rules:	This attribute is only evaluated if the current table is protected. The value of the attribute can be "`none`", "`hidden-and-protected`", or a space-separated list containing the values "`protected`" or "`formula-hidden`".
DTD:	`<!ATTLIST style:properties style:cell-protect CDATA #IMPLIED >`

4.20.16 Print Content

This property specifies whether or not the cell content can be printed.

XML Code:	`style:print-content`
Rules:	
DTD:	`<!ATTLIST style:properties style:print-content %boolean; #IMPLIED >`

4.20.17 Data Style

This property contains the name of a data style to use as the data style for the cell. The data style is represented by one of the style elements described in Section . The style can be referenced by a name.

XML Code:	`style:data-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties style:data-style-name %style-name; #REQUIRED>`

CHAPTER **5**

Graphic Content

This chapter provides the StarOffice XML specification for the core elements of the StarOffice graphics applications, StarOffice Draw and StarOffice Impress. It contains the following sections:

- Configuring a Graphics Document
- Master Pages
- Drawing Pages
- Drawing Shapes
- Presentation Shapes
- 3D Shapes
- Graphic Style Elements
- Stroke Properties
- Fill Properties
- Text Animation Properties
- Graphic Properties
- Animation Properties
- Shadow Properties
- Presentation Page Layouts
- Presentation Page Attributes

StarOffice Draw is a subset of StarOffice Impress, so both applications share the same core, and therefore recognize many of the same features. However, there are some differences between the two applications, such as non-supported presentation features in the StarOffice Draw application. In this chapter, these differences are marked as follows:

- For presentations only.

  The StarOffice Draw application ignores these features.

5.1 Configuring a Graphics Document

Information to be supplied.

5.2 Master Pages

You use **master pages** as common backgrounds for **drawing pages.** You assign master pages with the `styles:master-page` element.

XML Code:	`<style:master-page>`
Rules:	You must have one master page element. Each drawing page is linked to one master page. The master page used for the drawing page is set by the `style:parent-style-name` attribute of the drawing pages style. Master pages are stored inside the `<office:styles>` element.
DTD	`<!ELEMENT style:master-page (style:style*, (%shapes;)*, presentation:notes?)>`
Note:	Master pages can store a fixed set of presentation styles.

The attributes associated with the master page element are:

- Page name
- Page-master
- Page style

The elements that you can include in the master page element are:

- Presentation styles
- Presentation notes
- Shapes
- Frames

Page Name

The name of a master page is stored as a `styles:name` attribute. Each master page is referenced through the page name.

XML Code:	`styles:name`
Rules:	A page name is required for each master pageand the name must be unique.
DTD:	`<!ATTLIST style:master-page style:name %styleName; #REQUIRED>`

Page-master

A page-master specifies the size, border, and orientation of a master page. You assign the page-master with the `style:page-master-name` attribute.

XML Code:	`style:page-master-name`
Rules:	A page-master is required for each master page.
DTD:	`<!ATTLIST style:master-page style:page-master-name % styleName; #REQUIRED>`

Page Style

You can assign additional page style attributes to a drawing page. The fixed family for page styles is `drawing-page`.

XML Code:	`draw:style-name`
Rules:	This attribute is optional.
DTD:	`<!ATTLIST draw:page draw:style-name %styleName; #IMPLIED >`

5.2.1 Presentation Styles

You can attach a set of presentation styles to each master page. These are used for special presentation objects. Currently, only the following styles are supported:

- `title`
- `subtitle`
- `background`
- `background-objects`
- `notes`
- `outline1` to `outline9`

You attach your chosen set of presentation styles as a set of `<style:style>` elements.

XML Code:	`<style:style>`
Rules:	The style elements can only have the names of the styles listed above and the style family must be `presentation`.
DTD:	
Notes:	For presentations only.

5.2.2 Presentation Notes

Each drawing page of a presentation can have an additional presentation notes page, which contains a preview of the corresponding drawing page and additional graphic shapes.

All graphic shapes in a `<presentation:notes>` element on a master page are visible as a background of the notes page that corresponds to a drawing page that has this master page as a master.

XML Code:	`<presentation:notes>`
Rules:	
DTD:	`<!ELEMENT presentation:notes (%shapes;)*>`
Notes:	For presentations only.

Example: Master page

```
<office:styles>
  ...
  <style:master-page style:name="home" style:page-master="default">
    <style:style style:name="title" style:family="presentation">
      <style:properties fo:font-style="italic"/>
    </style:style>
    <style:style style:name="subtitle" style:family="presentation"
                 style:parent-style-name="title">
      <style:properties style:text-outline="true"/>
    </style:style>
    <draw:rectangle .../>
    <presentation:notes>
      <draw:text ...>this is a note</draw:text>
    </presentation:notes>
  </style:master-page>
  ...
</office:styles>
```

5.3 Drawing Pages

A drawing page is the main container for content in a drawing or presentation document. Drawing pages are used for the following:

- Drawings
- Slides for presentations

You must assign a master page to each drawing page.

XML Code:	`<draw:page>`
Rules:	
DTD:	`<!ELEMENT draw:page (%shapes;|presentation:notes)>`

The attributes associated with the drawing page element are:

- Page name
- Page style
- Master page
- Presentation page layout
- Presentation page attributes
- Background attributes

The elements that you can include in the drawing page element are:

- Shapes
- Frames
- Presentation notes

Page Name

The name of a drawing page is stored as a `draw:name` attribute. If you set the drawing page name attribute, then the document must have a unique drawing page name. If there is no drawing page name, then the application generates a unique and meaningful name.

XML Code:	`draw:name`
Rules:	This attribute is optional.
DTD:	`<!ATTLIST draw:page draw:name %styleName; #IMPLIED >`

Page Style

You can assign additional formating attributes by assigning a page style for this drawing page. The fixed family for page styles is `drawing-page`.

XML Code:	`draw:style-name`
Rules:	This attribute is optional.
DTD:	`<!ATTLIST draw:page draw:style-name %styleName; #IMPLIED >`

Master Page

Each drawing page must have one master page assigned to it. The master page defines properties like the size and borders for the drawing page, serves as a container for shapes that are used as a common background, and can be used for storing additional styles (in presentations only).

XML Code:	`draw:master-page-name`
Rules:	This attribute is required.
DTD:	`<!ATTLIST draw:page draw:master-page-name %styleName; #REQUIRED>`

Presentation Page Layout

This attribute links to a `<style:presentation-page-layout>` element. See Section 5.14 for information on the presentation page layout element

XML Code:	`presentation:presentation-page-layout-name`
Rules:	This attribute is optional.
DTD:	`<!ATTLIST draw:page presentation:presentation-page-layout-name %styleName; #IMPLIED>`

Presentation Page Properties

Each drawing page can have optional presentation properties, for example, the duration for which a page is displayed or a fade effect. For information on the attributes used to represent these properties, see Section 5.15.

Note: These attributes are for presentations only.

5.3.1 Background Style Properties

A drawing page can have an optional background that defines the background appearance of the page. If you set an optional background, it overrides the background of the assigned master page, but not the shapes on it. You can alter the background of the assigned master page by using one of the following elements in the style element of the page:

- `<style:background-image>` - see Section 3.12.26
- `<fo:background-color>` - see Section 3.12.25
- `<draw:hatch>` - see Section 5.7.2
- `<draw:gradient>` - see Section 5.7.1

5.3.2 Presentation Notes

Each drawing page of a presentation can have an additional presentation notes page, which contains a preview of the corresponding drawing page and additional graphic shapes. You can include the `<presentation:notes>` element in the `<draw:page>` element. See Section 5.2.2 for more information on this element.

Example: Drawing page

```
<office:automatic-styles>
 <style:style style:name="gg3434" style:family="drawing-page">
  <style:properties presentation:page-duration="5s">
 </style:style>
 <style:style style:name="titledia"
              style:family="presentation-page-layout">
  <presentation:placeholder presentation:object="title"
                                    svg:x="20%" svg:y="10%"
                              svg:width="80%" svg:height="10%" />
  <presentation:placeholder presentation:object="subtitle"
                              svg:x="20%" svg:y="30%"
                              svg:width="80%" svg:height="60%" />
 </style:style>
<office:automatic-styles>
<office:body>
 <draw:page office:name="Page 1"
            draw:style-name="gg3434"
            draw:master-page-name="home"
            presentation:page-layout-name="titeldia">
    <draw:rect .../>
    <presentation:notes>
      <draw:text ...>this is a note</draw:text>
    </presentation:notes>
 </draw:page>
<office:body>
```

5.4 Drawing Shapes

5.4.1 Rectangle

The `rect` element represents a rectangular drawing shape.

XML Code: `<draw:rect>`

Rules:

DTD:

```
<!ELEMENT draw:rect text:p*>
<!ATTLIST draw:rect %draw-position
                    %draw-size
                    %draw-style-name
                    %draw-transform>
```

The attributes associated with the rectangle element are:

- Position, Size, Style, and Transformation – see Section 5.4.13
- Round corners

Round Corners

This attribute specifies the radius of the circle used to round off the corners of the rectangle.

XML Code: `draw:corner-radius`

Rules:

DTD:

```
<!ATTLIST draw:rect draw:corner-radius %length; #IMPLIED>
```

Example: Rectangular drawing shape

```
<draw:rect svg:x="2cm" svg:y="3cm" svg:width="10cm" svg:height="20cm" svg:
transform="rotate(45)" draw:style-name="object-with-shadow">
```

5.4.2 Line

The `line` element represents a line in a drawing.

XML Code: `<draw:line>`

Rules:

DTD:

```
<!ELEMENT draw:line text:p*>
<!ATTLIST draw:line     svg:x1      %coordinate; #IMPLIED
                        svg:y1      %coordinate; #IMPLIED
                        svg:x2      %coordinate; #IMPLIED
                        svg:y2      %coordinate; #IMPLIED
                        %draw-style-name
                        %draw-transform>
```

The attributes associated with the line element are:

- Style and Transformation – see Section 5.4.13

- Start point
- End point

Start Point

The start point attributes specify the start coordinates of the line.

XML Code:	`svg:x1` and `svg:y1`
Rules:	
DTD:	See above.

End Point

The end point attributes specify the end coordinates of the line.

XML Code:	`svg:x2` and `svg:y2`
Rules:	
DTD:	See above.

5.4.3 Polyline

The `polyline` element represents a polyline drawing shape.

XML Code: `<draw:polyline>`

Rules:

DTD:

```
<!ELEMENT draw:polyline text:p*>
<!ATTLIST draw:polyline %draw-position
                        %draw-size
                        %draw-viewbox
                        draw:points %Points; #REQUIRED
                        %draw-style-name
                        %draw-transform>
```

The attributes associated with the polyline element are:

- Position, Size, ViewBox, Style, and Transformation – see Section 5.4.13
- Points

Points

Th points attribute stores a sequence of points which are connected by straight lines. Each point consists of two coordinates separated by a comma (,). The points are separated by white spaces.

XML Code:	`svg:points`
Rules:	
DTD:	See above.

Example: Polyline

```
<draw:polyline draw:points="10cm,10cm 12cm,12cm 13cm,13cm"/>
```

5.4.4 Polygon

A polygon is a closed set of straight lines.

XML Code: `<draw:polygon>`

Rules:

DTD:
```
<!ELEMENT draw:polygon text:p*>
<!ATTLIST draw:polygon %draw-position
                       %draw-size
                       %draw-viewbox
                       draw:points %Points; #REQUIRED
                       %draw-style-name
                       %draw-transform>
```

The attributes associated with the polygon element are:

- Position, Size, ViewBox, Style, and Transformation – see Section 5.4.13
- Points – see the previous Section 5.4.3

Path

A path is a shape with a user-defined outline. The shape is built using multiple drawing actions such as:

- *moveto* – set a new current point
- *lineto* – draw a straight line
- *curveto* – draw a curve using a cubic bezier
- *arc* – draw an elliptical or circular arc
- *closepath* – close the current shape by drawing a line to the last *moveto*

Compound paths are paths with subpaths, each subpath consisting of a single *moveto* followed by one or more line or curve operations. Compound paths can be used for effects such as donut holes in objects.

XML Code: `<draw:polygonpath>`

Rules:

DTD:
```
<!ELEMENT draw:polygonpath (text:p|svg:path)*>
<!ATTLIST draw:polygonpath  %draw-position
                       %draw-size
                       %draw-viewbox
                       svg:d %PathData; #REQUIRED
                       %draw-style-name
                       %draw-transform >
```

The attributes associated with the polygon path element are:

- Position, Size, ViewBox, Style, and Transformation – see Section 5.4.13. The `svg:viewbox` attribute is used to scale the points to the rectangle specified by the position and size attributes of the element.

- Path Data

Path Data

The syntax for this attribute is documented in Chapter 8 of the *Scalable Vector Graphics (SVG) 1.0 Specification W3C Working Draft.* See page 20 for a pointer to this document.

XML Code:	`svg:d`
Rules:	
DTD:	`<!ATTLIST svg:path d %PathData; #REQUIRED>`
Implementation limitation:	The current StarOffice drawing core only supports path shapes which have either open or closed curves. A mix of open and closed curves for one shape is not supported. The quadratic bezier and the elliptical arc commands are also not supported yet.

5.4.5 Circle

The circle element represents a circular drawing shape.

XML Code: `<draw:circle>`

Rules:

DTD:

```
<!ELEMENT draw:circle text:p*>
<!ATTLIST draw:circle svg:cx %coordinate; #IMPLIED
                      svg:cy %coordinate; #IMPLIED
                      svg:r %length; #IMPLIED
                      %draw-style-name
                      %draw-transform>
```

The attributes associated with the circle element are:

- Style and Transformation – see Section 5.4.13
- Center point
- Radius

Center Point

The center point attributes specify the coordinates of the center point of the circle.

XML Code: `svg:cx` and `svg:cy`

Rules:

DTD:

```
<!ATTLIST draw:circle svg:cx %coordinate; #IMPLIED
                      svg:cy %coordinate; #IMPLIED>
```

Radius

The radius attribute specifies the radius of the circle.

XML Code:	`svg:r`
Rules:	
DTD:	`<!ATTLIST draw:circle svg:r %length; #IMPLIED>`

5.4.6 Ellipse

The ellipse element represents an ellipse drawing shape.

XML Code: `<draw:ellipse>`

Rules:

DTD:

```
<!ELEMENT draw:ellipse text:p*>
<!ATTLIST draw:ellipse  svg:cx %coordinate; #IMPLIED
                        svg:cy %coordinate; #IMPLIED
                        svg:rx %length; #IMPLIED
                        svg:ry %length; #IMPLIED
                        %draw-style-name
                        %draw-transform>
```

The attributes associated with the ellipse element are:

- Style and Transformation – see Section 5.4.13
- Center point – see Section 5.4.5
- Radius – see Section 5.4.5

5.4.7 Connector

Information to be supplied.

5.4.8 Caption

Information to be supplied.

5.4.9 Measure

Information to be supplied.

5.4.10 Control

Information to be supplied.

5.4.11 Page

Information to be supplied.

5.4.12 Grouping

A group of drawing shapes is represented by a `<draw:g>` element.

XML Code:	`<draw:g>`
Rules:	

DTD:	`<!ELEMENT draw:g (%shapes;)*>`
	`<!ATTLIST draw:g %draw-transform >`

5.4.13 Common Drawing Shape Attributes

The attributes described in this section are common to all drawing shapes.

Position

The position attributes specify the *x* and *y* coordinate start postion of the drawing shape.

XML Code:	`svg:x` and `svg:y`
Rules:	

DTD:	`<!ENTITY % Coordinate "CDATA">`
	`<!ENTITY % draw-position "svg:x %coordinate; #IMPLIED`
	`svg:y %coordinate; #IMPLIED">`

Size

The size attributes specify the width and height of the drawing shape.

XML Code:	`svg:width` and `svg:height`
Rules:	

DTD:	`<!ENTITY % draw-size "svg:width %coordinate; #IMPLIED`
	`svg:height %coordinate; #IMPLIED>`

Transformation

The transform attribute specifies a list of transformations that can be applied to a drawing shape.

XML Code:	`svg:transform`
Rules:	The value of this attribute is a list of transform definitions, which are applied to the drawing shape in the order in which they are listed. The transform definitions in the list must be separated by a white space and/or a comma. The types of transform definitions available include: • `matrix(<a> <b> <c> <d> <e> <f>)`, which specifies a transformation in the form of a transformation matrix of six values. `matrix(a,b,c,d,e,f)` is the equivalent of applying the transformation matrix `[a b c d e f]`. • `translate(<tx> [<ty>])`, which specifies a translation by `tx` and `ty`. • `scale(<sx> [<sy>])`, which specifies a scale operation by `sx` and `sy`. If `<sy>` is not provided, it is assumed to be equal to `<sx>`. • `rotate(<rotate-angle>)`, which specifies a rotation by `<rotate-angle>` about the origin of the shapes coordinate system. • `skewX(<skew-angle>)`, which specifies a skew transformation along the X axis. • `skewY(<skew-angle>)`, which specifies a skew transformation along the Y axis.

DTD:	`<!ENTITY % transform-list; CDATA>`
	`<!ENTITY % draw-transform "svg:transform %TransformList; #IMPLIED">`

ViewBox

This attribute establishes a user coordinate system inside the physical coordinate system of the shape specified by the position and size attributes. This user coordinate system is used by the `svg:points` attribute and the `<svg:path>` element.

XML Code:	`svg:viewbox`
Rules:	The syntax for using this attribute is the same as the SVG syntax. The value of the attribute is four numbers seperated by white spaces, which define the left, top, right and bottom dimensions for the user coordinate system.
DTD:	`<!ENTITY %draw-viewbox "svg:viewbox CDATA #REQUIRED">`

Style

This attribute specifies a style for the drawing shape. The attributes of the specified style and its optional parent styles are used to format the shape.

XML Code:	`draw:style-name`
Rules:	The value of this attribute can be a `<style:style>` element with a style family value of `graphic.`
DTD:	`<!ENTITY % draw-style-name "draw:style-name %styleName; #REQUIRED">`

5.5 Presentation Shapes

Presentation shapes are special shapes contained in a presentation. Presentation shapes use styles with a style family value of `presentation`, unlike drawing shapes which use styles with a style family value of `graphic`. Presentation shapes can be empty, acting only as placeholders.

Standard drawing shapes can also be used in presentations. The `presentation:class` attribute distinguishes presentation shapes from drawing shapes.

5.5.1 Common Presentation Shape Attributes

Style

Presentation shapes can have styles from the style family "`presentation`" assigned to them. You can distinguish a presentation shape from a drawing shape by checking the style attribute used. A drawing shape uses a `draw:style-name` attribute with a style from the "`graphics`" family, while a presentation shape uses a `presentation:style-name` attribute with a style from the "`presentation`" family. This name links to a `<style:style>` element with the family "presentation". The attributes in this style and its optional parent styles are used to format this shape.

XML Code:	`presentation:style-name`
Rules:	
DTD:	`<!ENTITY % presentation-style-name "presentation:style-name % styleName; #IMPLIED">`

Class

This attribute assigns a presentation class to a drawing shape.

XML-Code:	`presentation:class`
Rules:	
DTD:	`<!ENTITY %presentation-class "presentation:class (title|outline|subtitle|text|graphic|object|chart|table|orgchart)`

5.5.2 Title

Titles are standard text shapes . The class name for this presentation shape is `presentation-title`.

5.5.3 Outline

Outlines are standard text shapes. The class name for this presentation shape is `presentation-outline`.

5.5.4 Subtitle

Subtitles are standard text shapes. The class name for this presentation shape is `presentation-subtitle`.

5.5.5 Text

Presentation texts are standard text shapes. The class name for this presentation shape is `presentation-text`.

5.5.6 Graphic

Presentation graphics are standard graphic shapes . The class name for this presentation shape is `presentation-text`.

5.5.7 Object

Presentation objects are standard OLE shapes. The class name for this presentation shape is `presentation-object`.

5.5.8 Chart

Presentation charts are standard OLE shapes. The class name for this presentation shape is `presentation-chart`.

5.5.9 Table

Presentation tables are standard OLE shapes. The class name for this presentation shape is `presentation-table`.

5.5.10 Orgcharts

Presentation organization charts are standard OLE shapes. The class name for this presentation shape is `presentation-orgchart.`

Note: Currently, orgcharts are not implemented in StarOffice.

5.6 3D Shapes

Information to be supplied.

5.7 Graphic Style Elements

The elements described in this section are located in the `<office:styles>` section of a document and are refered to by a unique name. The following styles for filling graphic objects are available:

- Gradient
- Hatch

- Image
- Transparency
- Marker

5.7.1 Gradient

This element defines a gradient for filling a drawing object.

XML Code:	`<draw:gradient>`
Rules:	This element must be located inside the `<office:styles>` elements.
DTD:	`<!ELEMENT draw:gradient EMPTY>`

The attributes associated with the gradient element are:

- Name
- Gradient style
- Gradient center
- Colors
- Intensity
- Angle
- Border

Name

This attribute uniquely identifies a gradient inside an `<office:styles>` element.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:name %styleName; #REQUIRED>`

Gradient Style

This attribute specifies the style of the gradient.

XML Code:	`draw:style`
Rules:	The gradient styles that StarOffice currently supports are `linear`, `axial`, `radial`, `ellipsoid`, `square`, and `rectangular`.
DTD:	`<!ENTITY % gradient-style "(linear|axial|radial|ellipsoid|square|rectangular)">` `<!ATTLIST draw:gradient draw:style %gradient-style; #REQUIRED>`

Gradient Center

If the gradient style is `radial`, `ellipsoid`, `square`, or `rectangular`, the gradient center attribute specifies the center of the geometry that is used for the gradient.

XML Code:	`draw:cx` and `draw:cy`
Rules:	The values of these attributes are always percentage values.

DTD:	`<!ATTLIST draw:gradient draw:cx %coordinate; #IMPLIED`
	`                         draw:cy %coordinate; #IMPLIED>`

Colors

The gradient interpolates between a start color and an end color, which are specified using the following attributes.

XML Code:	`draw:start-color` and `draw:end-color`
Rules:	

DTD:	`<!ATTLIST draw:gradient draw:start-color %color; #REQUIRED>`
	`<!ATTLIST draw:gradient draw:end-color %color; #REQUIRED>`

Intensity

The intensity attributes allow you to use base colors for interpolation and to modify the start and end color intensity using percentage values. In StarOffice, this functionality is only used where a common color is used for the user interface and it can be modified using a different intensity value.

XML Code:	`draw:start-intensity` and `draw:end-intensity`
Rules:	These attributes are optional. If the attributes are not specified, the colors are used as they are, that is at 100% intensity.

DTD:	`<!ATTLIST draw:gradient draw:start-intensity %percentage; #IMPLIED>`
	`<!ATTLIST draw:gradient draw:end-intensity %percentage; #IMPLIED>`

Angle

The angle attribute specifies an angle that rotates the axis at which the gradient values are interpolated.

XML Code:	`draw:angle`
Rules:	This attribute is ignored for radial style gradients.
DTD:	`<!ATTLIST draw:gradient draw:angle %angle; #IMPLIED>`

Border

Depending on the style of the gradient, the border attribute specifies a percentage value which is used to scale a border which is filled by the start or end color only.

XML Code:	`draw:border`
Rules:	
DTD:	`<!ATTLIST draw:gradient draw:border %percentage; #IMPLIED>`

5.7.2 Hatch

This element defines a hatch for filling graphic objects. A hatch is a simple pattern of straight lines that is repeated in the fill area.

XML Code:	`<draw:hatch>`
Rules:	These elements must be located inside the `<office:styles>` elements.
DTD:	`<!ELEMENT draw:hatch EMPTY>`

The attributes associated with the hatch element are:

- Name
- Style
- Color
- Distance
- Angle
- Background

Name

This attribute uniquely identifies a hatch inside an `<office:styles>` element.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:name %styleName; #REQUIRED>`

Style

The style attribute specifies the style of the hatch.

XML Code:	`draw:style`
Rules:	The hatch can have one of three styles: `single`, `double`, or `triple`.

DTD:	`<!ENTITY % hatch-styles "(single|double|triple)">`
	`<!ATTLIST draw:hatch draw:style %hatch-styles; #IMPLIED>`

Color

The color attribute specifies the color of the hatch lines.

XML Code:	`draw:color`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:color %color; #IMPLIED>`

Distance

The distance attribute specifies the distance between two hatch lines.

XML Code:	`draw:distance`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:distance %length; #IMPLIED>`

Angle

The angle attribute specified the rotation angle of the hatch lines.

XML Code:	`draw:rotation`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:rotation %angle; #IMPLIED>`

5.7.3 Image

This element specifies a link to a bitmap resource, for example, a .JPG file. This element follows the Xlink specification.

XML Code: `<draw:fill-image>`

Rules: These elements must be located inside the `<office:styles>` elements.

DTD:

```
<!ELEMENT draw:fill-image EMPTY>
<!ATTLIST draw:fill-image xlink:href %uri; #REQUIRED
                          xlink:type (simple) #IMPLIED
                          xlink:show (parsed) #IMPLIED
                          xlink:actuate (auto) #IMPLIED>
```

The attributes associated with the fill image element are:

- Name
- Size

Name

This attribute uniquely identifies a fill image inside an `<office:styles>` element.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:fill-image draw:name %styleName; #REQUIRED>`

Size

These optional attributes specify the size of the linked image.

XML Code:	`svg:width` `svg:height`
Rules:	These values are optional and are overridden by the physical size of the linked image resource. They can be used to get the size of an image before it is loaded.

DTD:	`<!ATTLIST draw:fill-image svg:width %length #IMPLIED`
	`svg:height %length #IMPLIED>`

5.7.4 Transparency Gradient

To specify a transparency gradient for a graphic object, you can define a transparency that works in a similar fashion to a gradient, except that the transparency is interpolated instead of the color.

XML Code:	`<draw:transparency>`
Rules:	

DTD:	`<!ENTITY % gradient-style "`
	`(linear|axial|radial|ellipsoid|square|rectangular)">`
	`<!ELEMENT draw:transparency EMPTY>`
	`<!ATTLIST draw:transparency draw:style %gradient-style;`
	`#REQUIRED`
	`draw:cx %coordinate; #IMPLIED`
	`draw:cy %coordinate; #IMPLIED`
	`draw:start-transparency %percentage; #IMPLIED`
	`draw:end-transparency %percentage; #IMPLIED`
	`draw:gradient-angle %angle; #IMPLIED`
	`draw:gradient-border %percentage; #IMPLIED>`

The attributes associated with the transparency gradient element are:

- Style
- Transparency center
- Transparency
- Angle
- Border

Style

This attribute is the same as the style attribute associated with the gradient element. See Section 5.7.1 for information.

Transparency Center

This attribute is the same as the gradient center attribute associated with the gradient element. See Section 5.7.1 for information.

Transparency

The transparency interpolates between a start and an end value.

XML Code:	`draw:start-transparency` and `draw:end-transparency`
Rules:	The values of these attributes are percentages where 0% is fully transparent and 100% is fully opaque.
DTD:	*To be supplied*

Angle

This angle rotates the axis at which the transparency values are interpolated. It is the same as the angle attribute associated with the gradient element. See Section 5.7.1 for more information.

Border

Depending on the style of the transparency, the border attribute specifies a percentage value which is used to scale a border which is only the start or end transparency used. This attribute is the same as the border attribute associated with the gradient element. See Section 5.7.1 for more information.

5.7.5 Marker

The marker element represents markers, which are used to draw polygons at the start and end points of strokes.

XML Code:	`<draw:marker>`
Rules:	These elements must be located inside the `<office:styles>` elements.
Implementation limitation:	Currently, markers only store the marker geometry.

DTD:	`<!ELEMENT draw:marker svg:path*>`
	`<!ATTLIST draw:marker draw:name %styleName; #IMPLIED`
	`%draw-viewbox;`
	`svg:d %PathData; #REQUIRED>`

See Sections 5.4.4 and 5.4.13 for information on the Path Data and ViewBox attributes that you can associate with the `<draw:marker>` element.

5.8 Stroke Properties

You use the following **stroke properties** to define drawing object line characteristics in all StarOffice documents:

- Style
- Dash
- Width
- Color
- Start marker

- End marker
- Start marker width
- End marker width
- Start marker center
- End marker center
- Transparency
- Joint

5.8.1 Style

This attribute specifies the style of the stroke on the current object.

XML Code:	`draw:stroke`
Rules:	
DTD:	`<!ATTLIST  style:properties draw:stroke (none|dash|solid) #IMPLIED>`

5.8.2 Dash

This attribute controls the pattern of dashes and gaps used to stroke paths.

XML Code:	`svg:stroke-dasharray`
Rules:	If the value of this attribute is `none` or `inherit`, this attribute is ignored. Otherwise, the value of the attribute can be a list of numbers separated by spaces or commas that specify the length of alternating dashes and gaps.
DTD:	`<!ATTLIST style:properties svg:stroke-dasharray (none|inherit|CDATA) #IMPLIED>`
Implementation limitations:	Currently, all gaps must be the same length. Possible attribute value sequences are: • A sequence of dashes with the same length. • A sequence of dashes with the same length followed by a sequence of dots. • A sequence of dots followed by a sequence of dashes with the same length. Dots are dashes with a length equal to the LineWidth.

5.8.3 Width

This attribute specifies the width of the stroke on the current object in either units of length or as a percentage.

XML Code:	`svg:stroke-width`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-width %length; #IMPLIED>`

5.8.4 Color

This attribute specifies the color of the stroke on the current object.

XML Code:	`svg:stroke-color`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-color %color; #IMPLIED>`

5.8.5 Start Marker

This attribute specifies a line start marker, which is a path that can be connected to the start of a stroke.

XML Code:	`draw:marker-start`
Rules:	If the value of the attribute is a Universal Resource Identifier (URI), it must be the name of a `<draw:marker>` element located inside the `<office:styles>` element.
DTD:	`<!ATTLIST style:properties draw:marker-start (none|%uri); #IMPLIED>`

5.8.6 End Marker

This attribute specifies a stroke end marker, which is a path that can be connected to the end of a stroke.

XML Code:	`draw:marker-end`
Rules:	If the value of the attribute is a URI, it must be the name of a `<draw:marker>` element located inside the `<office:styles>` element.
DTD:	`<!ATTLIST style:properties draw:marker-end (none|%uri); #IMPLIED>`

5.8.7 Start Marker Width

This attribute specifies the width of the marker at the start of the stroke.

XML Code:	`draw:marker-start-width`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-start-width %length; #IMPLIED>`

5.8.8 End Marker Width

This attribute specifies the width of the marker at the end of thestroke.

XML Code:	`draw:marker-end-width`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-end-width %length; #IMPLIED>`

5.8.9 Start Marker Center

This attribute specifies whether or not a start marker is centered at the start of a stroke.

XML Code:	`draw:marker-start-center`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-start-center % boolean; #IMPLIED>`

5.8.10 End Marker Center

This attribute specifies whether or not an end marker is centered at the end of a stroke.

XML Code:	`draw:marker-end-center`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-end-center %boolean; #IMPLIED>`

5.8.11 Opacity

This attribute specifies the opacity of a stroke.

XML Code:	`svg:stroke-opacity`
Rules:	The value of this attribute can be a number between 0 (fully transparent) and 1 (fully opaque) or in a percentage.
DTD:	`<!ATTLIST style:properties svg:stroke-opacity (%opacity-value;|inheritednumber) #IMPLIED>`

5.8.12 Joint

This attribute specifies the shape at the corners of paths or other vector shapes, when they are stroked.

XML Code:	`svg:stroke-linejoin`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-linejoin (miter|round|bevel|middle|none|inherit) #IMPLIED>`

5.9 Fill Properties

The fill properties used in StarOffice Draw and StarOffice Impress are as follows:

- Style
- Color
- Gradient
- Hatch
- Bitmap
- Transparency

5.9.1 Style

This attribute specifies the fill style for a graphic object. Graphic objects that are not closed, such as a path without a closepath at the end, can be filled. The fill operation automatically closes all open subpaths by connecting the last point of the subpath with the first point of the subpath before painting the fill.

XML Code:	`draw:fill`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill (none|solid|bitmap|gradient|hatch|inherited) #IMPLIED>`

5.9.2 Color

This attribute specifies the color of the fill for a graphic object.

XML Code:	`draw:fill-color`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill-color %color; #IMPLIED>`

5.9.3 Gradient

This attribute specifies a gradient style that is used for filling graphic objects.

XML Code:	`draw:fill-gradient-name`
Rules:	

XML Code:	`draw:fill-gradient-name`
DTD:	`<!ATTLIST style:properties draw:fill-gradient-name %styleName; #IMPLIED>`

5.9.4 Gradient Step Count

If a gradient is used for filling, you can set the gradient step count of the color interpolation to be a fixed value.

XML Code:	`draw:gradient-step-count`
Rules:	By default, the step count is automatically calculated based on the size and resolution of the filled area.
DTD:	`<!ATTLIST style:properties draw:gradient-step-count (auto|%value;) #IMPLIED>`

5.9.5 Hatch

This attribute specifies a hatch style that is used for filling.

XML Code:	`draw:fill-hatch-name`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill-hatch-name %styleName; #IMPLIED>`

5.9.6 Bitmap

The following attributes are used when an area is to be filled with a bitmap.

Image

The fill image attribute specifies a URI that links to a `<draw:fill-image>` element.

XML Code:	`draw:fill-image`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill-image %styleName; #IMPLIED>`

Rendering Style

A bitmap image can either be rendered in the given size, stretched to the filled area, or tiled over the area. The style repeat attribute specifies how the bitmap image should be treated.

XML Code:	`style:repeat`
Rules:	The value of this attribute can be `no-repeat`, `repeat`, or `stretch`.
DTD:	`<!ATTLIST style:properties style:repeat (no-repeat|repeat|stretch) #IMPLIED>`

Size

These optional size attributes can be used to override the logical size of the source image data.

XML Code:	`draw:fill-image-width` and `draw:fill-image-height`
Rules:	If the value of the `style:repeat` attribute is `stretch`, these attributes are ignored.
DTD:	`<!ATTLIST style:properties draw:fill-image-width %length; #IMPLIED draw:fill-image-height %length; #IMPLIED>`

Tile Reference Point

These reference point attributes specify the point inside the source image that is used as the top left starting point for tiling.

XML Code:	`draw:refX` and `draw:refY`
Rules:	These attributes are only interpreted if the value of the current `style:repeat` attribute is `repeat`.
DTD:	`<!ATTLIST style:properties draw:refX %percentage; #IMPLIED draw:refY %percentage; #IMPLIED>`

Tile Translation

This attribute defines the translation of each tile in relation to the previous tile.

XML Code:	`draw:tile-repeat-offset`
Rules:	This attribute is only interpreted if the value of the current `style:repeat` attribute is `tiled`. The value of this attribute is a percentage value representing the tiles repeat offset relative to the tiles height or width, followed by either the word `horizontal` or `vertical`.
DTD:	`<!ENTITY % tile-repeat-offset "CDATA">` `<!ATTLIST style:properties draw:tile-repeat-offset %tile-repeat-offset; #IMPLIED>`

Example: Tile translation

```
<style:properties draw:tile-repeat-offset="50% horizontal"/>
```

5.9.7 Transparency

The fill area of a graphic object can either have none, linear, or gradient transparency. None and linear transparency is selected using the `draw:transparency` attribute, while gradient transparency is selected using the `draw:transparency-name` attribute.

None and Linear Transparency

The `draw:transparency` attribute disables transparency or sets a linear transparency for the fill area of a graphic object.

XML Code:	`draw:transparency`
Rules:	
DTD:	`<!ATTLIST style:properties draw:transparency (none|%transparency;) #IMPLIED>`

Gradient Transparency

The `draw:transparency-name` attribute specifies a transparency gradient that defines the transparency for the fill area of a graphic object. When applying a transparency gradient, the transparency is interpolated as defined in the referenced transparency gradient style. This fill style is rendered independently from other fill styles like gradient, image, and hatch.

XML Code:	`draw:transparency-name`
Rules:	The value of this attribute overides the `draw:transparency` attribute.
DTD:	`<!ATTLIST style:properties draw:transparency-name %styleName; #IMPLIED>`

5.10 Text Animation Properties

Information to be supplied. (Kind, Direction, StartInside, StopInside, Count, Delay, Amount)

5.11 Graphic Properties

5.11.1 Color Mode

The color mode style affects the output of colors from a source bitmap or raster graphic.

XML Code:	`draw:color-mode`
Rules:	
DTD:	`<!ENTITY % color-mode ; (greyscale|mono|watermark|standard)>` `<!ATTLIST style:properties draw:color-mode %color-mode; #IMPLIED>`

5.11.2 Adjust Luminance

The luminance attribute specifies a signed percentage value that affects the output luminance of a bitmap or raster graphic.

XML Code:	`draw:luminance`
Rules:	
DTD:	`<!ATTLIST style:properties draw:luminance %signed-percent; #IMPLIED>`

5.11.3 Adjust Contrast

The contrast attribute specifies a signed percentage value that affects the output contrast of a bitmap or raster graphic.

XML Code:	`draw:contrast`
Rules:	
DTD:	`<!ATTLIST style:properties draw:contrast %signed-percent; #IMPLIED>`

5.11.4 Adjust Gamma

The gamma attribute specifies a value that affects the output gamma of a bitmap or raster graphic.

XML Code:	draw:gamma
Rules:	
DTD:	`<!ATTLIST style:properties draw:gamma %signed-percent; #IMPLIED>`

5.11.5 Adjust Red

The red attribute specifies a signed percentage value that affects the output of the red color space of a bitmap or raster graphic.

XML Code:	`draw:red`
Rules:	
DTD:	`<!ATTLIST style:properties draw:red %signed-percent; #IMPLIED>`

5.11.6 Adjust Green

The green attribute specifies a signed percentage value that affects the output of the green color space of a bitmap or raster graphic.

XML Code:	`draw:green`
Rules:	
DTD:	`<!ATTLIST style:properties draw:green %signed-percent; #IMPLIED>`

5.11.7 Adjust Blue

The blue attribute specifies a signed percentage value that affects the output of the blue color space of a bitmap or raster graphic.

XML Code:	`draw:blue`
Rules:	
DTD:	`<!ATTLIST style:properties draw:blue %signed-percent; #IMPLIED>`

5.12 Animation Properties

Information to be supplied.

5.13 Shadow Properties

Each drawing object can have an optional shadow. The following attributes define the rendering of this shadow:

- Shadow
- Offset
- Color
- Transparency

Shadow

The shadow attribute specifies whether the shadow of a shape is visible or hidden.

XML Code:	`draw:shadow`
Rules:	
DTD:	`<!ATTLIST style:properties draw:shadow %visibility>`

Offset

To render a shadow, a copy of the shape is rendered in the single shadow color behind the shape. The offset attributes specify the offset between the top left edge of the shape and the top left edge of the border .

XML Code:	`draw:shadow-distance-x` and `draw:shadow-distance-y`
Rules:	
DTD:	`<!ATTLIST style:properties draw:shadow-offset-x %length; #IMPLIED draw:shadow-offset-y %length; #IMPLIED>`

Color

The shadow color attribute specifies the color in which the shadow is rendered.

XML Code:	`draw:shadow-color`
Rules:	
DTD:	`<!ATTLIST style:properties draw:shadow-color %color; #IMPLIED>`

Transparency

The shadow transparency attribute specifies the transparency in which the shadow is rendered.

XML Code:	`draw:shadow-transparency`
Rules:	The value of this attribute is a percentage value.
DTD:	`<!ATTLIST style:properties draw:shadow-transparency % percent; #IMPLIED>`

5.14 Presentation Page Layouts

A presentation page layout is a container for placeholders, which define a set of empty presentation objects, for example, a title or outline. These placeholders are used as templates for creating new presentation objects and to mark the size and position of an object if the presentation page layout of a drawing page is changed.

XML Code:	`<style:presentation-page-layout>`
Rules:	
DTD:	`<!ELEMENT style:presentation-page-layout presentation: placeholder*>` `<!ATTLIST style:presentation-page-layout style:name %style-name; #REQUIRED>`
Note:	For presentations only.

5.14.1 Presentation Placeholder

The presentation placeholder element specifies a placeholder for presentation objects, for example, a title or outline.

XML Code:	`<presentation:placeholder>`
Rules:	

DTD:	`<!ENTITY % presentation-object "(title|outline|subtitle|text|graphic|object|chart|orgchart|table|page|notes|handout)" >`
	`<!ELEMENT presentation:placeholder EMPTY>`
	`<!ATTLIST presentation:placeholder presentation:object %presentation-object; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:x %Coordinate|%percent; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:y %Coordinate|%percent; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:width %Length|%percent; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:height %Length|%percent; #REQUIRED>`

5.15 Presentation Page Attributes

You can add transition, fade, and audio effects to each presentation page using the following optional presentation attributes:

- Transition Type
- Transition Style
- Transition Speed
- Page Duration
- Page Visibility
- Sound

The transition attributes are contained in the style element of the page.

5.15.1 Transition Type

You can set the mode of transition, for example manual, using the `transition-type` attribute.

XML Code:	`presentation:transition-type`
Rules:	

DTD:	`<!ENTITY % transition-type ; (manual|automatic|semi-automatic)>`
	`<!ATTLIST style:properties presentation:transition-type %transition-type; "manual">`

Note:	For presentations only.

5.15.2 Transition Style

The `transition-style` attribute specifies the way that each presentation page replaces the previous presentation page, for example left-to-right replacement, or fading.

XML Code:	`presentation:transition-style`
Rules:	
DTD:	`<!ENTITY % page-transition "(none|fade-from-left|fade-from-top|fade-from-right|fade-from-bottom|fade-to-center|fade-from-center|move-from-left|move-from-top|move-from-right|move-from-bottom|roll-from-left|roll-from-right|roll-from-bottom|vertical-stripes|horizontal-stripes|clockwise|counterclockwise|fade-from-upperleft|fade-from-upperright|fade-from-lowerleft|fade-from-lowerright|close-vertical|close-horizontal|open-vertical|open-horizontal|spiralin-left|spiralin-right|spiralout-left|spiralout-right|dissolve|wavyline-from-left|wavyline-from-top|wavyline-from-right|wavyline-from-bottom|random|stretch-from-left|stretch-from-top|stretch-from-right|stretch-from-bottom|vertical-lines|horizontal-lines)">` `<!ATTLIST style:properties presentation:transition-style %page-transition; "none">`
Note:	For presentations only.

5.15.3 Transition Speed

The `transition-speed` attribute controls the speed at which a presentation page is removed from display, and replaced by a new presentation page.

XML Code:	`presentation:transition-speed`
Rules:	
DTD:	`<!ENTITY % speed (slow|medium|fast)>` `<!ATTLIST style:properties presentation:transition-speed %speed; "medium">`
Note:	For presentations only.

5.15.4 Page Duration

The `page-duration` attribute controls the amount of time that the presentation page is displayed. T

XML Code:	`presentation:duration`
Rules:	
DTD:	`<!ATTLIST style:properties presentation:page-duration %timeDuration; #IMPLIED>`
Note:	For presentations only.

5.15.5 Page Visibility

You can mark a drawing page as hidden during a presentation by using the `visibility` attribute. A page marked with this attribute is only shown while editing the document but not during the presentation.

XML Code:	`presentation:visibility`
Rules:	

DTD:	`<!ENTITY %visibility (visible|hidden)>`
	`<!ATTLIST style:properties presentation:visibility %visibility; "visible">`

Note:	For presentations only.

5.15.6 Sound

You can add sound effects to your presentation pages using the `sound` element.

XML Code:	`presentation:sound`
Rules:	

DTD:	`<!ELEMENT presentation:sound EMPTY>`
	`<!ATTLIST presentation:sound xlink:href %url; #IMPLIED`
	`xlink:type (simple) #FIXED "simple"`
	`xlink:show (embed) "embed"`
	`xlink:actuate (onLoad) "onLoad">`

Note:	For presentations only.

CHAPTER **6**

Indexing

This chapter describes the StarOffice XML representation of indexes. It contains the following sections:

- Basic Components of StarOffice XML Indexes
- Index Entries
- Index Source Styles
- Index Marks
- Table of Contents
- Index of Illustrations
- Index of Tables
- Index of Objects
- User-Defined Index
- Alphabetical Index
- Bibliography

6.1 Basic Components of StarOffice XML Indexes

StarOffice can automatically generate several types of index, depending on:

- The type of data to be indexed
- The way in which the data for the index is gathered
- The formatting options required for the index

An example of one type of automatically generated index is a table of contents.

An index is represented by an index element and this element contains the following two child elements:

- Index source
- Index body

6.1.1 Index Source

The index source elements describe how the content of an index is generated. The index source element alone is sufficient to recreate the index content, provided that the user did not change the source content since the last index was generated. The index source element contains:

- Several attributes that aid the process of creating the index.
- Index entry template elements that describe the exact format of the individual index entries. For example, the index entries in the Table of Contents in this manual contain the section number, the section name, a tab stop, and the page number. The corresponding index entry template element contains one index entry element for each of these items.

6.1.2 Index Body

The index body element contains the text that makes up the body of the index. It is a standard block of text, with the possible addition of an index header element. If the write protection for the index is disabled, the user can modify the index body but these changes are lost when the index is updated again.

Since you can regenerate the index body at any time, it may seem unnecessary to export it. However, it is better to export the index body for the following reasons:

- It makes it easier to process the document using external tools because if it is not exported, the external tools must regenerate the index.
- Users can modify the index, even though their changes are lost when the index is updated.

The `<text:index-body>` element contains the text elements that make up the index. When the index is regenerated, the current index body content is overwritten.

XML Code:	`<text:index-body>`
Rules:	
DTD:	`<!ELEMENT text:index-body %text;>`

6.1.3 Index Title Template

The `<text:index-title-template>` element determines the style of the index title.

XML Code:	`<text:index-title-template>`
Rules:	The `text:style-name` attribute specifies the paragraph style to use for the index title.
	There can only be one `<text:index-title-template>` element contained in a `<text:table-of-content-source>` element.
DTD:	`<!ELEMENT text:index-title-template CDATA>` `<!ATTLIST text:index-title-template text:style-name % styleName; #IMPLIED>`

6.1.4 Index Entry Templates

The format of an index entry is determined by the index entry template element. There is a template element for each class of entry. For example, for a Table of Contents there is an element for the index header and an element for each outline level. For a Bibliography Index there is an element for the header and an element for each document class.

Each index template element contains a sequence of template elements, where each template element represents one part of an index entry. The most common format for index entries is the chapter number, the chapter title, a tabbed space filled with dots, and a page number. To achieve this the index entry is configured to contain elements for the chapter number, the entry text, the tab space, and the page number.

Different types of indexes support different index entry elements. Therefore, each type of index supports a specific index entry template element with the valid child elements.

XML Code:	`<text:index-entry-template>`
Rules:	
DTD:	`<!ENTITY % templateElement "text:index-entry-chapter|text:index-entry-page-number|text:index-entry-text|text:index-entry-span|text:index-entry-tab-stop">` `<!ELEMENT text:index-entry-template (%entryElement;)*>`
Limitation:	The StarOffice user interface only allows templates to use one page number and one entry text element.

Template Outline Level

This attribute specifies to which outline level this entry configuration applies. There may not be several `<text:outline-level>` elements for the same outline level within the same parent element.

XML Code:	`text:outline-level`
Rules:	This attribute must be unique among all `<text:outline-level>` elements. within the same parent element.
DTD:	`<!ATTLIST text:index-entry-template text:outline-level %number; #REQUIRED>`

Paragraph Style

The paragraph style attribute names the paragraph style to be used for instantiations of this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:index-entry-template text:style-name %styleName; #REQUIRED>`

6.1.5 Common Index and Index Source Attributes

The following attribute is supported by all index elements:

- `text:outline-level`

The following attributes are supported by all index source elements:

- `text:use-index-marks`
- `text:index-scope`

Style Name

An index is formatted using a section style. This allows the index to contain multiple columns. Index header elements use their own section style, which enables a multicolumn index to have a single column title. The `text:style-name` attribute identifies the section style used to format the index.

XML Code:	`text:style-name`
Rules:	
Sample DTD:	`<!ATTLIST text:table-of-content text:style-name %styleName; #IMPLIED>`

Index Scope

All index source elements contain a `text:index-scope` attribute which determines whether the index is generated for the entire document or for the current chapter only.

XML Code:	`text:index-scope`
Rules:	
Sample DTD:	`<!ATTLIST text:table-of-content-source text:index-scope (document|chapter) "document">`
Note:	This attribute is not supported for Bibliography Indexes.

Relative Tab Stops in Index Entries

The `text:relative-tab-stop-position` attribute determines whether the position of tab stops is relative to the left margin or to the left indent as determined by the paragraph style. This is useful if you want to copy the same entry configuration for all outline levels because with relative tab stop positions the tabs do not need to be adjusted to the respective paragraph format.

XML Code:	`text:relative-tab-stop-position`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the tab stop position is relative to the left indent. If the value is `false`, the tab stop position is relative to the left margin.
DTD Example:	`<!ATTLIST text:table-of-content-source text:relative-tab-stop-position %boolean; "true">`
Note:	This attribute is not supported for Bibliography Indexes.

6.2 Index Entries

There are nine types of index entries, as follows:

- Chapter number
- Chapter information
- Entry text
- Page number

- Fixed string
- Bibliography information
- Tab stop
- Hyperlink start and end

6.2.1 Chapter Number

The `<text:index-entry-chapter-number>` element displays the chapter number of the index entry. The character style for the chapter number can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-chapter-number>`
Rules:	

DTD:	`<!ELEMENT text:index-entry-chapter-number EMPTY>`
	`<!ATTLIST text:index-entry-chapter-number text:style-name %styleName; #IMPLIED>`

Note:	This element can only display the chapter number. To display the chapter name, you must use the `<text:index-entry-text>` elements.

6.2.2 Chapter Information

The `<text:index-entry-chapter>` element displays the chapter number of the index entry. The character style for the chapter number can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-chapter>`
Rules:	

DTD:	`<!ELEMENT text:index-entry-chapter-number EMPTY>`
	`<!ATTLIST text:index-entry-chapter-number text:style-name %styleName; #IMPLIED>`

Note:	This element can only display the chapter number. To display the chapter name, you must use the `<text:index-entry-text>` elements.

Display Chapter Format

The `text:display` attribute displays either the chapter number, the chapter name, or both.

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:index-entry-chapter text:display ( name | number | number-and-name ) "number-and-name">`

6.2.3 Entry Text

The `<text:index-entry-text>` element displays the text of the index entry, for example, the chapter name if the entry is derived from a header or the phrase contained in the index mark if the entry is derived from an index mark. The character style for the entry text can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-text>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-text EMPTY>` `<!ATTLIST text:index-entry-text text:style-name %styleName; #IMPLIED>`

6.2.4 Page Number

The `<text:index-entry-page-number>` element displays the page number on which the index entry is located. The character style for the page number can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-page-number>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-page-number EMPTY>` `<!ATTLIST text:index-entry-page-number text:style-name %styleName; #IMPLIED>`

6.2.5 Fixed String

The `<text:index-entry-span>` element represents a fixed string within an index entry. The character style for the entry text can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-span>`
Rules:	Unlike the `<text:span>` element, the `<text:index-entry-span>` element does not have any child elements.
DTD:	`<!ELEMENT text:index-entry-string CDATA>` `<!ATTLIST text:index-entry-string text:style-name %styleName; #IMPLIED>`

6.2.6 Bibliography Information

The `<text:index-entry-bibliography>` element introduces bibliography data into index entry templates.

XML Code:	`<text:index-entry-bibliography>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-bibliography EMPTY>` `<!ATTLIST text:index-entry-bibliography text:style-name %styleName; #IMPLIED>`

Each `<text:index-entry-bibliography>` element can contain:

- A `text:style-name` attribute specifying a character style for the entry
- A bibliography data field identifier

Bibliography Data Field Identifier

The `text:bibliography-data-field` attribute determines which part of the bibliography data field to display.

XML Code:	`text:bibliography-data-field`
Rules:	
DTD:	`<!ATTLIST text:index-entry-bibliography text:bibliography-data-field (address | annote | author | bibiliographic_type | booktitle | chapter | custom1 | custom2 | custom3 | custom4 | custom5 | edition | editor | howpublished | identifier | institution | isbn | journal | month | note | number | organizations | pages | publisher | report_type | school | series | title | url | volume | year) #REQUIRED>`

6.2.7 Tab Stop

The `<text:index-entry-tab-stop>` element represents a tab stop within an index entry. It also contains the position information for the tab stop.

XML Code:	`<text:index-entry-tab-stop>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-tab-stop EMPTY>` `<!ATTLIST text:index-entry-tab-stop text:style-name %styleName; #IMPLIED>`

The attributes that you can associate with the `<text:index-entry-tab-stop>` element are:

- `style:leader-char`
- `style:type`
- `style:position`

Leader Char

The `style:leader-char` attribute specifies the leader character.

XML Code:	`style:leader-char`
Rules:	
DTD:	`<!ATTLIST text:index-entry-tab-stop style:leader-char %char; " ">`

Tab Type

The `style:type` attribute specifies the tab stop type.

XML Code:	`style:type`
Rules:	The `<text:index-entry-tab-stop>` element only supports two types of tab: `left` and `right`. If the value of this attribute is `left`, the `style:position` attribute must also be used. Otherwise, this attribute must be omitted.
DTD:	`<!ATTLIST text:index-entry-tab-stop style:type (left|right) "left">`

Tab Position

The `style:position` attribute specifies the position of the tab.

XML Code:	`style:position`
Rules:	This attribute can only be used with `<text:index-entry-tab-stop>` elements where the value of the `style:type` attribute is `left`.
DTD:	`<!ATTLIST text:index-entry-tab-stop style:position %length; #IMPLIED>`
Note:	Depending on the value of the `text:relative-tab-stop-position` attribute in the `<text:index-entry-config>` element, the position of the tab is interpreted as being relative to the left margin or the left indent.

6.2.8 Hyperlink Start and End

The `<text:index-entry-link-start>` and `<text:index-entry-link-end>` elements mark the start and end of a hyperlink index entry. The character style for the hyperlink can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-link-start>`
	`<text:index-entry-link-end>`

Rules:	

DTD:	`<!ELEMENT text:index-entry-link-start EMPTY>`
	`<!ELEMENT text:index-entry-link-end EMPTY>`
	`<!ATTLIST text:index-entry-link-start text:style-name % styleName; #IMPLIED>`
	`<!ATTLIST text:index-entry-link-end text:style-name % styleName; #IMPLIED>`

6.2.9 Example of an Index Entry Configuration

The following is an example of the XML code for a table of contents called Table of Content with the following characteristics:

- It uses the top two outline levels.
- Each entry consists of the chapter number, a closing parenthesis, the chapter title, a tab stop, and the page

number.

- For the top outline level, the page number is formatted using a style called Bold.
- For the second outline level, a bracket is used instead of a closing parenthesis.

Example: Table of Content

```
<text:table-of-content>
  <text:table-of-content-source
    text:outline-level="2"
    text:use-index-marks="false"
    text:index-scope="document">

    <text:index-title-template text:style-name="Index 1">
      Table of Content
    </text:index-title-template>

    <text:index-entry-template
      text:ouline-level="1"
      text:style-name="Contents 1">
      <text:index-entry-chapter-number/>
      <text:index-entry-span>) </text:index-entry-span>
      <text:index-entry-text/>
      <text:index-entry-tab-stop style:type="right"/>
      <text:index-entry-page-number text:style-name="bold"/>
    </text:index-entry-template>

    <text:index-entry-template
      text:ouline-level="2"
      text:style-name="Contents 2">
      <text:index-entry-chapter-number/>
      <text:index-entry-span>] </text:index-entry-span>
      <text:index-entry-text/>
      <text:index-entry-tab-stop style:type="right"/>
      <text:index-entry-page-number/>
    </text:index-entry-template>

  </text:table-of-content-source>

  <text:table-of-content-body>
    [... header ...]
    <text:p text:style-name="...">1) Chapter
      <text:tab-stop/><text:span style-name="bold"> 1 </text:span>
    </text:p>
    <text:p text:style-name="...">1.1] Subchapter
       <text:tab-stop/>1
    </text:p>
    [... more entries ...]
  </text:table-of-content-body>

</text:table-of-content>
```

6.3 Index Source Styles

The table of content index can gather index entries from paragraphs formatted using certain paragraph styles. The `<text:index-source-styles>` element contains all of the `<text:index-source-style>` elements for a particular outline level.

XML Code:	`<text:index-source-styles>`
Rules:	
DTD:	`<!ELEMENT text:index-source-styles (text:index-source-style)*>`

Source Styles Outline Level

The `text:outline-levels` attribute determines at which outline level to list the index entries gathered from the respective paragraph styles.

XML Code:	`text:outline-levels`
Rules:	
DTD:	`<!ATTLIST text:index-source-styles text:outline-level %number; #IMPLIED>`

6.3.1 Index Source Style

All paragraphs formatted using the style specified in the `<text:index-source-style>` element are included in the index.

XML Code:	`<text:index-source-style>`
Rules:	
DTD:	`<!ELEMENT text:index-source-style EMPTY>`

The attribute associated with the `<text:index-source-style>` element is:

- Style name

Style Name

The `text:style-name` attribute specifies the paragraph style. Paragraphs formatted using this style are included in the index.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:index-source-style text:style-name %styleName; #REQUIRED>`

6.4 Index Marks

There are three types of index marks to correspond to the three types of index that use of index marks. The three types of index marks are:

- Table of content index marks
- User-defined index marks
- Alphabetical index marks

The XML code for index marks is similar to the code for Bookmarks and References. The following are some basic rules about index marks:

- Each index mark is represented by a start and an end element.
- Both elements use an ID attribute to match the appropriate start and end elements.
- The start and end elements for an index mark must be contained in the same paragraph, with the start element occurring first.
- The attributes associated with the index mark are attached to the start element.
- The text between the start and end elements is the text the index entry.
- The formatting attributes for index marks can overlap.

6.4.1 Table of Content Index Marks

The `<text:toc-mark-start>` element marks the start of a table of content index entry.

XML Code:	`<text:toc-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:toc-mark-start EMPTY>`
	`<!ATTLIST text:toc-mark-start text:id ID #REQUIRED>`
	`<!ATTLIST text:toc-mark-start text:outline-level %number; #IMPLIED>`

The attributes associated with the `<text:toc-mark-start>` element are:

- A `text:id` attribute to allow the start and end elements to be matched.
- A `text:outline-level` attribute to specify the outline level of the resulting table of content index entry.

The `<text:toc-mark-end>` element marks the end of a table of contents index entry.

XML Code:	`<text:toc-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. The must be a start element to match the end element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:toc-mark-end EMPTY>`
	`<!ATTLIST text:toc-mark-end text:id ID #REQUIRED>`

Table of content index marks also have a variant that does not enclose the text to be indexed. This is represented using the `<text:toc-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, a `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:toc-mark>`
Rules:	

DTD:	`<!ELEMENT text:toc-mark EMPTY>`
	`<!ATTLIST text:toc-mark text:string-value %string #REQUIRED>`
	`<!ATTLIST text:toc-mark text:outline-level %number; #IMPLIED>`

6.4.2 User-Defined Index Marks

The `<text:user-index-mark-start>` element marks the start of a user-defined index entry.

XML Code:	`<text:user-index-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:user-index-mark-start EMPTY>` `<!ATTLIST text:user-index-mark-start text:id ID #REQUIRED>` `<!ATTLIST text:user-index-mark-start text:outline-level %number; #IMPLIED>`

The `<text:user-index-mark-end>` element marks the end of the user-defined index entry.

XML Code:	`<text:user-index-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be a start element to match the end element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:user-index-mark-end EMPTY>` `<!ATTLIST text:user-index-mark-end text:id ID #REQUIRED>`

User index marks also have a variant that does not enclose the text to be indexed. This is represented by the `<text:user-index-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, the `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:user-index-mark>`
Rules:	
DTD:	`<!ELEMENT text:user-index-mark EMPTY>` `<!ATTLIST text:user-index-mark text:string-value %string; #REQUIRED>` `<!ATTLIST text:user-index-mark text:outline-level %number; #IMPLIED>`

Name of User Index

There can be more than one user-defined index. In this case, the user index must be named using the `text:index-name` attribute. This attribute determines to which user-defined index an index mark belongs. If no name is given, the default user-defined index is used.

XML Code:	`text:index-name`
Rules:	
DTD:	`<!ATTLIST text:user-index-mark text:index-name %string #IMPLIED>` `<!ATTLIST text:user-index-mark-start text:index-name %string #IMPLIED>`

6.4.3 Alphabetical Index Mark

The `<text:alpha-index-mark-start>` element marks the start of an alphabetical index entry. Since alphabetical entries may make use of two keys to structure entries, there are two optional attributes for these keys. There is also a boolean attribute that determines if this entry is intended to be the main entry, if there are several equal entries.

XML Code:	`<text:alpha-index-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:alpha-index-mark-start EMPTY>`
	`<!ATTLIST text:alpha-index-mark-start text:id ID #REQUIRED>`

The attributes associated with the `<text:toc-mark-start>` element are:

- A `text:id` attribute to allow the start and end elements to be matched.
- Additional keys
- Main entry

The `<text:alpha-index-mark-end>` element marks the end of an alphabetical index entry.

XML Code:	`<text:alpha-index-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be a start element to match the end element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:alpha-index-mark-end EMPTY>`
	`<!ATTLIST text:alpha-index-mark-end text:id ID #REQUIRED>`

Alphabetical index marks also have a variant that does not enclose the text to be indexed. This is represented using the `<text:alpha-index-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, a `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:alpha-index-mark>`
Rules:	

DTD:	`<!ELEMENT text:alpha-index-mark EMPTY>`
	`<!ATTLIST text:alpha-index-mark text:string-value %string;`
	`#REQUIRED>`

Additional Keys

The `text:key1` and `text:key2` attributes specify additional keys for the alphabetical index mark.

XML Code:	`text:key1` `text:key2`
Rules:	If only one key is used, it must be contained in the `text:key1` attribute.

DTD:

```
<!ATTLIST text:alpha-index-mark-start
                    text:key1 %string; #IMPLIED
                    text:key2 %string; #IMPLIED >
<!ATTLIST text:alpha-index-mark
                    text:key1 %string; #IMPLIED
                    text:key2 %string; #IMPLIED >
```

Main Entry

If there are several index marks for the same entry, you can declare one entry as the main entry using the `text:main-entry` attribute.

XML Code: `text:main-entry`

Rules:

DTD:

```
<!ATTLIST text:alpha-index-mark-start
                text:main-entry %boolean; "false">
<!ATTLIST text:alpha-index-mark
                text:main-entry %boolean; "false">
```

6.4.4 Bibliography Index Mark

The `<text:bibliography-mark>` element contains the text and informatin for a bibliography index entry. It supports attributes for each type of bibliographical data that a bibliography index may contain.

XML Code: `<text:bibliography-mark>`

Rules:

DTD:

```
<!ELEMENT text:bibliography-mark (#PCDATA)>
<!ATTLIST text:bibliography-mark text:bibiliographic-type
   ( article | book | booklet | conference | custom1 |
     custom2 | custom3 | custom4 | custom5 | email | inbook |
     incollection | inproceedings | journal | manual |
     mastersthesis | misc | phdthesis | proceedings |
     techreport | unpublished | www ) #REQUIRED >
<!ATTLIST text:bibliography-mark text:identifier CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:address CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:annote CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:author CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:booktitle CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:chapter CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:edition CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:editor CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:howpublished CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:institution CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:journal CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:month CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:note CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:number CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:organizations CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:pages CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:publisher CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:school CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:series CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:title CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:report-type CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:volume CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:year CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:url CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom1 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom2 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom3 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom4 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom5 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:isbn CDATA #IMPLIED>
```

6.5 Table of Contents

A table of contents can be created from:

- Table of content index marks in the text
- The outline structure formed by the headers
- Arbitrary paragraph formats

The table of contents is represented by the `<text:table-of-content>` element.

XML Code:	`<text:table-of-content>`
Rules:	
DTD:	`<!ELEMENT text:table-of-content (text:table-of-content-source, text:index-body)>`

The attribute that you can associate with the `<text:table-of-content>` element is:

- `text:style-name`

  This attribute specifies the section style to use for formatting the table of contents.

  DTD:
  ```
  <!ATTLIST text:table-of-content text:style-name %styleName;
  #IMPLIED>
  ```

6.5.1 Table of Content Source

The `<text:table-of-content-source>` element specifies how the table of contents is generated. It specifies how the entries are gathered.

XML Code: `<text:table-of-content-source>`

Rules:

DTD:
```
<!ELEMENT text:table-of-content-source
          (text:index-header-template? |
           text:table-of-content-entry-template* |
           text:index-source-styles* ) >
```

The attributes that you can attach to the `<text:table-of-content-source>` element are:

- Outline level
- Use index marks
- Use index source styles
- Index scope

  See Section 6.1.5 for information about this attribute.

  DTD:
  ```
  <!ATTLIST text:table-of-content-source text:index-scope
  (document|chapter) "document">
  ```

- Relative tab stop position

  See Section 6.1.5 for information about this attribute.

  DTD:
  ```
  <!ATTLIST text:table-of-content-source text:relative-tab-
  stop-position %boolean; "true">
  ```

Outline Level

The `text:outline-level` attribute specifies which outline levels are used when generating the table of contents.

XML Code: `text:outline-level`

Rules:	The value of this attribute must be `none` or an integer greater than zero. If the value of this attribute is `none`, no entries are generated from the outline. If this attribute is omitted, all outline levels are used by default.
DTD:	`<!ATTLIST text:table-of-content-source text:outline-level CDATA #REQUIRED>`

Use Index Marks

The `text:use-index-marks` attribute determines whether or not to use index marks to generate the table of content.

XML Code:	`text:use-index-marks`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the table of contents includes entries generated from table of content index marks. See Section 6.4 for more information on index marks.
DTD:	`<!ATTLIST text:table-of-content-source text:use-index-marks %boolean; "true">`

Use Index Source Styles

The `text:use-index-source-styles` attribute determines whether or not index entries are generated for paragraph formatted using certain paragraph styles.

XML Code:	`text:use-index-source-styles`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the table of contents includes an entry for every paragraph formatted with one of the styles specified in a `<text:index-source-style>` element.
DTD:	`<!ATTLIST text:table-of-content-source text:use-index-source-styles %boolean; "false">`

6.5.2 Table of Content Entry Template

The `<text:table-of-content-entry-template>` element determines the format of an index entry for a particular outline level.

XML Code:	`<text:table-of-content-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-chapter-number` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop` • `text:index-entry-link-start` • `text:index-entry-link-end`

DTD:	`<!ELEMENT text:table-of-content-entry-template`
	`( text:index-entry-chapter |`
	`text:index-entry-page-number |`
	`text:index-entry-text |`
	`text:index-entry-span |`
	`text:index-entry-tab-stop |`
	`text:index-entry-link-start |`
	`text:index-entry-link-end )*>`

Limitation:	The StarOffice user interface only allows templates to use one page number and one entry text element.

The attributes that you can associate with the `<text:table-of-content-entry-template>` element are:

- Template outline level
- Paragraph style

Template Outline Level

This attribute specifies to which outline level the entry configuration applies.

XML Code:	`text:outline-level`
Rules:	You cannot have several `<text:outline-level>` elements for the same outline level within a parent element.

DTD:	`<!ATTLIST text:table-of-content-entry-template`
	`text:outline-level %number; #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	

DTD:	`<!ATTLIST text:table-of-content-entry-template`
	`text:style-name %styleName; #REQUIRED>`

6.6 Index of Illustrations

The index of illustrations lists all images and graphics in the current document or chapter. The index entries can be derived from the caption of the illustration or the name of the illustration.

XML Code: `<text:illustration-index>`

Rules:

DTD:
```
<!ELEMENT text:illustration-index
           (text:illustration-index-source, text:index-body)>
```

The attributes that you can attach to the `<text:illustration-index>` element are:

- `text:style-name`

  This attribute specifies the section style to use for the index of illustrations.

  DTD:
  ```
  <!ATTLIST text:illustration-index
                         text:style-name %styleName; #IMPLIED>
  ```

6.6.1 Index of Illustration Source

The `<text:illustration-index-source>` element specifies how the index of illustrations is generated.

XML Code: `<text:illustration-index-source>`

Rules:

DTD:
```
<!ELEMENT text:illustration-index-source
                  (text:index-header-template? |
                   text:illustration-index-entry-template?) >
```

The attributes you can use with a `<text:illustration-index-source>` element are:

- Use caption
- Caption sequence name
- Caption sequence format
- Index scope

  This attribute specifies whether the index applies to the entire document or only the the current chapter.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:index-scope
  (document|chapter) "document">
  ```

- `text:relative-tab-stop-position`

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:relative-tab-
  stop-position %boolean; "true">
  ```

Use Caption

In StarOffice, each object contained in a text document has a name. In addition, images also have a caption. The image caption or the image name can be gathered for the index of illustrations.

XML Code:	`text:use-caption`
Rules:	This attribute can have a value of `true` or `false`. If the value is `true`, the image caption is used. If the value is `false`, the image name is used.
DTD:	`<!ATTLIST text:illustration-index-source text:use-caption %boolean; "true">`

Caption Sequence Name

Captions are associated with a sequence name. If the `text:use-caption` attribute is set to `true`, you must use this attribute to specify the sequence with which the captions are associated.

XML Code:	`text:caption-sequence-name`
Rules:	If this attribute is omitted, the default sequence for the object type is used, for example the sequence "Illustration" is used for illustrations.
DTD:	`<!ATTLIST text:illustration-index-source text:caption-sequence-name %string; #IMPLIED>`

Caption Sequence Format

If the entries for the index of illustrations are obtained from the image captions, you must use this attribute to specify the format for the entries.

XML Code:	`text:sequence-format`
Rules:	The value of this attribute can be `text`, `category-and-value`, or `caption`.
DTD:	`<!ATTLIST text:illustration-index-source text:sequence-format (text|category-and-value|caption) #IMPLIED>`

6.6.2 Illustration Index Entry Template

The illustration index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:illustration-index-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`
DTD:	`<!ELEMENT text:illustration-index-entry-template ( text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>`
Limitation:	The StarOffice user interface only allows templates to use one page number and one entry text element.

Since you can only have one `<text:illustration-index-entry-template>` element, you do not need to use the `text:outline-level` attribute. The attribute that you can associate with the `<text:illustration-index-entry-template>` element is:

- Paragraph style

Paragraph Style

This attribute identifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:illustration-index-entry-template text:style-name %styleName; #REQUIRED>`

6.7 Index of Tables

The index of tables lists all of the tables in the current document or chapter. It works in exactly the same way as the index of illustrations.

XML Code:	`<text:table-index>`
Rules:	

DTD:	`<!ELEMENT text:table-index (text:table-index-source, text:index-body)>`
	`<!ATTLIST text:table-index text:style-name %styleName; #IMPLIED>`

6.7.1 Table Index Source

The `<text:table-index-source>` element specifies how the index of tables is generated.

XML Code:	`<text:table-index-source>`
Rules:	
DTD:	see below

```
<!ELEMENT text:table-index-source
                         (text:index-header-template?,
                          text:table-index-entry-template?) >
<!ATTLIST text:table-index-source
          text:index-scope (document|chapter) "document"
          text:relative-tab-stop-position %boolean; "true"
          text:use-caption %boolean; "true"
          text:caption-sequence-name %string; #IMPLIED
          text:sequence-format
                   (text|category-and-value|caption) #IMPLIED>
```

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-source>` element. See Section 6.6.1 for detailed information about these attributes.

6.7.2 Table Index Entry Template

The table index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:table-index-entry-template>`
Rules:	
DTD:	see below

```
<!ELEMENT text:table-index-entry-template
         ( text:index-entry-page-number |
           text:index-entry-text |
           text:index-entry-span |
           text:index-entry-tab-stop )*>
<!ATTLIST text:table-index-entry-template
           text:style-name %styleName; #REQUIRED>
```

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See Section 6.6.1 for detailed information about these attributes.

6.8 Index of Objects

The index of objects lists all of the objects in the current document or chapter. It gathers its entries from the known object types.

XML Code:	`<text:object-index>`
Rules:	
DTD:	see below

```
<!ELEMENT text:object-index
               (text:object-index-source, text:index-body)>
<!ATTLIST text:object-index
                      text:style-name %styleName; #IMPLIED>
```

6.8.1 Object Index Source

The `<text:object-index-source>` element determines which object types to include in the index of objects. It also supports the standard index source attributes.

XML Code: `<text:object-index-source>`

Rules:

DTD:

```
<!ELEMENT text:object-index-source
                        (text:index-header-template?,
                         text:object-index-entry-template?) >
```

The attributes that you can associate with the `<text:object-index-source>` element are:

- Use attributes, `text:use-*-objects`
- Index scope

  This attribute specifies whether the index applies to the entire document or only the the current chapter.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:index-scope
  (document|chapter) "document">
  ```

- Relative tab stop position

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:relative-tab-
  stop-position %boolean; "true">
  ```

Use Attributes

The `text:use-*-objects` attributes specify which types of objects to include in the index of objects. There is an attribute for each type of StarOffice object as follows:

- `text:use-spreadsheet-objects`
- `text:use-draw-objects`
- `text:use-chart-objects`
- `text:use-math-objects`

Other objects are included or omitted using the following attribute:

- `text:use-other-objects`

XML Code:

```
text:use-spreadsheet-objects
text:use-math-objects
text:use-draw-objects
text:use-chart-objects
text:use-other-objects
```

Rules:

DTD:

```
<!ATTLIST text:object-index-source
          text:use-spreadsheet-objects %boolean;
                                          "false"
          text:use-draw-objects %boolean; "false"
          text:use-chart-objects %boolean; "false"
          text:use-other-objects %boolean; "false"
          text:use-math-objects %boolean; "false">
```

6.8.2 Object Index Entry Template

The object index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:object-index-entry-template>`
Rules:	
DTD:	see below

```
<!ELEMENT text:object-index-entry-template
         ( text:index-entry-page-number |
           text:index-entry-text |
           text:index-entry-span |
           text:index-entry-tab-stop )*>
<!ATTLIST text:object-index-entry-template
            text:style-name %styleName; #REQUIRED>
```

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See Section 6.6.1 for detailed information about these attributes.

6.9 User-Defined Index

A user-defined index combines the capabilities of the indexes discussed earlier in this chapter. A user-defined index can gather entries from the following sources:

- Index marks
- Paragraphs formatted using particular paragraph styles
- Tables, images, or objects
- Text frames

The `<text:user-index>` element represents a user-defined index.

XML Code:	`<text:user-index>`
Rules:	
DTD:	see below

```
<!ELEMENT text:user-index
                     (text:user-index-source, text:index-body)>
<!ATTLIST text:user-index
                      text:style-name %styleName; #IMPLIED>
```

6.9.1 User-Defined Index Source

The `<text:user-index-source>` element can contain several attributes that determine how the index entries are gathered. It also supports an attribute that determines how the outline levels of the index entries are gathered.

The paragraph formats that are used as index marks are encoded in `<text:index-source-styles>` elements, just like in `<text:table-of-content-source>` elements.

XML Code: `<text:user-index-source>`

Rules:

DTD:

```
<!ELEMENT text:user-index-source
                      ( text:index-header-template?,
                        text:user-index-entry-template*,
                        text:index-source-styles* ) >
```

The attributes you can use with `<text:object-index-source>` elements are:

- Use attributes, `text:use-*`
- Copy outline level
- Index scope

  This attribute specifies whether the index applies to the entire document or only to the current chapter.

  DTD:

  ```
  <!ATTLIST text:illustration-index-source text:index-scope
  (document|chapter) "document">
  ```

- Relative tab stop position

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

  DTD:

  ```
  <!ATTLIST text:illustration-index-source text:relative-tab-
  stop-position %boolean; "true">
  ```

Use Attributes

The `text:use-*` attributes specify which entries to include in the user-defined index. The attributes that you can specify are:

- `text:use-index-marks`
- `text:use-graphics`
- `text:use-tables`
- `text:use-floating-frames`
- `text:use-objects`

XML Code:

```
text:use-index-marks
text:use-graphics
text:use-tables
text:use-floating-frames
text:use-objects
```

Rules:

DTD:

```
<!ATTLIST text:user-index-source
          text:use-index-marks %boolean; "false"
          text:use-graphics %boolean; "false"
          text:use-tables %boolean; "false"
          text:use-floating-frames %boolean;
                                         "false"
          text:use-objects %boolean; "false">
```

Copy Outline Level

XML Code:	`text:copy-outline-level`
Rules:	This attribute can have a value of `true` or `false`. If the value is `true`, the entries are gathered at the outline level of the source element to which they refer. If the value is `false`, all index entries gathered are at the top outline level. For example, if an image appears in Section 1.2.3, the entry for the image is located at outline level 3.
DTD:	`<!ATTLIST text:user-index-source text:copy-outline-level %boolean; "false">`

6.9.2 User-Defined Index Entry Template

User index entry templates support entry elements for chapter number, page number, entry text, text spans, and tab stops.

XML Code:	`<text:user-index-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-chapter-number` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`
DTD:	`<!ELEMENT user-index-entry-template ( text:index-entry-chapter | text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>`

The attributes that you can associate with the `<text:user-index-entry-template>` elements are:

- Template outline level
- Paragraph style

Template Outline Level

The `text:outline-level` attribute specifies to which outline level this entry configuration applies.

XML Code:	`text:outline-level`
Rules:	You cannot have several `<text:outline-level>` elements for the same outline level within a parent element.
DTD:	`<!ATTLIST text:table-of-content-entry-template text:outline-level %number; #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

6.10 Alphabetical Index

An alphabetical index gathers its entries solely from index marks.

XML Code:	`<text:alphabetical-index>`
Rules:	
DTD:	`<!ELEMENT text:alphabetical-index (text:alphabatical-index-source, text:index-body)>` `<!ATTLIST text:alphabetical-index text:style-name %styleName; #IMPLIED>`

6.10.1 Alphabetical Index Source

The `<text:alphabetical-index-source>` element specifies how the alphabetical index is generated.

XML Code:	`<text:alphabetical-index-source>`
Rules:	
DTD:	`<!ELEMENT text:alphabetical-index-source ( text:index-header-template?, text:alphabetical-index-entry-template* ) >`

The attributes you can associate with `<text:alphabetical-index-source>` elements are:

- Ignore case
- Main entry style name
- Alphabetical separators
- Combine entries attributes
- Use keys as entries
- Capitalize entries
- Comma separated entries
- Index scope

  This attribute specifies whether the index applies to the entire document or only to the current chapter.

DTD:	`<!ATTLIST text:alphabetical-index-source text:index-scope (document|chapter) "document">`

- Relative tab stop position

This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

DTD:	`<!ATTLIST text:alphabetical-index-source text:relative-tab-stop-position %boolean; "true">`

Ignore Case

The `text:ignore-case` attribute determines whether or not the capitalization of words is ignored.

XML Code:	`text:ignore-case`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the capitalization is ignored and entries that are identical except for character case are listed as the same entries. If the value is `false`, the capitalization of words is not ignored.
DTD:	`<!ATTLIST text:alphabetical-index-source text:ignore-case %boolean; "false">`

Main Entry Style Name

The `text:main-entry-style-name` attribute determines the character style to use for main entries. Sub-entries are formatted using the default character style determined by the paragraph style of the entries.

XML Code:	`text:main-entry-style-name`
Rules:	
DTD:	`<!ATTLIST text:alphabetical-index-source text:main-entry-style-name %styleName; #IMPLIED>`

Alphabetical Separators

The `text:alphabetical-separators` attribute determines whether or not entries beginning with the same letter are grouped and separated from the entries beginning with the next letter, and so on.

XML Code:	`text:alphabetical-separators`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, all entries beginning with the same letter are grouped together. The index contains headings for each section, for example, A for all entries starting with the letter A, B for all entries starting with the letter B, and so on.
DTD:	`<!ATTLIST text:alphabetical-index-source text:alphabetical-separators %boolean; "false">`

Combining Entries

The StarOffice software provides several options for dealing with the common situation where you have multiple index entries for the same word or phrase, as follows:

- Multiple entries for the same word can be combined into a single entry using the `text:combine-entries` attribute.
- If the combined entry contains a sequence of pages, the pages can be formatted:
  - As a range of numbers separated by a dash using the `text:combine-entries-with-dash` attribute
  - As the start number with a pp label, or the appropriate label for the chosen language, using the `text:combine-entries-with-pp` attribute

XML Code:	`text:combine-entries`
	`text:combine-entries-with-dash`
	`text:combine-entries-with-pp`

Rules:	

DTD:	`<!ATTLIST text:alphabetical-index-source`
	`  text:combine-entries %boolean; "true"`
	`  text:combine-entries-with-dash %boolean; "false"`
	`  text:combine-entries-with-pp %boolean; "true">`

Example: Combining index entries

An index mark for the word *XML* occurs on pages 45, 46, 47, and 48. The entries can be formatted as follows:

Entry formatted as	Result
Separate entries	XML.....................................45 XML.....................................46 etc.
Simple combined entries	XML.....................................45, 46, 47, 48
Entries combined with dash	XML.....................................45-48
Entries combined with pp	XML.....................................45pp

Use Keys as Entries

In addition to a keyword, index marks can have up to two keys.

XML Code:	`text:use-keys-as-entries`
Rules:	If the value of this atttribute is `true`, the keys are used as additional entries. If the value of this attribute is `false`, the keys are used as sub-entries.
DTD:	`<!ATTLIST text:alphabetical-index-source text:use-keys-as-entries %boolean; "false">`

Capitalize Entries

The `text:capitalize-entries` attribute determines whether or not the StarOffice software capitalizes all entries in the index.

XML Code:	`text:capitalize-entries`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST text:alphabetical-index-source text:capitalize-entries %boolean; "false">`

Comma Separated Entries

The `text:comma-separated` attribute specifies how to treat multiple index entries. Instead of listing each index entry on a separate line, the StarOffice software can list multiple entries on a single line separated by a comma.

XML Code:	`text:comma-separated`
Rules:	If the value of this attribute is `true`, multiple entries are listed on a single line separated by a comma. By default, the value of this attribute is `false` and each index entry is displayed on a separate line.

DTD:	`<!ATTLIST text:alphabetical-index-source text:`
	`comma-separated %boolean; "false">`

6.10.2 Alphabetical Index Entry Template

Alphabetical indexes support three levels; one level for the main index entry, and up to two additional levels for keys associated with the index entries. Alphabetical indexes also use an entry template for the alphabetical separator.

XML Code:	`<text:alphabetical-index-entry-template>`
Rules:	Alphabetical indexes support the following child elements: • `text:index-entry-chapter` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`

DTD:	`<!ELEMENT alphabetical-index-entry-template`
	`( text:index-entry-chapter |`
	`text:index-entry-page-number |`
	`text:index-entry-text |`
	`text:index-entry-span |`
	`text:index-entry-tab-stop )*>`

The attributes that you can associate with the `<text:alphabetical-index-entry-template>` elements are:

- Template outline level
- Paragraph style

Template Outline Level

This attribute specifies whether the template applies to:

- One of the three levels 1,2,or 3

or

- The alphabetical separator

XML Code:	`text:outline-level`
Rules:	The value of this attribute can be `1`, `2`, `3`, or `separator`. You cannot have several `<text:outline-level>` elements for the same outline level within the same parent element.

DTD:	`<!ATTLIST text:table-of-content-entry-template`
	`   text:outline-level (1|2|3|separator) #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

XML Code:	`text:style-name`
Rules:	

DTD:	`<!ATTLIST text:table-of-content-entry-template`
	`           text:style-name %styleName; #REQUIRED>`

6.11 Bibliography

A bibliography index gathers its entries from bibliography index marks. The `<text:bibliography>` element represents a bibliography.

XML Code:	`<text:bibliography>`
Rules:	

DTD:	`<!ELEMENT text:bibliography (text:bibliography-source, text:index-body)>`
	`<!ATTLIST text:bibliography text:style-name %styleName; #IMPLIED>`

6.11.1 Bibliography Index Source

The `<text:bibliography-source>` element specifies how the bibliography is generated.

XML Code:	`<text:bibliography-source>`
Rules:	This element does not have an associated attributes.

DTD:	`<!ELEMENT text:alphabetical-index-source`
	`                   ( text:index-header-template?,`
	`                     text:bibliography-entry-template*) >`

Implementation limitation:	Bibliography do not currently support the common index source attributes described in Section 6.1.5.

6.11.2 Bibliography Entry Template

Bibliography entry templates support entry elements for bibliography data, text spans, and tab stops. There is one entry template element for each type of entry.

XML Code:	`<text:bibliography-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-span` • `text:index-entry-tab-stop` • `text:index-entry-bibliography-data-field`
DTD:	`<!ELEMENT text:bibliography-entry-template ( text:index-entry-span | text:index-entry-tab-stop | text:index-entry-bibliography-data-field )*>`

The attributes that you can associate with the `<text:bibligraphy-entry-template>` elements are:

- Bibliography type
- Paragraph style

Bibliography Type

This attribute specifies to which type of bibliographical entry the template applies.

XML Code:	`text:bibliography-type`
Rules:	This attribute must be unique among all `<text:bibliography-type>` elements within the same parent element.
DTD:	`<!ATTLIST text:bibliography-entry-template text:outline-level (article | book | booklet | conference | custom1 | custom2 | custom3 | custom4 | custom5 | email | inbook | incollection | inproceedings | journal | manual | mastersthesis | misc | phdthesis | proceedings | techreport | unpublished | www) #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

CHAPTER **7**

Chart Content

This chapter describes StarOffice XML chart content. It contains the following sections:

- Introduction
- Chart
- Title
- Subtitle
- Legend
- Plot Area
- Wall
- Floor
- Axis
- Series
- Categories
- Data Point
- Common Chart Properties

7.1 Introduction

In StarOffice XML, chart documents can exist as:

- Standalone documents
  The chart data is contained in a `<table:table>` element inside the chart document. To create a standalone chart document, set the value of the `office:class` attribute to `chart` in the `<office:document>` element.

- Documents contained inside other XML documents
  The chart data may be contained in a `<table:table>` element in the surround document, for example, a spreadsheet or text document.

To reference the correct table and cells you can use the `table:cell-range-address` attributes, which are applied to the <`chart:series>` elements that represent the data series in the chart.

7.2 Chart

The chart element represents an entire chart, including titles, a legend , and the graphical object that visualizes the underlying data called the plot area.

XML Code:	`<chart:chart>`
Rules:	
DTD:	`<!ELEMENT chart:chart ( chart:plot-area, chart:title?, chart:subtitle?, chart:legend? )>`

Class

The class attribute specifies the chart type. If you need to specify the type more precisely there are additional properties that you can assign using styles. For example, if you want to specify a 3D bar chart with horizontal bars, you must set the class attribute to `bar` and you must set the properties for three dimensional and horizontal arrangement in the corresponding style attribute.

XML Code:	`chart:class`
Rules:	The value of this attribute can be one of the main categories of chart types: `line`, `area`, `circle`, `ring`, `scatter`, `radar`, `bar`, or `stock`. The value `bubble` is not yet supported by the `<chart:chart>` element.
Implementation limitation:	Currently, this attribute is only supported by the `<chart:chart>` element.
DTD:	`<!ENTITY % chart-class "(line|area|circle|ring|scatter|radar|bar|stock|bubble)">` `<!ATTLIST chart:chart chart:class %chart-class; #REQUIRED svg:width %length; #IMPLIED svg:height %length; #IMPLIED chart:style-name %style-name; #IMPLIED >`

The `svg:width` and `svg:height` attributes define the extent of the entire chart. Normally, the size of the chart is determined by the size of the window in which the chart is displayed. You can set these attributes as a reference size, so that positions and sizes in sub-elements can be adapted.

General Style Properties

The scale text property allows you to specify that all text objects in the chart should be scaled whenever the size of the chart changes .

XML Code:	`chart:scale-text`
Rules:	To enable scaling, set the value of this property to `true`.
DTD:	`<!ATTLIST style:properties chart:scale-text %boolean; "true" >`

To set the background properties for a <`chart:chart`> element, you can use the Fill Properties (described in Section 7.13.1) and the Stroke Properties (described in Section 7.13.2).

7.3 Title

The title element represents a main title object in a chart document.

XML Code:	`<chart:title>`
Rules:	This element can contain fixed text or it can contain a `<table:cell-address>` element pointing to the text that should be displayed as the title. This element can also be a sub-element of `chart:axis,`see Section 7.9. In this case the title is displayed beside the axis object.
Implementation limitation:	Currently, only inplace titles are supported.
DTD:	see below

```
<!ELEMENT chart:title text:p?>
<!ATTLIST chart:title
          table:cell-range %cell-address; #IMPLIED
          svg:x %Coordinate; #IMPLIED
          svg:y %Coordinate; #IMPLIED
          chart:style-name %style-name; #IMPLIED >
```

Properties

You can apply fill and stroke properties to the surrounding title box. See Sections 7.13.1 and 7.13.2 for more information. You can also apply text properties to the title text itself, see Section 7.13.3. You can also apply two alignment properties, Orientation and RotationAngle, see Section 7.13.4.

7.4 Subtitle

The subtitle element represents a subtitle which can be used for additional title information in a chart. The structure of the subtitle element is similar to that of the title element.

XML Code:	`<chart:subtitle>`
Rules:	
Implementation limitation:	Currently, only inplace titles are supported.
DTD:	see below

```
<!ELEMENT chart:subtitle text:p?>
<!ATTLIST chart:subtitle
          table:cell-range %cell-address; #IMPLIED
          svg:x %Coordinate; #IMPLIED
          svg:y %Coordinate; #IMPLIED
          chart:style-name %style-name; #IMPLIED >
```

Properties

You can apply the same properties to the `<chart:subtitle>` element as you can apply to the `<chart:title>` element. See Section 7.3 for more information.

7.5 Legend

The legend element determines whether or not a legend is displayed in the chart. You can set either a relative or an absolute position for the legend. The size of the legend is calculated automatically and therefore cannot be set as attribute.

XML Code:	`<chart:legend>`
Rules:	
DTD:	see below

```
<!ELEMENT chart:legend EMPTY>
<!ATTLIST chart:legend
       chart:legend-position (top|left|bottom|right) "right"
       svg:x %Coordinate; #IMPLIED
       svg:y %Coordinate; #IMPLIED
       chart:style-name %style-name; #IMPLIED >
```

Properties

You can apply fill and stroke properties to the legend object, see Sections 7.13.1and 7.13.2. You can also set text properties for the text inside the legend object, see Section 7.13.3.

7.6 Plot Area

The plot area element is a container for the graphics objects that represent chart data. The main purpose of the plot area is to be a container for the series elements that represent single data series, and the axis elements.

XML Code:	`<chart:plot-area>`
Rules:	
DTD:	see below

```
<!ELEMENT chart:plot-area
         (chart:series*,
          chart:axis*,
          chart:wall?,
          chart:floor?) >
<!ATTLIST chart:plot-area
          svg:x %Coordinate; #IMPLIED
          svg:y %Coordinate; #IMPLIED
          svg:width %length; #IMPLIED
          svg:height %length; #IMPLIED
          chart:style-name %style-name; #IMPLIED >
```

The style that you apply to the plot area element is used for all data elements contained inside the plot area, unless you specify extra styles in one of those sub elements. These data elements can be `<chart:series>` and `<chart:data-point>` elements.

If the position and size attributes are not specified, the values are calculated by the render application.

The only purpose of the style attribute is to store scene properties for three-dimensional charts.

Properties

If the chart is three-dimensional, you can apply scene properties to the plot area. See Section 5.6 for more information.

7.7 Wall

The wall element can be contained in the plot area element. For two-dimensional charts, the wall element spans the entire plot area. For three-dimensional charts, the wall element usually consists of two perpendicular rectangles. You can use the width attribute to set the width of a wall for three-dimensional charts.

XML Code:	`<chart:wall>`
Rules:	
Implementation limitation:	StarOffice Chart does not yet support the width of the wall. Walls are always drawn as zero-width rectangles.
DTD:	see below

```
<!ELEMENT chart:wall EMPTY>
<!ATTLIST chart:wall
          svg:width %length; #IMPLIED
          chart:style-name %style-name; #IMPLIED >
```

Properties

You can apply fill and stroke properties to a wall. See Sections 7.13.1 and 7.13.2 for more information.

7.8 Floor

The floor element can be contained in the plot area element. For three-dimensional charts, the floor element is present in addition to the wall element. The size of the floor is determined in respect of the size of the plot area, which is always a two-dimensional rectangle that serves as a bounding rectangle of the three-dimensional scene. You can use the width attribute to set the width of the floor.

XML Code:	`<chart:floor>`
Rules:	
Implementation limitation:	StarOffice Chart does not yet support the floor thickness. The floor thickness is always a fixed width.
DTD:	see below

```
<!ELEMENT chart:floor EMPTY>
<!ATTLIST chart:floor
          svg:width %length; #IMPLIED
          chart:style-name %style-name; #IMPLIED >
```

Properties

You can apply fill and stroke properties to a floor. See Sections 7.13.1 and 7.13.2 for more information.

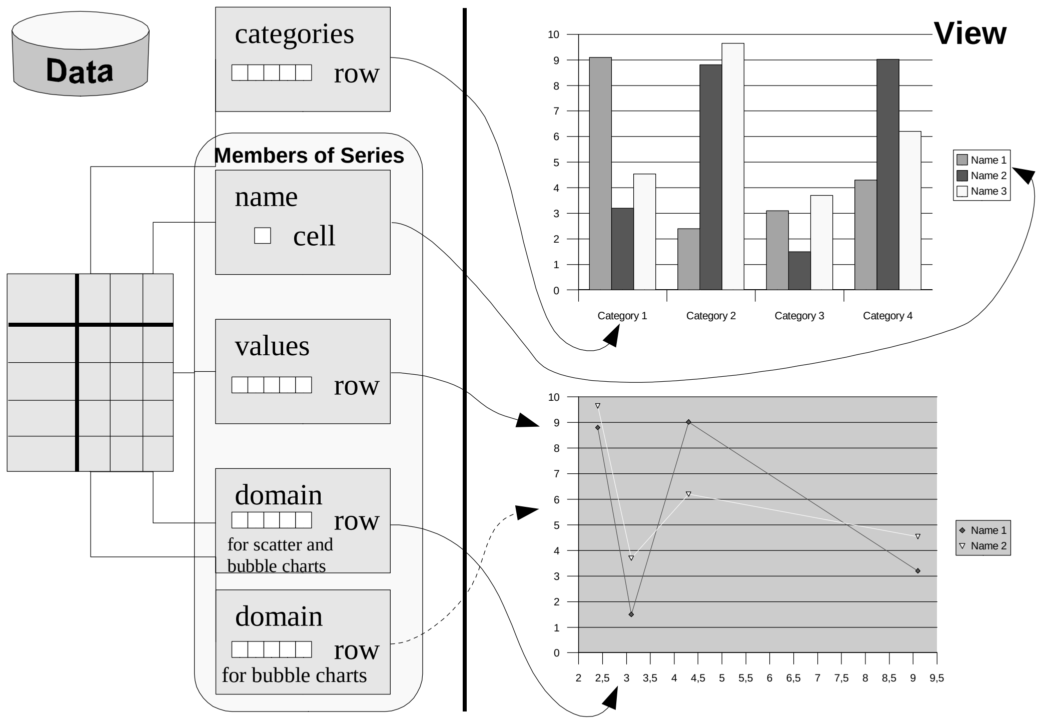
7.9 Axis

The axis element mainly contains style information, in particular scaling information. Chart data is usually structured as follows:

- Several data series each consisting of a name, for example, the name of a company.
- Values, for example, the yield of the company in different years.

- One value in each series belongs to a category, for example, the year.

Figure: Chart data and its representation



XML Code:	`<chart:axis>`
Rules:	Use the `chart:axis-class` attribute to specify which type of data the axis is associated with.
DTD:	see below

```
<!ELEMENT chart:axis (chart:title?,chart:grid?)>
<!ATTLIST chart:axis
  chart:axis-class (category|value|series|domain) #REQUIRED
  chart:axis-name %string #IMPLIED;
  chart:style-name %style-name; #IMPLIED >
```

Current implementation limitations:

- Titles are only supported for a maximum of one axis per class.
- StarOffice Chart only supports the following axes, the numbers in parenthesis indicating how far the value can extended:

Direction	2D		3D	
	Number	**Attachable**	**Number**	**Attachable**
`category`	2	0	1(2-4)	0
`value`	2	2	1(3-4)	1(3-4)
`series`	-	-	1(2-4)	0
`domain` (for scatter/bubble charts)	2	0	-	-

Defining Axes

Here are some guidelines for defining axes in a chart document:

1. The first axis you might want to apply to a chart is an axis representing categories. To do this, you insert an axis element with the `axis:class` attribute set to `category`.

2. Next, you insert an axis showing a scale for your values. To do this, you insert an axis element with the `axis:class` attribute set to `value`.

3. In three-dimensional charts the names of the series, that are usually displayed in the legend, can also be displayed on an axis. To do this, insert an axis element with the `axis:class` attribute set to `series`.

4. If you have a scatter or bubble chart, each series has a domain of values specifying the x-coordinate, and y-coordinate for bubble charts, apart from the values that are to be visualized. For these types of charts, you can insert an axis element with the `axis:class` attribute set to `domain`, which will result in an axis similar to the axis described in Step 2.

5. A chart can contain more than one axis of the same type. For example, if you have two value axes, data series can be attached to either axis. This way data can be grouped for different scaling. To attach a specific axis to a series element you must refer to the axis by the `chart:axis-name` attribute. The axis name is required whenever you intend to attach data series to an axis. Otherwise the axis becomes a copy of an existing axis of the same class.

The position of an axis in a chart is determined by the render application and depends on the chart type. If you have a chart with horizontal bars, the render application usually paints the value axis on the bottom of the plot area. If you have two value axes, a render application might paint the second axis at the top of the plot area.

Note: If your data consists of numbers only and you want to create a scatter chart, the axis representing the values from the x-axis must have the `axis:class` attribute set to `domain` although your domain consists of values.

Example: Bar chart

In this example, there are two value axes and one axis has the name `primary-value`. You can attach a data series to that named axis by using the name. There is no data attached to the second axis, therefore you do not need to specify a name and the axis is just a copy of the first one.

```
<chart:chart chart:class="bar">
  <chart:title>
   <text:p>Title of my chart</text:p>
  </chart:title>
  <chart:plot-area>
    ...
    <chart:axis chart:axis-class="category" chart:axis-name="x"/>
    <chart:axis chart:axis-class="value" chart:axis-name="primary-value"/>
    <chart:axis chart:axis-class="value"/>  <!-- copy of previous axis -->
    ...
    <chart:series chart:values-address="Sheet1.A1:.A7"
     chart:attached-axis-name="primary-value"/>
    ...
  </chart:plot-area>
</chart:chart>
```

General Properties

You can apply stroke properties to axes, see Section 7.13.2. These properties affect all lines of the axis object. You also can apply text properties to axes, see Section 7.13.3. These properties affect the appearance of all text objects.

Number Format Properties

You can apply number format properties to axes, which affect the numbers displayed beside the axis. See Section for information on number format properties. If you omit these properties, the standard number format is used. If the chart is embedded in a spreadsheet and you omit the number format properties, the number format is taken from the number format settings of the spreadsheet cells that contain the chart data.

DTD:	`<!ATTLIST style:properties style:data-style-name %style-name; #IMPLIED >`

Visibility Property

To determine whether or not an axis object is visible, use the `chart:axis-visible` style property. This way, you can provide a chart with scaling information without displaying the axis object.

XML Code:	`chart:axis-visible`
Rules:	
DTD:	`<!ATTLIST style:properties chart:axis-visible %boolean; "true" >`

Scaling Properties

If a scaling attribute is omitted, the axis is set to adaptation mode. This means that the value is not set to a fixed value but may be changed by the render application if data changes. However, the `chart:axis-logarithmic` attribute is set to `false`.

XML Code:

Rules:

DTD:

```
<!ATTLIST style:properties
          chart:axis-minimum %float; #IMPLIED
          chart:axis-maximum %float; #IMPLIED
          chart:axis-interval-major %float; #IMPLIED
          chart:axis-interval-minor %float; #IMPLIED
          chart:axis-origin %float; #IMPLIED
          chart:axis-logarithmic %boolean; "false">
```

Tickmark Properties

The tickmark properties allow you to specify the existence of tickmarks at an axis. The major marks are drawn with respect to the major interval that may be specified by the `chart:axis-interval-major` attribute. The minor tick marks refer to the `chart:axis-interval-minor` attribute. Inner marks are drawn towards the inside of the plot area, that is to the right for an axis displayed on the left hand side of the plot area, and to the left for an axis displayed on the right hand side of the plot area. Outer marks point in the opposite direction. If both properties are specified, one tick mark is drawn that crosses the axis.

XML Code:	`chart:axis-ticks-major-inner`, `chart:axis-ticks-major-outer`, `chart:axis-ticks-minor-inner`, and `chart:axis-ticks-minor-outer`
Rules:	
DTD:	see below

```
<!ATTLIST style:properties
          chart:axis-ticks-major-inner %boolean; "false"
          chart:axis-ticks-major-outer %boolean; "true"
          chart:axis-ticks-minor-inner %boolean; "false"
          chart:axis-ticks-minor-outer %boolean; "false" >
```

Description Properties

The description properties influence the descriptive text underneath the axis object.

XML Code:	`chart:axis-show-text`, `style:rotation-angle`, `chart:axis-text-overlap`, and `chart:axis-text-break`
Rules:	
DTD:	see below

```
<!ATTLIST style:properties
          chart:axis-show-text %boolean; "true"
          style:rotation-angle %integer; "0"
          chart:axis-text-overlap %boolean; "false"
          chart:axis-text-break %boolean; "true" >
```

7.9.1 Grid

Grids can be added to axis elements. If you apply a major grid to an axis, the major tickmarks are extended to gridlines. If a grid is minor, any minor tickmarks assigned to the axis are used.

XML Code:	`<chart:grid>`
Rules:	
DTD:	see below

```
<!ELEMENT chart:grid EMPTY>
<!ATTLIST chart:grid grid:class (major|minor) "major"
chart:style-name %style-name; >
```

General Properties

You can apply stroke properties to grids, which affect the lines of the grid. See Section 7.13.2 for information on these stroke properties.

7.10 Series

The series element represents a data series in a chart. The source from which chart data is retrieved is specified by the `%cell-range-address;` entity, which references a table that may reside inside the chart document or in the surrounding container document.

XML Code:	`<chart:series>`
Rules:	
Implementation limitation:	StarOffice Chart does not currently support the `chart:class` attribute for a series. You can only set this attribute for an entire chart.
DTD:	`<!ELEMENT chart:series (chart:class?, chart:domain*, chart:data-point* )>` `<!ATTLIST chart:series chart:values-cell-range-address %cell-range-address; #REQUIRED chart:label-cell-address %cell-address; #IMPLIED chart:class %chart-class; #IMPLIED chart:style-name %style-name; #IMPLIED >`

The `chart:values-cell-range-address` attribute allows you to specify a range that contains the values that should be visualized by this data series. The `chart:label-cell-address` attribute allows to provide a name for the series. If the chart requires more input data like scatter and bubble charts, you must define `chart:domain` sub-elements that mainly contain the `cell-range-address` of the corresponding data.

General Properties

You can apply fill and stroke properties for series, see Sections 7.13.1 and 7.13.2 for information. You can also apply text properties to the descriptive text underneath the series, see Section 7.13.3 for information.

7.10.1 Domain

For scatter and bubble charts, you must specify a domain for the series. For example, one `cell-range-address` value that points to the coordinate values for the scatter chart, or two `cell-range-address` values for the x and y coordinate values for bubble charts. For these chart types, you need at least one series with the necessary number of domain sub-elements. All other series can omit these, the first domain specified is used.

XML Code:	`<chart:domain>`
Rules:	
Implementation limitation:	In StarOffice Chart, you can only give one range address, which may be compound, from which values are taken in a fixed order. For example, in scatter charts the first row/column specifies the x-values for all series, the second row/column represents the values of the first series and so on.
DTD:	`<!ELEMENT chart:domain EMPTY>` `<!ATTLIST chart:domain chart:coordinate-address %cell-range-address; #REQUIRED >`

7.11 Categories

The categories element represents the range of cell addresses that contains the captions for the categories contained in each series.

XML Code:	`<chart:categories>`
Rules:	
Implementation limitation:	StarOffice Chart does not currently support the `chart:class` attribute for a series. You can only set this attribute for an entire chart.

DTD:	`<!ELEMENT chart:categories EMPTY>`
	`<!ATTLIST chart:categories`
	`table:cell-range-address %cell-range-address; >`

7.12 Data Point

If a single data point in a data series should have a specific appearance, the data point element is used to apply the required properties.

XML Code:	`<chart:data-point>`
Rules:	

DTD:	`<!ELEMENT chart:data-point>`
	`<!ATTLIST chart:data-point chart:index %nonNegativeInteger;`
	`#REQUIRED chart:style-name %style-name; >`

General Properties

You can apply fill and stroke properties to each data point object, see Sections 7.13.1 and 7.13.2. You can also apply text properties to the descriptive text located underneath the data points, see Section 7.13.3.

7.13 Common Chart Properties

The properties described in this section apply to all types of data representation objects, including the elements `<chart:plot-area>`, `<chart:series>`, and `<chart:data-point>`.

Properties are applied in a hierarchical manner. If a property is set in the `<chart:chart>` element, it applies to all data points contained in the chart. If the same property is set in a `<chart:series>` element, it only applies to the data points contained in that specific series. To set a formatting property for one data point only, you should set the property in the `<chart:data-point>` element.

7.13.1 Fill Properties

The fill properties apply to all solid objects like rectangles or circles. See Section 5.9 for information on fill properties.

7.13.2 Stroke Properties

The stroke properties apply to all line objects like the axis, grid, or linear parts of a rectangle or circle. See 5.8 for information stroke properties.

7.13.3 Text Properties

The text properties apply to all objects that display text, for example, the `legend`, `title`, `subtitle`, `axis`, `chart`, `series`, and `data-point`. See Section 3.1.6 for information on text properties.

7.13.4 Alignment Properties

The alignment properties described in this section apply to several text objects. They determine the way text is positioned inside the surrounding box.

Stacked Text

This property determines whether or not text is displayed vertically without rotating the letters. To display the text vertically, set the property value to `true`.

XML Code:	`chart:text-stacked`
Rules:	The value of this property can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:text-stacked %boolean; "false" >`

Rotation Angle

This property specifies the value of a rotation angle in degrees. See Section 4.20.13 for information on using this property.

7.13.5 Data Label Properties

Data labels can be applied to data series and data points as well as to an entire chart. In the latter case, labels are shown for all data points. Data labels can consist of the following three parts:

- The value, which can be displayed as a percentage or the value itself.
- The label of the corresponding series.
- The legend symbol.

Value

This attribute represents the value of the data label.

XML Code:	`chart:data-label-number`
Rules:	
DTD:	`<!ATTLIST style:properties chart:data-label-number (none|value|percentage) "none" >`

Label

This attribute determines whether or not to display the label of the corresponding series.

XML Code:	`chart:data-label-text`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:data-label-text %boolean; "false" >`

Legend Symbol

This attribute determines whether or not to display the legend symbol.

XML Code:	`chart:data-label-symbol`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:data-label-symbol %boolean; "false" >`

Glossary

Term	***Definition***
attributes	Attributes are used to associate name-value pairs with elements. Each attribute specification has a name and a value. Attribute specifications may appear only within start-tags and empty-element tags.
automatic styles	A style that is automatically generated when the document is exported. The automatic style is assigned to objects such as paragraphs, so that a separation of content and layout is achieved.
body group	A body group is a group of rows or columns that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.
characters	Parsed data is made up of characters, some of which form character data, and some of which form markup.
collapsing border model	This is a model of how to represent table borders. The collapsing border model means that when two adjacent cells have different borders, the wider border appears as the border between the cells. Each cell receives half of the width of the border.
column group	You can group columns in column groups. These groups specify whether or not to repeat a column on the next page. Column groups can be either body groups or header groups. A table must contain at least one column body group, but only one column header group.
comments	Comments may appear anywhere in a document outside other markup. In addition, they may appear within the document type declaration at places allowed by the grammar. They are not part of the character data of a document. An XML processor may, but need not, make it possible for an application to retrieve the text of comments.
common styles	The style that a StarOffice user who doesn't care about the StarOffice XML file format expects to be a style. The term *common styles* is used to differentiate in cases where there might be confusion with *automatic styles*, otherwise it is the same as *styles*.
content	The text between the start-tag and end-tag is the content of an *element*.
CSS	Cascading Style Sheet
CSS2	Cascading style sheets, level 2.
current file format version	This version stores all the information contained in the document without losing any information when the document is read again.
DDE	Dynamic Data Exchange.
document element	A StarOffice XML document begins with a document *element*.

Term	*Definition*
document root	The document root element is the primary element for defining the characteristics of an XML document.
document type declaration	The document type declaration can point to an external subset containing markup declarations, or can contain the markup declarations directly in an internal subset, or can do both. The DTD for a document consists of both subsets taken together.
drawing page	The main location for content in a drawing or presentation document.
DTD	The XML document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD.
elements	Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, and may have a set of attribute specifications.
entity	XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Entities may refer to other entities to cause their inclusion in the document.
EOL	End-of-line
external styles	Styles not contained within the XML file that contains the content.
generic identifier	The type of an element.
GI	See *generic identifier*.
header group	A header group is a group of rows or columns that repeat on each page if the table extends over several pages.
internal styles	Styles contained within the XML file that contains the content.
markup	Code description of the storage layout and logical structure of an XML document. Markup takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, and processing instructions.
master page	The common background for a drawing page. Each drawing page must be linked to one master page.
metadata	Metadata is general information about the document, such as title, author, creation date, and so on.
name	A token beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters.
namespaces	Namespaces allow you to define specific names for elements. The purpose of namespaces is to avoid conflicts between documents from various authors with different naming conventions.
nmtoken	Name token; any mixture of name characters.
orphan	An orphan is a short line at the start of a paragraph, which when printed, appears alone at the bottom of the page.
page master	A page-master specifies the size, border and orientation of a master page, q.v.
row group	You can group rows in row groups. These groups specify whether or not to repeat a row on the next page. Row groups can be either body groups or header groups. A table must contain at least one row body group, but only one row header group.

Term	*Definition*
separating border model	This is a model of how to represent table borders. The separating border model means that borders appear within the cell that specifies the border.
SGML	Standard Generalized Markup Language.
stroke properties	SVG stroke properties define graphic object line characteristics in StarOffice Draw and StarOffice Impress documents:
styles	The XML representations of the styles that are available in the StarOffice user interface.
subtable	A subtable is a table within another table. It occupies one cell and no other content can appear in this cell. If a table cell contains a subtable, it cannot contain any paragraphs. Subtables are sometimes referred to as nested tables.
SVG	Scalar Vector Graphics. An XML language for creating vector graphics in Internet-viewable documents.
type	The *name* in the start-tags and end-tags gives the element type, see also *generic identifier*.
valid	An XML document is valid if it has an associated document type declaration and if the document complies with the constraints expressed in it.
well-formed document	In a well-formed document, the logical and physical structures in an XML document are properly nested. Consequently, no start-tag, end-tag, empty-element tag, element, comment, processing instruction, character reference, or entity reference can begin in one entity and end in another.
W3C	World Wide Web Committee.
widow	A widow is a short line at the end of a paragraph, which when printed, appear alone at the top of the next page.
XLinks	Hyperlinks in XML documents, identified by the `xml:links` attribute. XLink governs how you insert links into your XML document, and where the link might points to.
XML	The eXtensible Markup Language (XML) is a subset of SGML described in the W3C XML Specification. The goal of XML is to enable generic SGML to be served, received, and processed on the Web in a way comparable to HTML. XML provides ease of implementation and interoperability with both SGML and HTML.
XML documents	A class of data objects described by XML. XML documents are conforming SGML documents.
XML processor	A software module used to read XML documents and provide access to their content and structure.
XPath	XML Path Language; a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer.
XSL	Extensible Style Sheet Language; a stylesheet language for XML. When you use XSL all your documents are formatted the same way, no matter which application or platform they are on.
XSLT	XSLT is a language for transforming XML documents into other XML documents. XSLT is designed for use as part of XSL, or independently of XSL.
XPointer	XPointer governs the fragment identifier that can go on a URL when you're linking to an XML document from anywhere, for example from an HTML file. XPointer and XLinks are part of the same package.

Index

Index