Open Document Format for Office Applications (OpenDocument) v1.1

OASIS Standard, 1 Feb 2007

Specification URIs:

This Version:

http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.odt
http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf
http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.html.zip

Previous Version:

http://www.oasis-open.org/committees/download.php/19275/OpenDocument-v1.0ed2-cs1.odt
http://www.oasis-open.org/committees/download.php/19274/OpenDocument-v1.0ed2-cs1.pdf

Latest Version:

http://docs.oasis-open.org/office/v1.1/OpenDocument-v1.1.odt
http://docs.oasis-open.org/office/v1.1/OpenDocument-v1.1.pdf
http://docs.oasis-open.org/office/v1.1/OpenDocument-v1.1.html.zip

Latest Approved Version:

http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.odt
http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf
http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.html.zip

Technical Committee:

OASIS Open Document Format for Office Applications (OpenDocument) TC

Chair:

Michael Brauer, Sun Microsystems, Inc.

Editors:

Patrick Durusau, Individual
Michael Brauer, Sun Microsystems, Inc.
Lars Oppermann, Sun Microsystems, Inc.

Related Work:

This specification supersedes OASIS OpenDocument v1.0.

Declared XML Namespaces:

A list of XML namespaces declared by this specification is available in section 1.3.

Abstract:

This is the specification of the Open Document Format for Office Applications (OpenDocument) format, an open, XML-based file format for office applications, based on OpenOffice.org XML [OOo].

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at

www.oasis-open.org/committees/office.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page

(www.oasis-open.org/committees/office/ipr.php.

The non-normative errata page for this specification is located at www.oasis-open.org/committees/office.

Notices

Table of Contents

1 Introduction

1.1 Introduction

This document defines an XML schema for office applications and its semantics. The schema is suitable for office documents, including text documents, spreadsheets, charts and graphical documents like drawings or presentations, but is not restricted to these kinds of documents.

The schema provides for high-level information suitable for editing documents. It defines suitable XML structures for office documents and is friendly to transformations using XSLT or similar XML-based tools.

Chapter 1 contains the introduction to the OpenDocument format. The structure of documents that conform to the OpenDocument specification is explained in chapter 2. Chapter 3 described the meta information that can be contained in such documents. Chapters 4 and 5 describe their text and paragraph content. Text Fields are described in chapter 6, text indices in chapter 7.

Chapter 8 describes the table content of a document in OpenDocument format, chapter 9 its graphical content, chapter 10 its chart content, and chapter 11 its form content. Content that is common to all documents is described in chapter 12. The integration of SMIL animation markup into the OpenDocument schema is described in chapter 13. Chapter 14 explains style information content, chapter 15 specifies formatting properties that are can be used within styles. The data types used by the OpenDocument schema are described in chapter 16.

The OpenDocument format makes use of a package concept. These packages are described in chapter 17.

1.2 Notation

Within this specification, the key words "**shall**", "**shall not**", "**should**", "**should not**" and "**may**" are to be interpreted as described in Annex H of [ISO/IEC Directives] if they appear in bold letters.

1.3 Namespaces

Table 1 lists the namespaces that are defined by the OpenDocument format and their default prefixes. For more information about XML namespaces, please refer to the *Namespaces in XML* specification [xml-names].

Table 1: XML Namespaces defined by the OpenDocument schema

Prefix	***Description***	***Namespace***
office	For all common pieces of information that are not contained in another, more specific namespace.	urn:oasis:names:tc:opendocument:xmlns:office:1.0
meta	For elements and attributes that describe meta information.	urn:oasis:names:tc:opendocument:xmlns:meta:1.0
config	For elements and attributes that describe application specific settings.	urn:oasis:names:tc:opendocument:xmlns:config:1.0

Prefix	***Description***	***Namespace***
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.	urn:oasis:names:tc:opendocument:xmlns:text:1.0
table	For elements and attributes that may occur within spreadsheets or within table definitions of a text document.	urn:oasis:names:tc:opendocument:xmlns:table:1.0
drawing	For elements and attributes that describe graphic content.	urn:oasis:names:tc:opendocument:xmlns:drawing:1.0
presentation	For elements and attributes that describe presentation content.	urn:oasis:names:tc:opendocument:xmlns:presentation:1.0
dr3d	For elements and attributes that describe 3D graphic content.	urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0
anim	For elements and attributes that describe animation content.	urn:oasis:names:tc:opendocument:xmlns:animation:1.0
chart	For elements and attributes that describe chart content.	urn:oasis:names:tc:opendocument:xmlns:chart:1.0
form	For elements and attributes that describe forms and controls.	urn:oasis:names:tc:opendocument:xmlns:form:1.0
script	For elements and attributes that represent scripts or events.	urn:oasis:names:tc:opendocument:xmlns:script:1.0
style	For elements and attributes that describe the style and inheritance model used by the OpenDocument format as well as some common formatting attributes.	urn:oasis:names:tc:opendocument:xmlns:style:1.0
number	For elements and attributes that describe data style information.	urn:oasis:names:tc:opendocument:xmlns:data style:1.0
manifest	For elements and attribute contained in the package manifest.	urn:oasis:names:tc:opendocument:xmlns:manifest:1.0

Table 2 lists the namespaces that are defined by the OpenDocument format, but contain elements and attributes whose semantics are compatible to elements and attributes from other specifications.

Table 2: XML Namespaces defined by the OpenDocument schema that include elements and attributes that are compatible to elements and attributes of other standards.

Prefix	***Description***	***Namespace***
fo	For attributes that are compatible to attributes defined in [XSL].	urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0
svg	For elements and attributes that are compatible to elements or attributes defined in [SVG].	urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0

Prefix	Description	Namespace
smil	For attributes that are compatible to attributes defined in [SMIL20].	urn:oasis:names:tc:opendocument:xmlns:smil-compatible:1.0

Table 3 lists the namespaces that are imported into the OpenDocument format and their default prefixes.

Table 3: XML Namespaces used by the OpenDocument schema

Prefix	Description	Namespace
dc	The Dublin Core Namespace (see [DCMI]).	http://purl.org/dc/elements/1.1/
xlink	The XLink namespace (see [XLink]).	http://www.w3.org/1999/xlink
math	MathML Namespace (see [MathML])	http://www.w3.org/1998/Math/MathML
xforms	The XForms namespace (see [XForms]).	http://www.w3.org/2002/xforms

1.4 Relax-NG Schema

The normative XML Schema for the OpenDocument format is embedded within this specification. It can be obtained from the specification document by concatenating all schema fragments contained in chapters 1 to 16. All schema fragments have a gray background color and line numbers.

The schema language used within this specification is Relax-NG (see [RNG]). The attribute default value feature specified in [RNG-Compat] is used to provide attribute default values.

The schema provided in this specification permits arbitrary content within meta information elements and formatting properties elements as described in section 1.5. Appendix A contains a schema that restricts the content within these elements to the attributes and elements defined in this specification.

Prefix for the normative Relax-NG schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    OASIS OpenDocument v1.1
    OASIS Standard, 1 Feb 2007
    Relax-NG Schema

    $Id$

    © 2002-2007 OASIS Open
    © 1999-2007 Sun Microsystems, Inc.
-->

<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"

    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"

    xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
    xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
    xmlns:config="urn:oasis:names:tc:opendocument:xmlns:config:1.0"
    xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
    xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
    xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0"
    xmlns:presentation="urn:oasis:names:tc:opendocument:xmlns:presentation:1.0"
    xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0"
    xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0"
    xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
    xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
    xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
    xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
    xmlns:anim="urn:oasis:names:tc:opendocument:xmlns:animation:1.0"

    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:math="http://www.w3.org/1998/Math/MathML"
    xmlns:xforms="http://www.w3.org/2002/xforms"

    xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0"
    xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0"
    xmlns:smil="urn:oasis:names:tc:opendocument:xmlns:smil-compatible:1.0"
>
```

1.5 Document Processing and Conformance

Documents that conform to the OpenDocument specification **may** contain elements and attributes not specified within the OpenDocument schema. Such elements and attributes must not be part of a namespace that is defined within this specification and are called foreign elements and attributes.

Conforming applications either **shall** read documents that are valid against the OpenDocument schema if all foreign elements and attributes are removed before validation takes place, or **shall** write documents that are valid against the OpenDocument schema if all foreign elements and attributes are removed before validation takes place.

Conforming applications that read and write documents **may** preserve foreign elements and attributes.

In addition to this, conforming applications should preserve meta information and the content of styles. This means:

- The various `<style:*-properties>` elements (see section 15) **may** have arbitrary attributes attached and **may** have arbitrary element content. All attributes attached to these elements and elements contained within these elements **should** be preserved (see section 15.1.3);
- elements contained within the `<office:meta>` element **may** have arbitrary element content and **should** be preserved (see section 2.2.1).

Foreign elements **may** have an `office:process-content` attribute attached that has the value `true` or `false`. If the attribute's value is `true`, or if the attribute does not exist, the element's content **should** be processed by conforming applications. Otherwise conforming applications **should not** process the element's content, but **may** only preserve its content. If the element's content should be processed, the document itself **shall** be valid against the OpenDocument schema if the unknown element is replaced with its content only.

Conforming applications **shall** read documents containing processing instructions and **should** preserve them.

There are no rules regarding the elements and attributes that actually have to be supported by conforming applications, except that applications should not use foreign elements and attributes for features defined in the OpenDocument schema. See also appendix D.

```
<define name="office-process-content">
    <optional>
        <attribute name="office:process-content" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

1.6 White-Space Processing and EOL Handling

In conformance with the **W3C** XML specification [XML1.0], optional white-space characters that are contained in elements that have element content (in other words that must contain elements only but not text) are ignored. This 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.

1.7 MIME Types and File Name Extensions

Appendix C contains a list of MIME types and file name extensions to be used for office documents that conform to this specification and that are contained in a package (see section 2.1). This MIME types and extensions either have been registered following the procedures described in [RFC2048], or a registration is in progress.

Office documents that conform to this specification but are not contained in a package **should** use the MIME type `text/xml`.

Only MIME types and extensions that have been registered according to [RFC2048] **should** used for office documents that conform to this specification. The MIME types and extensions listed in appendix C **should** be used where appropriate.

2 Document Structure

This chapter introduces the structure of the OpenDocument format. The chapter contains the following sections:

- Document Roots
- Document Metadata
- Body Element and Document Types
- Application Settings
- Scripts
- Font Face Declarations
- Styles
- Page Styles and Layout

In the OpenDocument format, each structural component is represented by an **element**, with associated **attributes**. The structure of a document in OpenDocument format applies to all document types. 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. Document content that is common to all document types can be exchanged from one type of document to another.

2.1 Document Roots

A **document root element** is the primary element of a document in OpenDocument format. It contains the entire document. All types of documents, for example, text documents, spreadsheets, and drawing documents use the same types of document root elements.

The OpenDocument format supports the following two ways of document representation:

- As a single XML document.
- As a collection of several subdocuments within a package (see section 17), each of which stores part of the complete document. Each subdocument has a different document root and stores a particular aspect of the XML document. For example, one subdocument contains the style information and another subdocument contains the content of the document. All types of documents, for example, text and spreadsheet documents, use the same document and subdocuments definitions.

There are four types of subdocuments, each with different root elements. Additionally, the single XML document has its own root element, for a total of five different supported root elements. The root elements are summarized in the following table:

Root Element	***Subdocument Content***	***Subdoc. Name in Package***
`<office:document>`	Complete office document in a single XML document.	n/a
`<office:document-content>`	Document content and automatic styles used in the content.	content.xml

Root Element	***Subdocument Content***	***Subdoc. Name in Package***
`<office:document-styles>`	Styles used in the document content and automatic styles used in the styles themselves.	styles.xml
`<office:document-meta>`	Document meta information, such as the author or the time of the last save action.	meta.xml
`<office:document-settings>`	Application-specific settings, such as the window size or printer information.	settings.xml

The definitions of the root elements described in the table above are analogous to the definition of `<office:document>`, except that the child element specification is suitably restricted.

```
<start>
    <choice>
        <ref name="office-document"/>
        <ref name="office-document-content"/>
        <ref name="office-document-styles"/>
        <ref name="office-document-meta"/>
        <ref name="office-document-settings"/>
    </choice>
</start>
```

2.1.1 Document Root Element Content Models

The content models of the five root elements is summarized in the following table. Note that `<office:document>` may contain all supported top-level elements. None of the four subdocument root elements contain the complete data, but four combined do.

Root Element	***meta-data***	***app. sett.***	***script***	***font decls***	***style***	***auto style***	***mast style***	***body***
`<office:document>`	✓	✓	✓	✓	✓	✓	✓	✓
`<office:document-content>`			✓	✓		✓		✓
`<office:document-styles>`				✓	✓	✓	✓	
`<office:document-meta>`	✓							
`<office:document-settings>`		✓						

The `<office:document>` root contains a complete document:

```
<define name="office-document">
    <element name="office:document">
        <ref name="office-document-attrs"/>
        <ref name="office-document-common-attrs"/>
        <ref name="office-meta"/>
        <ref name="office-settings"/>
        <ref name="office-scripts"/>
        <ref name="office-font-face-decls"/>
        <ref name="office-styles"/>
        <ref name="office-automatic-styles"/>
        <ref name="office-master-styles"/>
        <ref name="office-body"/>
    </element>
</define>
```

The `<office:document-content>` root contains only the document content, along with the automatic styles needed for the document content:

```
<define name="office-document-content">
    <element name="office:document-content">
        <ref name="office-document-common-attrs"/>
        <ref name="office-scripts"/>
        <ref name="office-font-face-decls"/>
        <ref name="office-automatic-styles"/>
        <ref name="office-body"/>
    </element>
</define>
```

The `<office:document-styles>` root contains all named styles of a document, along with the automatic styles needed for the named styles:

```
<define name="office-document-styles">
    <element name="office:document-styles">
        <ref name="office-document-common-attrs"/>
        <ref name="office-font-face-decls"/>
        <ref name="office-styles"/>
        <ref name="office-automatic-styles"/>
        <ref name="office-master-styles"/>
    </element>
</define>
```

The `<office:document-meta>` root contains the meta information about a document.

```
<define name="office-document-meta">
    <element name="office:document-meta">
        <ref name="office-document-common-attrs"/>
        <ref name="office-meta"/>
    </element>
</define>
```

The `<office:document-settings>` root contains application specific settings to be applied when processing this document.

```
<define name="office-document-settings">
    <element name="office:document-settings">
        <ref name="office-document-common-attrs"/>
        <ref name="office-settings"/>
    </element>
</define>
```

2.1.2 Document Root Attributes

Version

All root elements take an `office:version` attribute, which indicates which version of this specification it complies with. The version number is in the format `revision.version`. If the file has a version known to an XML processor, it may validate the document. Otherwise, it is optional to validate the document, but the document must be well formed.

```
<define name="office-document-common-attrs" combine="interleave">
    <optional>
        <attribute name="office:version">
```

```
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

MIME Type

The `<office:document>` element takes an `office:mimetype` attribute, which indicates the type of document (text, spreadsheet etc.). This attribute is especially important for flat XML files, where this is the only way the type of document can be detected (in a package, the MIME type is also present in a separate file, see section 17.4). Its values are the MIME types that are used for the packaged variant of office documents (see section 1.7).

```
<define name="office-document-attrs" combine="interleave">
    <attribute name="office:mimetype">
        <ref name="string"/>
    </attribute>
</define>
```

2.2 Document Metadata

Metadata is general information about a document. In the OpenDocument format, all of the metadata elements are contained in an `<office:meta>` element, usually located at start of the document. Metadata elements may be omitted or occur multiple times. It is application-specific how to update multiple instances of the same elements.

```
<define name="office-meta">
    <optional>
        <element name="office:meta">
            <ref name="office-meta-content"/>
        </element>
    </optional>
</define>

<define name="office-meta-content">
    <ref name="anyElements"/>
</define>

<define name="office-meta-content-strict">
    <zeroOrMore>
        <ref name="office-meta-data"/>
    </zeroOrMore>
</define>
```

2.2.1 Pre-Defined vs. Custom Metadata

In the OpenDocument schema the metadata is comprised of pre-defined metadata elements, user defined metadata, as well as custom metadata elements. The pre-defined metadata elements have defined semantics. They **should** be processed and updated by editing applications. They can be referenced from within the document through the use of suitable text fields.

User-defined metadata is a more generic mechanism which specifies a triplet of name, type, and value. Supporting applications can present these value to the user, making use of the supplied data type. The user-defined metadata can be referenced from within the document through the use of suitable text fields.

Custom metadata are arbitrary elements inside `<office:meta>`. Since their semantics is not defined in this specification, conforming applications in general cannot process or display this data. Applications **should** preserve this data when editing the document.

2.2.2 Sample Metadata

Example: Sample metadata of a document in OpenDocument format

```
<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:editing-duration>PT5H10M10S</meta:editing-duration>
    <meta:keyword>First keyword</meta:keyword>
    <meta:keyword>Second keyword</meta:keyword>
    <meta:keyword>Third keyword</meta:keyword>
    <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"
    meta:value-type="string">Value 1</meta:user-defined>
    <meta:user-defined meta:name="Field 2"
    meta:value-type="float">1.234</meta:user-defined>
</office:meta>
```

2.3 Body Element and Document Types

The document body contains an element to indicate which type of content this document contains. Currently supported document types are:

- text documents
- drawing documents
- presentation documents
- spreadsheet documents
- chart documents
- image documents

All document types share the same content elements, but different document types place different restrictions on which elements may occur, and in what combinations. The document content is typically framed by a prelude and epilogue, which contain additional information for a specific type of document, like form data or variable declarations.

```
<define name="office-body">
```

```
    <element name="office:body">
        <ref name="office-body-content"/>
    </element>
</define>
```

2.3.1 Text Documents

The content of text documents mainly consists of a sequence containing any number of paragraphs, tables, indices, text frames, text sections, and graphical elements. Additionally, a text document may contain forms, change tracking information and variable declarations. Each of these is defined in the document prelude, and may be referenced from the document content.

```
<define name="office-body-content" combine="choice">
    <element name="office:text">
        <ref name="office-text-attlist"/>
        <ref name="office-text-content-prelude"/>
        <zeroOrMore>
            <ref name="office-text-content-main"/>
        </zeroOrMore>
        <ref name="office-text-content-epilogue"/>
    </element>
</define>
```

Text Document Content Model

The text document prelude contains the document's form data, change tracking information, and variable declarations. To allow office applications to implement functionality that usually is available in spreadsheets for text documents, it may also contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-text-content-prelude">
    <ref name="office-forms"/>
    <ref name="text-tracked-changes"/>
    <ref name="text-decls"/>
    <ref name="table-decls"/>
</define>
```

The main document content contains any sequence of text content elements, which includes paragraphs (and headings), text sections (and indices), tables, and graphical shapes. As an alternative, a text document may contain of a single page sequence.

It is not required that a text document contains a paragraph. A text document may consist of a sequence frames only.

```
<define name="office-text-content-main">
    <choice>
        <zeroOrMore>
            <ref name="text-content"/>
        </zeroOrMore>
        <group>
            <ref name="text-page-sequence"/>
            <zeroOrMore>
                <choice>
                    <ref name="draw-a"/>
                    <ref name="shape"/>
                </choice>
            </zeroOrMore>
        </group>
    </choice>
</define>

```

```
<define name="text-content">
    <choice>
        <ref name="text-h"/>
        <ref name="text-p"/>
        <ref name="text-list"/>
        <ref name="text-numbered-paragraph"/>
        <ref name="table-table"/>
        <ref name="draw-a"/>
        <ref name="text-section"/>
        <ref name="text-soft-page-break"/>
        <ref name="text-table-of-content"/>
        <ref name="text-illustration-index"/>
        <ref name="text-table-index"/>
        <ref name="text-object-index"/>
        <ref name="text-user-index"/>
        <ref name="text-alphabetical-index"/>
        <ref name="text-bibliography"/>
        <ref name="shape"/>
        <ref name="change-marks"/>
    </choice>
</define>
```

There are no text documents specific epilogue elements, but the epilogue may contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-text-content-epilogue">
    <ref name="table-functions"/>
</define>
```

Global Text Documents

There is a common use case for large documents to be edited in separate entities, such that there is a 'global' document, containing several linked constituent subdocuments. This can be implemented by using linked text sections (see section 4.4). To facilitate an editing application adapting the user interface to better support the notion of 'global' document with constituent parts (as opposed to a document with arbitrary linked content), the `text:global` flag can be used. If set to `true`, it informs applications that linked sections in this document have part-of semantics. The actual XML representation of the sections does not change.

```
<define name="office-text-attlist" combine="interleave">
    <optional>
        <attribute name="text:global" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Use Soft Page Breaks

The `text:use-soft-page-breaks` attribute specifies whether the document contains soft page breaks.

A soft page break is a page break that a has been included by a page oriented processor at a position where the document itself does not include a page break (e.g. by using the `fo:break-before` and `fo:break-after` formatting properties described in section 15.5.2).

Soft page breaks are specified by the `<text:soft-page-break>` elements described in sections 4.7 and 5.1.1:Soft Page breaks.

The use of the `<text:soft-page-break>` elements is always optional. An application generating the format **may** include the element if it has computed a paginated layout. A consuming application **may** handle the element while computing the layout, but it **shall not** depend on its existence. Soft page breaks are only supported within text documents.

A generating application that stores soft page breaks **shall** indicate this by setting the `text:use-page-breaks` attribute to `true`. A generating application that does not store soft page breaks **shall** indicate that by omitting this attribute, or by setting it to false.

An application that does not support pagination and soft page-breaks, that modifies an OpenDocument file, which includes soft page-breaks, **shall** at least set the `text:use-page-breaks` attribute to `false` (or remove it). It **should** also remove the text:soft-page-break elements from the document but is not required to do so.

An application that computes a paginated layout of a document **should** provide a facility to turn on export of soft page breaks for the purposes of consistent page breaks and for proper conversion to digital talking book formats (such as [DAISY]).

For `<text:soft-page-break>` elements that appear within table rows, the maximum number of `<text:soft-page-break>` elements that appear within the single table cells determines the number of page breaks that appear within the table row. The `<text:soft-page-break>` elements contained in each cell determine the positions where these page breaks appear within the cell content.

Similarly, `<text:soft-page-break>` elements that appear within text boxes and other content displayed outside the text flow, do not start a new page, but only indicate where the text-box's content breaks between two pages.

```
<define name="office-text-attlist" combine="interleave">
    <optional>
        <attribute name="text:use-soft-page-breaks" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

2.3.2 Drawing Documents

The content of drawing document consists of a sequence of draw pages.

```
<define name="office-body-content" combine="choice">
    <element name="office:drawing">
        <ref name="office-drawing-attlist"/>
        <ref name="office-drawing-content-prelude"/>
        <ref name="office-drawing-content-main"/>
        <ref name="office-drawing-content-epilogue"/>
    </element>
</define>

<define name="office-drawing-attlist">
    <empty/>
</define>
```

Drawing Document Content Model

The drawing document prelude may contain text declarations only. To allow office applications to implement functionality that usually is available in spreadsheets for drawing documents, it may also contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-drawing-content-prelude">
```

```
    <ref name="text-decls"/>
    <ref name="table-decls"/>
</define>
```

The main document content contains a sequence of draw pages.

```
<define name="office-drawing-content-main">
    <zeroOrMore>
        <ref name="draw-page"/>
    </zeroOrMore>
</define>
```

There are no drawing documents specific epilogue elements, but the epilogue may contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-drawing-content-epilogue">
    <ref name="table-functions"/>
</define>
```

2.3.3 Presentation Documents

The content of presentation document consists of a sequence of draw pages.

```
<define name="office-body-content" combine="choice">
    <element name="office:presentation">
        <ref name="office-presentation-attlist"/>
        <ref name="office-presentation-content-prelude"/>
        <ref name="office-presentation-content-main"/>
        <ref name="office-presentation-content-epilogue"/>
    </element>
</define>

<define name="office-presentation-attlist">
    <empty/>
</define>
```

Presentation Document Content Model

The presentation document prelude equals the one of a drawing document, but may contain some additional declarations. See also section 2.3.2.

```
<define name="office-presentation-content-prelude">
    <ref name="text-decls"/>
    <ref name="table-decls"/>
    <ref name="presentation-decls"/>
</define>
```

The main document content contains a sequence of draw pages.

```
<define name="office-presentation-content-main">
    <zeroOrMore>
        <ref name="draw-page"/>
    </zeroOrMore>
</define>
```

The epilogue of presentation documents may contain presentation settings. Additionally, it may contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-presentation-content-epilogue">
    <ref name="presentation-settings"/>
    <ref name="table-functions"/>
</define>
```

2.3.4 Spreadsheet Documents

The content of spreadsheet documents mainly consists of a sequence of tables. Additionally, a spreadsheet document may contain forms, change tracking information and various kinds of declarations that simplify the usage of spreadsheet tables and their analysis. Each of these are contained in either the document prelude, or the document epilogue.

```
<define name="office-body-content" combine="choice">
    <element name="office:spreadsheet">
        <ref name="office-spreadsheet-attlist"/>
        <ref name="office-spreadsheet-content-prelude"/>
        <ref name="office-spreadsheet-content-main"/>
        <ref name="office-spreadsheet-content-epilogue"/>
    </element>
</define>
```

Spreadsheet Document Content Model

The spreadsheet document prelude contains the document's form data, change tracking information, calculation setting for formulas, validation rules for cell content and declarations for label ranges.

```
<define name="office-spreadsheet-content-prelude">
    <optional>
        <ref name="table-tracked-changes"/>
    </optional>
    <ref name="text-decls"/>
    <ref name="table-decls"/>
</define>

<define name="table-decls">
    <optional>
        <ref name="table-calculation-settings"/>
    </optional>
    <optional>
        <ref name="table-content-validations"/>
    </optional>
    <optional>
        <ref name="table-label-ranges"/>
    </optional>
</define>
```

The main document is a list of tables.

```
<define name="office-spreadsheet-content-main">
    <zeroOrMore>
        <ref name="table-table"/>
    </zeroOrMore>
</define>
```

The epilogue of spreadsheet documents contains declarations for named expressions, database ranges, data pilot tables, consolidation operations and DDE links.

```
<define name="office-spreadsheet-content-epilogue">
    <ref name="table-functions"/>
</define>

<define name="table-functions">
    <optional>
        <ref name="table-named-expressions"/>
    </optional>
    <optional>
```

```
        <ref name="table-database-ranges"/>
    </optional>
    <optional>
        <ref name="table-data-pilot-tables"/>
    </optional>
    <optional>
        <ref name="table-consolidation"/>
    </optional>
    <optional>
        <ref name="table-dde-links"/>
    </optional>
</define>
```

2.3.5 Chart Documents

The content of chart documents mainly consists of a chart element.

```
<define name="office-body-content" combine="choice">
    <element name="office:chart">
        <ref name="office-chart-attlist"/>
        <ref name="office-chart-content-prelude"/>
        <ref name="office-chart-content-main"/>
        <ref name="office-chart-content-epilogue"/>
    </element>
</define>

<define name="office-chart-attlist">
    <empty/>
</define>
```

Chart Document Content Model

To allow office applications to implement functionality that usually is available in spreadsheets for the table that may be contained in a chart, the chart document prelude may contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-chart-content-prelude">
    <ref name="text-decls"/>
    <ref name="table-decls"/>
</define>
```

The main document is a chart element only.

```
<define name="office-chart-content-main">
    <ref name="chart-chart"/>
</define>
```

There are no chart documents specific epilogue elements, but the epilogue may contain elements that implement enhanced table features. See also section 2.3.4.

```
<define name="office-chart-content-epilogue">
    <ref name="table-functions"/>
</define>
```

2.3.6 Image Documents

The content of an image document is a frame element only. The frame element must contain a single image element.

```
<define name="office-body-content" combine="choice">
    <element name="office:image">
```

```
        <ref name="office-image-attlist"/>
        <ref name="office-image-content-prelude"/>
        <ref name="office-image-content-main"/>
        <ref name="office-image-content-epilogue"/>
    </element>
</define>

<define name="office-image-attlist">
    <empty/>
</define>
```

Image Document Content Model

The image document prelude is empty.

```
<define name="office-image-content-prelude">
    <empty/>
</define>
```

The main document content contains a frame only.

```
<define name="office-image-content-main">
    <ref name="draw-frame"/>
</define>
```

There are no image documents specific epilogue elements.

```
<define name="office-image-content-epilogue">
    <empty/>
</define>
```

2.4 Application Settings

Application settings are contained in a `<office:settings>` element.

```
<define name="office-settings">
    <optional>
        <element name="office:settings">
            <oneOrMore>
                <ref name="config-config-item-set"/>
            </oneOrMore>
        </element>
    </optional>
</define>
```

The settings for office applications may be divided into several categories each represented by a `<config:config-item-set>` element. For instance the following two categories may exist:

- Document settings, for example default printer.
- View settings, for example zoom level.

2.4.1 Sequence of Settings

The `<config:config-item-set>` element is a container element for all types of setting elements. The settings can be contained in the element is any order.

```
<define name="config-config-item-set">
    <element name="config:config-item-set">
        <ref name="config-config-item-set-attlist"/>
        <ref name="config-items"/>
    </element>
</define>

<define name="config-items">
    <oneOrMore>
        <choice>
            <ref name="config-config-item"/>
            <ref name="config-config-item-set"/>
            <ref name="config-config-item-map-named"/>
            <ref name="config-config-item-map-indexed"/>
        </choice>
    </oneOrMore>
</define>
```

Config Name

The `config:name` attribute identifies the name of the setting container. For top level `<config:config-item-set>` elements, that are elements that are direct children of the `<office:settings>` element, the name should be preceded by a namespace prefix that identifies the application the settings belong to.

```
<define name="config-config-item-set-attlist" combine="interleave">
    <attribute name="config:name">
        <ref name="string"/>
    </attribute>
</define>
```

Example:

```
<office:settings>
    <config:config-item-set xmlns:ooo="http://www.openoffice.org/...";
                            config:name="ooo:view-settings">
        <config:config-item config:name="ViewAreaTop"
                            config:type="int">0</config:config-item>
    </config:config-item-set>
</office:settings>
```

2.4.2 Base Settings

The `<config:config-item>` element contains all base settings. The value of the setting is stored in the element.

```
<define name="config-config-item">
    <element name="config:config-item">
        <ref name="config-config-item-attlist"/>
        <text/>
    </element>
</define>
```

Config Name

The `config:name` attribute identifies the name of the setting.

```
<define name="config-config-item-attlist" combine="interleave">
    <attribute name="config:name">
        <ref name="string"/>
    </attribute>
</define>
```

Config Type

The `config:type` attribute identifies the data type of setting.

```
<define name="config-config-item-attlist" combine="interleave">
    <attribute name="config:type">
        <choice>
            <value>boolean</value>
            <value>short</value>
            <value>int</value>
            <value>long</value>
            <value>double</value>
            <value>string</value>
            <value>datetime</value>
            <value>base64Binary</value>
        </choice>
    </attribute>
</define>
```

2.4.3 Index Access of Sequences

The `<config:config-item-map-indexed>` element is a container element for sequences. The order specifies the index of the elements

```
<define name="config-config-item-map-indexed">
    <element name="config:config-item-map-indexed">
        <ref name="config-config-item-map-indexed-attlist"/>
        <oneOrMore>
            <ref name="config-config-item-map-entry"/>
        </oneOrMore>
    </element>
</define>
```

Config Name

The `config:name` attribute identifies the name of the setting sequence.

```
<define name="config-config-item-map-indexed-attlist" combine="interleave">
    <attribute name="config:name">
        <ref name="string"/>
    </attribute>
</define>
```

2.4.4 Map Entry

The `<config:config-item-map-entry>` element represents an entry in an indexed or named settings sequence. It is a container element for all types of setting elements.

```
<define name="config-config-item-map-entry">
    <element name="config:config-item-map-entry">
        <ref name="config-config-item-map-entry-attlist"/>
        <ref name="config-items"/>
    </element>
</define>
```

Config Name

The `config:name` attribute identifies the name of the setting sequence.

```
<define name="config-config-item-map-entry-attlist" combine="interleave">
    <optional>
        <attribute name="config:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

2.4.5 Name Access of Sequences

The `<config:config-item-map-named>` element is a container element for sequences, where each setting in the sequence is identified by its name.

```
<define name="config-config-item-map-named">
    <element name="config:config-item-map-named">
        <ref name="config-config-item-map-named-attlist"/>
        <oneOrMore>
            <ref name="config-config-item-map-entry"/>
        </oneOrMore>
    </element>
</define>
```

Config Name

The `config:name` attribute identifies the name of the setting sequence.

```
<define name="config-config-item-map-named-attlist" combine="interleave">
    <attribute name="config:name">
        <ref name="string"/>
    </attribute>
</define>
```

2.4.6 Cursor Position Setting

A common view setting for editing applications is the position where the text cursor was while saving the document. For WYSIWYG applications, this usually will be a position within a paragraph only. For applications that provide an XML based view of the document, the cursor position could be also between arbitrary elements, or even within tags.

To represent a text cursor position within a document, a processing instruction with `PITarget opendocument` (see §2.6 of [XML1.0]) **should** be used. The name of the cursor position processing instruction, `cursor-position`, **shall** follow the `PITarget opendocument`. The processing instruction may have arbitrary application specific attributes, for instance to connect the cursor position with a certain view of the document, where the views themselves are specified as application specific settings. The syntax for these attributes **shall** be the same as for attributes within XML start tags.

Where a text cursor position is not sufficient to recreate a document view, applications may use arbitrary document specific settings in addition to the cursor position processing instruction. They may also use arbitrary document specific settings if the cursor position is not a text cursor position, but for instance a selection of drawing objects.

Example: cursor position processing instruction

```
<text:p>This is<?opendocument cursor-position view-id="view1"?> an
example.</text:p>
```

2.5 Scripts

A document may contain several scripts in different scripting languages. Each script is represented by a `<office:script>` element. All these script elements are contained in a single `<office:scripts>` element.

Scripts do not imply a scripting language or an object model. A script can for instance operate on the Document Object Model (DOM) composed from the XML representation of a document in OpenDocument format (see [DOM2]), or on an application specific API.

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

In addition to `<office:script>` elements, the `<office:scripts>` element may also contain an `<office:event-listeners>` element which contains the events assigned to the document itself. Examples for these are events called when the document is opened or closed. See section 12.4 for more information on the `<office:event-listeners>` element.

```
<define name="office-scripts">
    <optional>
        <element name="office:scripts">
            <zeroOrMore>
                <ref name="office-script"/>
            </zeroOrMore>
            <optional>
                <ref name="office-event-listeners"/>
            </optional>
        </element>
    </optional>
</define>
```

2.5.1 Script

The `<office:script>` element contains script language specific content. In most situations, the element contains the source code of the script, but it may also contain a compiled version of the script or a link to some external script code.

```
<define name="office-script">
    <element name="office:script">
        <ref name="office-script-attlist"/>
        <mixed>
            <ref name="anyElements"/>
        </mixed>
    </element>
</define>
```

Script Language

The attribute `script:language` specifies the language of the script by its name. Since script language names are application specific, the name should be preceded by a namespace prefix.

```
<define name="office-script-attlist">
    <attribute name="script:language">
        <ref name="string"/>
    </attribute>
</define>
```

2.6 Font Face Declarations

A document in OpenDocument format may contain font face declarations. A font face declaration provides information about the fonts used by the author of a document, so that these fonts or fonts that are very close to these fonts may be located on other systems. See section 14.6 for details.

```
<define name="office-font-face-decls">
    <optional>
        <element name="office:font-face-decls">
            <zeroOrMore>
                <ref name="style-font-face"/>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

2.7 Styles

The OpenDocument format supports the following types of **styles**:

- **Common styles**
  Most office applications support styles within their user interface. Within this specification, the XML representations of such styles are referred to as styles. When 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 an office application user 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. In other words, formatting properties that are immediately assigned to a specific object are represented by an automatic style. 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 are master pages. Master pages can be used in graphical applications. In this case, the additional content is any drawing shapes that are displayed as the background of the draw page. Master pages can also be used in text documents. 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 office application 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 any device, because this is common practice for documents created by word processors.

This type of style information differs from [CSS2] 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 document in OpenDocument format 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.

See section 14 for more information on styles.

2.7.1 Location of Styles

Common and automatic styles have the same 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:
- `<office:master-styles>`

```
<define name="office-styles">
    <optional>
        <element name="office:styles">
            <interleave>
                <ref name="styles"/>
                <zeroOrMore>
                    <ref name="style-default-style"/>
                </zeroOrMore>
                <optional>
                    <ref name="text-outline-style"/>
                </optional>
                <zeroOrMore>
                    <ref name="text-notes-configuration"/>
                </zeroOrMore>
                <optional>
                    <ref name="text-bibliography-configuration"/>
                </optional>
                <optional>
                    <ref name="text-linenumbering-configuration"/>
                </optional>
                <zeroOrMore>
                    <ref name="draw-gradient"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="svg-linearGradient"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="svg-radialGradient"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="draw-hatch"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="draw-fill-image"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="draw-marker"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="draw-stroke-dash"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="draw-opacity"/>
                </zeroOrMore>
                <zeroOrMore>
                    <ref name="style-presentation-page-layout"/>
                </zeroOrMore>
            </interleave>
        </element>
    </optional>
</define>
<define name="office-automatic-styles">
    <optional>
        <element name="office:automatic-styles">
            <interleave>
                <ref name="styles"/>
                <zeroOrMore>
                    <ref name="style-page-layout"/>
                </zeroOrMore>
            </interleave>
        </element>
    </optional>
</define>
<define name="office-master-styles">
    <optional>
        <element name="office:master-styles">
            <interleave>
                <zeroOrMore>
                    <ref name="style-master-page"/>
                </zeroOrMore>
                <optional>
                    <ref name="style-handout-master"/>
                </optional>
                <optional>
                    <ref name="draw-layer-set"/>
                </optional>
            </interleave>
        </element>
    </optional>
</define>

<define name="styles">
    <interleave>
        <zeroOrMore>
            <ref name="style-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="text-list-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-number-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-currency-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-percentage-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-date-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-time-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-boolean-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="number-text-style"/>
        </zeroOrMore>
    </interleave>
```

```
</define>
```

The following examples illustrate the different types of OpenDocument styles.

Example: OpenDocument styles

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

2.8 Page Styles and Layout

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

- Page Layouts
- Master Pages

A **page layout** 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 layout 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, a master page can be assigned 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, master pages can be assigned to drawing pages using a `style:parent-style-name` attribute.

Note: The OpenDocument 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. In the OpenDocument format, headers and footers are contained in page styles. With either approach, the content of headers and footers can be changed or omitted without affecting the document content.

Page layouts are described in section 14.3. Master pages are described in section 14.4.

3 Metadata Elements

The metadata elements borrow heavily upon the metadata standards developed by the Dublin Core Metadata Initiative (http://www.dublincore.org). Metadata elements drawn directly from the Dublin Core work use its namespace prefix (see section 1.3).

3.1 Pre-Defined Metadata Elements

There is a set of pre-defined metadata elements which should be processed and updated by the applications. Metadata elements may be omitted or occur multiple times. It is application-specific how to update multiple instances of the same elements.

3.1.1 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. This string **should** match the definition for user-agents in the HTTP protocol a specified in section 14.43 of [RFC2616]. The generator string **should** allow product versions to differ between all released versions of a user agent, for instance by including build ids or patch level information.

Conforming applications **may** use the generator string to work around bugs that exist or existed in certain applications, but **shall not** deliberately implement a different behavior depending on a certain generator string.

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 **shall not** export the original identifier belonging to the application that created the document.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:generator">
        <ref name="string"/>
    </element>
</define>
```

3.1.2 Title

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

```
<define name="office-meta-data" combine="choice">
    <element name="dc:title">
        <ref name="string"/>
    </element>
</define>
```

3.1.3 Description

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

```
<define name="office-meta-data" combine="choice">
    <element name="dc:description">
        <ref name="string"/>
    </element>
</define>
```

3.1.4 Subject

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

```
<define name="office-meta-data" combine="choice">
    <element name="dc:subject">
        <ref name="string"/>
    </element>
</define>
```

3.1.5 Keywords

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

```
<define name="office-meta-data" combine="choice">
    <element name="meta:keyword">
        <ref name="string"/>
    </element>
</define>
```

3.1.6 Initial Creator

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

```
<define name="office-meta-data" combine="choice">
    <element name="meta:initial-creator">
        <ref name="string"/>
    </element>
</define>
```

3.1.7 Creator

The `<dc:creator>` element specifies the name of the person who last modified the document. The name of this element was chosen for compatibility with the Dublin Core, but this definition of "creator" used here differs from Dublin Core, which defines creator as "An entity primarily responsible for making the content of the resource." In OpenDocument terminology, the last person to modify the document is primarily responsible for making the content of the document.

```
<define name="office-meta-data" combine="choice">
    <ref name="dc-creator"/>
</define>
<define name="dc-creator">
    <element name="dc:creator">
        <ref name="string"/>
    </element>
</define>
```

3.1.8 Printed By

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

```
<define name="office-meta-data" combine="choice">
    <element name="meta:printed-by">
        <ref name="string"/>
    </element>
```

```
</define>
```

3.1.9 Creation Date and Time

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

To conform with [xmlschema-2], the date and time format is YYYY-MM-DDThh:mm:ss.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:creation-date">
        <ref name="dateTime"/>
    </element>
</define>
```

3.1.10 Modification Date and Time

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

To conform with [xmlschema-2], the date and time format is YYYY-MM-DDThh:mm:ss.

The name of this element was chosen for compatibility with the Dublin Core.

```
<define name="office-meta-data" combine="choice">
    <ref name="dc-date"/>
</define>
<define name="dc-date">
    <element name="dc:date">
        <ref name="dateTime"/>
    </element>
</define>
```

3.1.11 Print Date and Time

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

To conform with [xmlschema-2], the date and time format is YYYY-MM-DDThh:mm:ss.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:print-date">
        <ref name="dateTime"/>
    </element>
</define>
```

3.1.12 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.

This element conforms to the XLink Specification. See [XLink].

The attributes that may be associated with the <meta:template> element are:

- Template location
- Template title
- Template modification date and time

Template Location

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

Template Title

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

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.

To conform with [xmlschema-2], the date and time format is YYYY-MM-DDThh:mm:ss.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:template">
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <value>simple</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onRequest">
                <value>onRequest</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:title">
                <ref name="string"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:date">
                <ref name="dateTime"/>
            </attribute>
        </optional>
    </element>
</define>
```

3.1.13 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.

The attributes that may be 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.

Reload Delay

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

To conform with the duration data type of [xmlschema-2], the format of the value of this attribute is `PnYnMnDTnHnMnS`. See §3.2.6 of [xmlschema-2] for more detailed information on this duration format.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:auto-reload">
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <value>simple</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show" a:defaultValue="replace">
                <value>replace</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onLoad">
                <value>onLoad</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:href">
                <ref name="anyURI"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:delay">
                <ref name="duration"/>
            </attribute>
        </optional>
    </element>
</define>
```

3.1.14 Hyperlink Behavior

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

The only attribute that may be 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.

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-behaviour>` element. 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`.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:hyperlink-behaviour">
        <optional>
            <attribute name="office:target-frame-name">
                <ref name="targetFrameName"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show">
                <choice>
                    <value>new</value>
                    <value>replace</value>
                </choice>
            </attribute>
        </optional>
    </element>
</define>
```

3.1.15 Language

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

The manner in which the language is represented is similar to the language tag described in [RFC3066]. It consists of a two or three 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.

```
<define name="office-meta-data" combine="choice">
    <element name="dc:language">
        <ref name="language"/>
    </element>
</define>
```

3.1.16 Editing Cycles

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

The value of this element is incremented every time the document is saved. The element contains the number of editing cycles as text.

```
<define name="office-meta-data" combine="choice">
```

```
    <element name="meta:editing-cycles">
        <ref name="nonNegativeInteger"/>
    </element>
</define>
```

3.1.17 Editing Duration

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

The duration is represented in the duration data type of [xmlschema-2], that is `PnYnMnDTnHnMnS`. See §3.2.6 of [xmlschema-2] for more detailed information on this duration format.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:editing-duration">
        <ref name="duration"/>
    </element>
</define>
```

3.1.18 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:image-count`
	`meta:object-count`
	`meta:ole-object-count`
	`meta:paragraph-count`
	`meta:word-count`
	`meta:character-count`
	`meta:row-count`
	`meta:frame-count`
	`meta:sentence-count`
	`meta:syllable-count`
	`meta:non-whitespace-character-count`

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

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

```
<define name="office-meta-data" combine="choice">
    <element name="meta:document-statistic">
        <optional>
```

```
            <attribute name="meta:page-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:table-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:draw-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:image-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:ole-object-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:object-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:paragraph-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:word-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:character-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="frame-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="sentence-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="syllable-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="non-whitespace-character-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:row-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:cell-count">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
    </element>
</define>
```

3.2 User-defined Metadata

The `<meta:user-defined>` element specifies any additional user-defined metadata for the document. 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.
- An optional `meta:value-type` attribute, which identifies the type of the metadata element. The allowed meta types are `float`, `date`, `time`, `boolean` and `string` (see also section 6.7.1).
- The value of the element, which is the metadata in the format described in section 6.7.1 as value of the `office:value` attributes for the various data types.

The default type for meta-data elements is string.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:user-defined">
        <attribute name="meta:name">
            <ref name="string"/>
        </attribute>
        <choice>
            <group>
                <attribute name="meta:value-type">
                    <value>float</value>
                </attribute>
                <ref name="double"/>
            </group>
            <group>
                <attribute name="meta:value-type">
                    <value>date</value>
                </attribute>
                <ref name="dateOrDateTime"/>
            </group>
            <group>
                <attribute name="meta:value-type">
                    <value>time</value>
                </attribute>
                <ref name="duration"/>
            </group>
            <group>
                <attribute name="meta:value-type">
                    <value>boolean</value>
                </attribute>
                <ref name="boolean"/>
            </group>
            <group>
                <attribute name="meta:value-type">
                    <value>string</value>
                </attribute>
                <ref name="string"/>
            </group>
            <text/>
        </choice>
    </element>
</define>
```

3.3 Custom Metadata

In addition to the pre-defined metadata elements, applications should also preserve any additional content found inside the `<office:meta>` element. As there is no semantics specified for such foreign content, applications need not process this information other than to preserve it when editing the document.

4 Text Content

4.1 Headings, Paragraphs and Basic Text Structure

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

The elements `<text:h>` and `<text:p>` represent headings and paragraphs, respectively, and are collectively referred to as **paragraph elements**. All text content in an OpenDocument file must be contained in either of these elements.

4.1.1 Headings

Headings define the chapter structure for a document. A chapter or subchapter begins with a heading and extends to the next heading at the same or higher level.

```
<define name="text-h">
    <element name="text:h">
        <ref name="heading-attrs"/>
        <ref name="paragraph-attrs"/>
        <optional>
            <ref name="text-number"/>
        </optional>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </element>
</define>
```

Heading Level

The `text:outline-level` attribute associated with the heading element determines the level of the heading, starting with 1. Headings without a level attribute are assumed to be at level 1.

```
<define name="heading-attrs" combine="interleave">
    <attribute name="text:outline-level">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

Heading Numbering

Header numbering can be changed by additional attributes, similar to those on list items (see section 4.3.2, below). The numbering of headers can be restarted by setting the `text:restart-numbering` attribute to `true`.

```
<define name="heading-attrs" combine="interleave">
    <optional>
        <attribute name="text:restart-numbering" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Start Value

The attribute `text:start-value` may be used to restart the numbering of headers of the current header's level, by setting a new value for the numbering.

```
<define name="heading-attrs" combine="interleave">
    <optional>
        <attribute name="text:start-value">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Suppress Header Numbering

It is sometimes desired to have a specific heading which should not be numbered. This corresponds to unnumbered list headers in lists (see sections 4.3). To facilitate this, an optional attribute `text:is-list-header` can be used. If `true`, the given header will not be numbered, even if an explicit list-style is given.

```
<define name="heading-attrs" combine="interleave">
    <optional>
        <attribute name="text:is-list-header" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Formatted Heading Number

If a heading has a numbering applied, the text of the formatted number can be included in a `<text:number>` element. This text can be used by applications that do not support numbering of headings, but it will be ignored by applications that support numbering.

```
<define name="text-number">
    <element name="text:number">
        <ref name="string"/>
    </element>
</define>
```

4.1.2 Paragraphs

Paragraphs are the basic unit of text.

```
<define name="text-p">
    <element name="text:p">
        <ref name="paragraph-attrs"/>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </element>
</define>
```

4.1.3 Common Paragraph Elements Attributes

The paragraph elements have `text:style-name`, `text:class-names` and `text:cond-style-name` attributes. These attributes must reference paragraph styles.

A `text:style-name` attribute references a paragraph style, while a `text:cond-style-name` attribute references a conditional-style, that is, a style that contains conditions and maps to other styles (see section 14.1.1). If a conditional style is applied to a paragraph, the `text:style-name` attribute contains the name of the style that was the result of the conditional style evaluation, while the conditional style name itself is the value of the `text:cond-style-name` attribute. 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.

A `text:class-names` attribute takes a whitespace separated list of paragraph style names. The referenced styles are applied in the order they are contained in the list. If both, `text:style-name` and `text:class-names` are present, the style referenced by the `text:style-name` attribute is as the first style in the list in `text:class-names`. If a conditional style is specified together with a `style:class-names` attribute, but without the `text:style-name` attribute, then the first style in the style list is used as the value of the missing `text:style-name` attribute.

Conforming applications should support the `text:class-names` attribute and also should preserve it while editing.

```
<define name="paragraph-attrs">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:class-names">
            <ref name="styleNameRefs"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:cond-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Example: Styles and conditional styles

```
<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>
```

A paragraph may have an ID. This ID can be used to reference the paragraph from other elements.

```
<define name="paragraph-attrs" combine="interleave">
    <optional>
        <ref name="text-id"/>
    </optional>
</define>
```

4.2 Page Sequences

A page sequence element `<text:page-sequence>` specifies a sequence of master pages that are instantiated in exactly the same order as they are referenced in the page sequence. If a text document contains a page sequence, it will consist of exactly as many pages as specified. Documents with page sequences do not have a main text flow consisting of headings and paragraphs as is the case for documents that do not contain a page sequence. Text content is included within text boxes for documents with page sequences. The only other content that is permitted are drawing objects.

Example: Page Sequence

```
<style:automatic-style>
    <style:page-layout name="pm1">
        <!-- portrait page -->
    </style:page-layout>
    <style:page-layout name="pm2">
        <!-- landscape page -->
    </style:page-layout>
</style:automatic-style>
...
<style:master-styles>
    <style:master-page name="portrait" style:page-layout-name="pm1"/>
    <style:master-page name="landscape" style:page-layout-name="pm2"/>
</style:master-styles>
...
<office:body>
    <text:page-sequence>
        <text:page text:master-page-name="portrait"/>
        <text:page text:master-page-name="portrait"/>
        <text:page text:master-page-name="landscape"/>
        <text:page text:master-page-name="landscape"/>
        <text:page text:master-page-name="portrait"/>
    </text:page-sequence>
    <draw:frame ...>
        <draw:text-box ...>
            <text:p>Example text.</text:p>
            ...
        </draw:text-box>
    </draw:frame>
</office:body>
```

```
<define name="text-page-sequence">
    <element name="text:page-sequence">
        <oneOrMore>
            <ref name="text-page"/>
        </oneOrMore>
    </element>
</define>
```

4.2.1 Page

The `<text:page>` element specifies a single page within a page sequence.

```
<define name="text-page">
    <element name="text:page">
        <ref name="text-page-attlist"/>
        <empty/>
    </element>
</define>
```

Master Page Name

The `text:master-page-name` attribute specifies the master page that is instantiated.

```
<define name="text-page-attlist">
    <attribute name="text:master-page-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

4.3 Lists

The OpenDocument format supports list structures, similar to those found in [HTML4]. A list is a paragraph-level element, which contains an optional list header, followed by a sequence of list items. The list header and each list item contains a sequence of paragraph or list elements. Lists can be nested.

Lists may be numbered. The numbering may be restarted with a specific numbering at each list item. Lists may also continue numbering from other lists, allowing the user to merge several lists into a single, discontinuous list. Note that whether the list numbering is displayed depends on a suitable list style being used.

In addition to this structural information, lists can have list styles associated with them, which contain the relevant layout information, such as

- the type of list item label, such as bullet or number,
- list item label width and distance,
- bullet character or image (if any),
- number format for the bullet numbering (if any),
- paragraph indent for list items.

4.3.1 List Block

A list is represented by the `<text:list>` element. It contains an optional list header, followed by any number of list items.

Every list has a *list level*, which is determined by the nesting of the `<text:list>` elements. 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 list is contained in a table cell or text box, the list level returns to 1, even though the table or textbox itself may be nested within another list.

The attributes that may be associated with the list element are:

- Style name
- Continue numbering

```
<define name="text-list">
    <element name="text:list">
        <ref name="text-list-attr"/>
        <optional>
            <ref name="text-list-header"/>
        </optional>
        <zeroOrMore>
            <ref name="text-list-item"/>
        </zeroOrMore>
    </element>
</define>
```

Style Name

The optional `text:style-name` attribute specifies the name of the list style that is applied to the list.

If this attribute is not included and therefore no list style is 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.

To determine which formatting properties are applied to a list, the list level and list style name are taken into account. See section 14.10 for more information on list formatting properties.

```
<define name="text-list-attr" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Continue Numbering

By default, the first list item in a list starts with the number specified in the list style. The continue numbering attribute can be used to continue the numbering from the preceding list.

This attribute can be used with the `<text: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.

```
<define name="text-list-attr" combine="interleave">
    <optional>
        <attribute name="text:continue-numbering">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

4.3.2 List Item

List items contain the textual content of a list. A `<text:list-item>` element can contain paragraphs, headings, lists or soft page breaks. A list item cannot contain tables.

```
<define name="text-list-item">
    <element name="text:list-item">
        <ref name="text-list-item-attr"/>
        <ref name="text-list-item-content"/>
    </element>
</define>
<define name="text-list-item-content">
```

```
    <optional>
        <ref name="text-number"/>
    </optional>
    <zeroOrMore>
        <choice>
            <ref name="text-p"/>
            <ref name="text-h"/>
            <ref name="text-list"/>
            <ref name="text-soft-page-break"/>
        </choice>
    </zeroOrMore>
</define>
```

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.

The only attribute that may be associated with the `<text:list-item>` element is:

- Start value

Start Value

The numbering of the current list can be restarted at a certain number. The `text:start-value` attribute is used to specify the number with which to restart the list.

This attribute can only be applied to items in a list with a numbering list style. It restarts the numbering of the list at the current item.

```
<define name="text-list-item-attr" combine="interleave">
    <optional>
        <attribute name="text:start-value">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Formatted Number

If a list item has a numbering applied, the text of the formatted number can be included in a `<text:number>` element. This text can be used by applications that do not support numbering, but it will be ignored by applications that support numbering. See also section 4.1.1.

Example: Lists and sublists

```
<text: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 a sub list.</text:p>
    <text:list>
        <text:list-item><text:p>This is a sub list item.</text:p>
        </text:list-item>
        <text:list-item><text:p>This is a sub list item.</text:p>
        </text:list-item>
        <text:list-item><text:p>This is a sub list item.</text:p>
        </text:list-item>
    </text:list>
    </text:list-item>
    <text:list-item>
    <text:p>This is the third list item</text:p>
    </text:list-item>
</text:list>
```

4.3.3 List Header

A list header is a special kind of list item. It 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.

```
<define name="text-list-header">
    <element name="text:list-header">
        <ref name="text-list-item-content"/>
    </element>
</define>
```

4.3.4 Numbered Paragraphs

In some instances, it is desirable to specify a list not as a structural element comprising of several list items, but to determine on a per-paragraph level whether the paragraph is numbered, and at which level. To facilitate this, the `<text:numbered-paragraph>` element allows the numbering of an individual paragraph, as if it was part of a list at a specified level.

Numbered paragraphs may use the same continuous numbering properties that list items use, and thus form an equivalent, alternative way of specifying lists. A list in `<text:list>` representation could be converted into a list in `<text:numbered-paragraph>` representation and vice versa.

```
<define name="text-numbered-paragraph">
    <element name="text:numbered-paragraph">
        <ref name="text-numbered-paragraph-attr"/>
        <optional>
            <ref name="text-number"/>
        </optional>
        <choice>
            <ref name="text-p"/>
            <ref name="text-h"/>
        </choice>
    </element>
</define>
```

A numbered paragraph can be assigned a list level. A numbered paragraph is equivalent to a list nested to the given level, containing one list item with one paragraph. If no level is given, the numbered paragraph is interpreted as being on level 1.

```
<define name="text-numbered-paragraph-attr" combine="interleave">
    <optional>
        <attribute name="text:level" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

As a numbered paragraph combines the functionality of a (possibly nested) list with a single list item, it can also use the attributes of those elements.

```
<define name="text-numbered-paragraph-attr" combine="interleave">
    <ref name="text-list-attr"/>
```

```
</define>
<define name="text-numbered-paragraph-attr" combine="interleave">
    <ref name="text-list-item-attr"/>
</define>
```

The text of a formatted number can be included in a `<text:number>` element. This text can be used by applications that do not support numbering, but it will be ignored by applications that support numbering. See also section 4.1.1.

4.4 Text Sections

A text section is a named region of paragraph-level text content. Sections start and end on paragraph boundaries and can contain any number of paragraphs.

Sections have two uses in the OpenDocument format: They can be used to assign certain formatting properties to a region of text. They can also be used to group text that is automatically acquired from some external data source.

In addition to 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.

Sections can have settings for text columns, background color or pattern, and notes configuration. These settings form the section style, which is represented in a `<style:style>` element. See section 14.8.3 for details.

The formatting properties for sections are explained in section 15.7.

Sections support two ways of linking to external content. 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

Linking information for external content is contained in the section element's first child. A section that links to external content contains the full representation of the data source, so that processors need to understand the linking information only if they wish to update the contents of the section.

```
<define name="text-section">
    <element name="text:section">
        <ref name="text-section-attr"/>
        <choice>
            <ref name="text-section-source"/>
            <ref name="text-section-source-dde"/>
            <empty/>
        </choice>
        <zeroOrMore>
            <ref name="text-content"/>
        </zeroOrMore>
    </element>
</define>
```

Note: List items may not contain sections. Thus, lists may only be wholly contained within section elements. If it is desired to achieve the effect of overlapping lists and sections, or of sections contained within lists, the lists must be split into several lists, each of which would then be wholly contained within a section. When splitting the list, suitable attributes for continuous numbering should be set such that display and behavior are the same as with the original list not interrupted by sections.

4.4.1 Section Attributes

Text indices, described in chapter 7, may be considered a special kind of text section, as they share the same general structure as well as certain attributes. These are combined in the following definition:

```
<define name="text-section-attr" combine="interleave">
    <ref name="sectionAttr"/>
</define>
```

The remaining attributes in this section are specific to the `<text:section>` element.

Section Style

The `text:style-name` attribute refers to a section style.

```
<define name="sectionAttr" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Section Name

Every section must have a name that uniquely identifies the section. The `text:name` attribute contains the name of the section.

```
<define name="sectionAttr" combine="interleave">
    <attribute name="text:name">
        <ref name="string"/>
    </attribute>
</define>
```

Protected Sections

A section can be protected, which means that a user can not edit the section. The `text:protected` attribute indicates whether or not a section is protected. The user interface must enforce the protection attribute if it is enabled.

```
<define name="sectionAttr" combine="interleave">
    <optional>
        <attribute name="text:protected">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

A user can use the user interface to reset the protection flag, unless the section is further protected by a password. In this case, the user must know the password in order to reset the protection flag. The `text:protection-key` attribute specifies the password that protects the section. To avoid saving the password directly into the XML file, only a hash value of the password is stored.

```
<define name="sectionAttr" combine="interleave">
    <optional>
        <attribute name="text:protection-key">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

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 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.

The `text:condition` attribute specifies the condition under which the section is hidden. The condition is encoded as a string. If the value of `text:display` is `condition`, the `text:condition` attribute must be present.

```
<define name="text-section-attr" combine="interleave">
    <choice>
        <attribute name="text:display">
            <choice>
                <value>true</value>
                <value>none</value>
            </choice>
        </attribute>
        <group>
            <attribute name="text:display">
                <value>condition</value>
            </attribute>
            <attribute name="text:condition">
                <ref name="string"/>
            </attribute>
        </group>
        <empty/>
    </choice>
</define>
```

4.4.2 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.

```
<define name="text-section-source">
    <element name="text:section-source">
        <ref name="text-section-source-attr"/>
    </element>
</define>
```

The attributes that may be 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. The name of the target section is identified by the local part of the URL, following the hash mark. The `xlink:href` attribute is implied because `<text:section-source>` elements may also link to internal sections.

```
<define name="text-section-source-attr" combine="interleave">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <value>simple</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show" a:defaultValue="embed">
                <value>embed</value>
            </attribute>
        </optional>
    </optional>
</define>
```

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. If the attribute is not present, the link targets the entire document.

```
<define name="text-section-source-attr" combine="interleave">
    <optional>
        <attribute name="text:section-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Filter Name

The `text:filter-name` attribute specifies which filter type was used to import the link target. The value of this attribute is implementation dependent.

```
<define name="text-section-source-attr" combine="interleave">
    <optional>
        <attribute name="text:filter-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

4.4.3 DDE Source

If sections are linked via DDE, their linking information is represented by `<office:dde-source>` elements. It contains attributes that specify the application, topic and item of the DDE connection. Note that because the section contains the XML rendition of the DDE link's content, this information only needs to be processed if updated data from the DDE link are desired.

See section 12.6 for the use of DDE connections.

```
<define name="text-section-source-dde">
    <ref name="office-dde-source"/>
</define>
```

4.5 Page-bound graphical content

Within text documents, images, embedded objects and other drawing objects appear at the level of a paragraph if they are anchored to a page rather than to a paragraph or a character position within a paragraph. See section 9.2 for details on drawing objects, and section 9.2.16 for their anchoring.

4.6 Change Tracking

This section describes how changes in text documents can be represented.

4.6.1 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. If the `<text:tracked-changes>` element is absent, change tracking is not enabled.

```
<define name="text-tracked-changes">
    <optional>
        <element name="text:tracked-changes">
            <ref name="text-tracked-changes-attr"/>
            <zeroOrMore>
                <ref name="text-changed-region"/>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

Track Changes

This attribute determines whether or not user agents should track and record changes for this document.

```
<define name="text-tracked-changes-attr" combine="interleave">
    <optional>
        <attribute name="text:track-changes" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

4.6.2 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.

```
<define name="text-changed-region">
    <element name="text:changed-region">
        <ref name="text-changed-region-attr"/>
        <ref name="text-changed-region-content"/>
    </element>
</define>
```

Change ID

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.

```
<define name="text-changed-region-attr" combine="interleave">
    <attribute name="text:id">
        <ref name="ID"/>
    </attribute>
</define>
```

4.6.3 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.

```
<define name="text-changed-region-content" combine="choice">
    <element name="text:insertion">
        <ref name="office-change-info"/>
    </element>
</define>
```

Example: Insertion of text

```
<text:tracked-changes>
    <text:changed-region text:id="c001">
        <text:insertion>
            <office:change-info>
                <dc:creator>Michael Brauer</dc:creator>
                <dc:date>1999-05-18T12:56:04</dc:date>
            </office:change-info>
        </text:insertion>
    </text:changed-region>
</text:tracked-changes>

<text:p>
    This is the original text<text:change-start text:change-id="c001"/>,
    but this has been added<text:change-end text:change-id="c001"/>.
</text:p>
```

4.6.4 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.

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.

```
<define name="text-changed-region-content" combine="choice">
    <element name="text:deletion">
        <ref name="office-change-info"/>
        <zeroOrMore>
            <ref name="text-content"/>
        </zeroOrMore>
    </element>
</define>
```

Example: Deletion of text

```
<text:tracked-changes>
    <text:changed-region text:id="c002">
        <text:deletion>
            <office:change-info>
                <dc:creator>Michael Brauer</dc:creator>
                <dc:date>1999-05-18T12:56:04</dc:date>
            </office:change-info>
            <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.`

Note that the deleted text, like all text in the OpenDocument format, is contained in a paragraph element. To reconstruct the original text, this paragraph is merged with its surrounding. In other words, a deletion consisting of only a single word would be represented as a paragraph containing the word.

To reconstruct the text before the deletion took place, do:

- If the change mark is inside a paragraph, insert the text content of the <text:deletion> element as if the beginning <text:p> and final </text:p> tags were missing.
- If the change mark is inside a header, proceed as above, except adapt the end tags to match their new counterparts.
- Otherwise, simply copy the text content of the <text:deletion> element in place of the change mark.

Example: Given the following change:

```
<text:changed-region text:id="example">
    <text:deletion>
        <office:change-info>...</office:change-info>
        <text:p>Hello</text:p>
        <text:p>World!</text:p>
    </text:deletion>
</text:changed-region>
```

The first (and most common) case occurs if a change mark is inside a regular paragraph:

```
<text:p>abc<text:change text:id="example/>def</text:p>
```

To reconstruct the original text, the two <text:p> elements are copied to replace the change mark, except the beginning and ending tags are missing:

```
<text:p>abcHello</text:p>
<text:p>World!def</text:p>
```

If the change mark occurred inside a header, the same procedure is followed, except the copied tags are adapted to make sure we still have well-formed XML.

```
<text:h>abc<text:change text:id="example/>def</text:h>
```

becomes:

```
<text:h>abcHello</text:h>
<text:h>World!def</text:h>
```

The third case occurs when a change occurs outside of a paragraph. In this case, the deleted text is simply copied verbatim.

```
<text:p>abcdef</text:p>
<text:change text:id="example/>
<text:p>ghijkl</text:p>
```

This becomes:

```
<text:p>abcdef</text:p>
<text:p>Hello</text:p>
<text:p>World!</text:p>
<text:p>ghijkl</text:p>
```

If, in the first two cases, the deletion contains complete paragraphs, then additional empty paragraphs must be put into the <text:deletion> element to achieve the desired result.

The change that took place from

```
<text:p>abc</text:p>
<text:h>Hello</text:h>
<text:h>World!</text:h>
<text:p>def</text:p>
```

to

```
<text:p>abc<text:change text:id="example/>def</text:p>
```

would be represented as:

```
<text:changed-region text:id="example">
   <text:deletion>
       <office:change-info>...</office:change-info>
       <text:p/>
       <text:h>Hello</text:h>
       <text:h>World!</text:h>
       <text:p/>
   </text:deletion>
</text:changed-region>
```

4.6.5 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.

```
<define name="text-changed-region-content" combine="choice">
    <element name="text:format-change">
        <ref name="office-change-info"/>
    </element>
</define>
```

Note: A format change element does not contain the actual changes that took place.

4.6.6 Change Info

The change info element contains meta information who made the change and when. It is also used for spreadsheet documents, and thus described in a section 12.3 (Change Tracking Metadata).

4.6.7 Change Marks

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.

All three elements have an attribute that specifies the ID of the region to which they belong.

```
<define name="change-marks">
    <choice>
        <element name="text:change">
            <ref name="change-mark-attr"/>
        </element>
        <element name="text:change-start">
            <ref name="change-mark-attr"/>
        </element>
        <element name="text:change-end">
            <ref name="change-mark-attr"/>
        </element>
    </choice>
</define>
<define name="change-mark-attr">
    <attribute name="text:change-id">
        <ref name="IDREF"/>
    </attribute>
</define>
```

4.7 Soft Page Break

The `<text:soft-page-break>` element represents a soft page break.

See section 2.3.1:Use Soft Page BreaksUse Soft Page Breaks for details regarding soft page breaks.

```
<define name="text-soft-page-break">
    <element name="text:soft-page-break">
        <empty/>
    </element>
</define>
```

4.8 Text Declarations

Several text elements need per-document declarations before they can be used. For example, variable fields require that the variables used are being declared at the beginning of the document. These declarations are collected at the beginning of a text document. All such declarations are optional. The detailed description for each declaration can be found in the appropriate chapter.

The supported text declarations are:

- variable declarations – These declarations are used for variable fields. (cf. section 6.3.1).

- user field declarations – These declarations are used for user-defined fields (cf. section 6.3.5).
- sequence declarations – These declarations are used for sequence fields (cf. section 6.3.8).
- DDE connections – These declarations are used for DDE fields and DDE sections (cf. sections 6.6.9 and 4.4.3, respectively).
- auto mark file – This declaration is used for generation of alphabetical indices (cf. section 7.8.2).

```
<define name="text-decls">
    <optional>
        <element name="text:variable-decls">
            <zeroOrMore>
                <ref name="text-variable-decl"/>
            </zeroOrMore>
        </element>
    </optional>
    <optional>
        <element name="text:sequence-decls">
            <zeroOrMore>
                <ref name="text-sequence-decl"/>
            </zeroOrMore>
        </element>
    </optional>
    <optional>
        <element name="text:user-field-decls">
            <zeroOrMore>
                <ref name="text-user-field-decl"/>
            </zeroOrMore>
        </element>
    </optional>
    <optional>
        <element name="text:dde-connection-decls">
            <zeroOrMore>
                <ref name="text-dde-connection-decl"/>
            </zeroOrMore>
        </element>
    </optional>
    <optional>
        <ref name="text-alphabetical-index-auto-mark-file"/>
    </optional>
</define>
```

5 Paragraph Elements Content

5.1 Basic Text Content

Paragraph element's children make up the text content of any document. All text contained in a paragraph element or their children is text content, with few exceptions detailed later. This should significantly ease transformations into other formats, since transformations may ignore any child elements of paragraph elements and only process their text content, and still obtain a faithful representation of text content.

Text content elements that do not contain in-line text children are:

- (foot- and end-)notes (see section 5.3)

  Foot- and endnotes contain text content, but are typically displayed outside the main text content, e.g., at the end of a page or document.

- rubies (see section 5.4)

  Ruby texts are usually displayed above or below the main text.

- annotations (see section 5.5)

  Annotations are typically not displayed.

```
<define name="paragraph-content" combine="choice">
    <text/>
</define>
```

5.1.1 White-space Characters

If the paragraph element or any of its child elements contains white-space characters, they are collapsed. Leading white-space characters at the paragraph start as well as trailing white-space characters at the paragraph end are ignored. In detail, the following conversions take place:

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 characters at the start or end of the paragraph are ignored, regardless whether they are contained in the paragraph element itself, or in a child element in which white-space characters are collapsed as described above.

These white-space processing rules shall enable authors to use white-space characters to improve the readability of the XML source of an OpenDocument document in the same way as they can use them in HTML.

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], white-space processing of a paragraph of text can be enabled by attaching an `fo:white-space="collapse"` attribute to the `<fo:block>` element that corresponds to the paragraph element.

, in other words they are processed in the same way that [HTML4] processes them.

Space Character

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).

This element uses an optional attribute called `text:c` to specify the number of SPACE characters that the element represents. A missing `text:c` attribute is interpreted as meaning a single SPACE character.

This element is required to represent the second and all following SPACE characters in a sequence of SPACE characters. It is not 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:s">
        <optional>
            <attribute name="text:c">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
    </element>
</define>
```

Tab Character

The `<text:tab>` element represents the [UNICODE] tab character HORIZONTAL TABULATION (0x0009) in a heading or paragraph. A `<text:tab>` element reserves space from the current position up to the next tab-stop, as defined in the paragraph's style information.

```
<define name="paragraph-content" combine="choice">
    <element name="text:tab">
        <ref name="text-tab-attr"/>
    </element>
</define>
```

To determine which tab-stop a tab character will advance to requires layout information. To make it easier for non-layout oriented processors to determine this information, applications may generate a text:tab-ref attribute as a hint that associates a tab character with a tab-stop in the current paragraph style. It contains the number of the tab-stop that the tab character refers to. The position 0 has a special meaning and signifies the start margin of the paragraph.

```
<define name="text-tab-attr">
    <optional>
        <attribute name="text:tab-ref">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Note: The text:tab-ref attribute is only a hint to help non-layout oriented processors to determine the tab/tab-stop association. Layout oriented processors should determine the tab positions solely based on the style information.

Line Breaks

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

```
<define name="paragraph-content" combine="choice">
    <element name="text:line-break">
        <empty/>
    </element>
</define>
```

Soft Page Break

The `<text:soft-page-break>` element represents a soft page break within a heading or paragraph.

See section 2.3.1:Use Soft Page BreaksUse Soft Page Breaks for details regarding soft page breaks.

```
<define name="paragraph-content" combine="choice">
    <ref name="text-soft-page-break"/>
</define>
```

5.1.2 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

5.1.3 Attributed Text

The `<text:span>` element represents portions of text that are attributed using a certain text style or class. The content of this element is the text that uses the text style.

The name of the a text style or text class is the value of a `text:style-name` or `text:class-names` attributes, respectively, attached to the `<text:span>` element. These attributes must refer to text styles or classes.

A `text:style-name` attribute references a single text style. A text:class-names attribute takes a whitespace separated list of text style names. The referenced text styles are applied in the order they are contained in the list. If both, `text:style-name` and `text:class-names` are present, the style referenced by the `text:style-name` attribute is treated as the first style in the list in `text:class-names`. Conforming application should support the `text:class-names` attribute and also should preserve it while editing.

`<text:span>` elements can be nested.

White-space characters contained in this element are collapsed.

```
<define name="paragraph-content" combine="choice">
    <element name="text:span">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="text:class-names">
                <ref name="styleNameRefs"/>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </element>
</define>
```

Example: Text style in OpenDocument documents:

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

5.1.4 Hyperlinks

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

This element also contains an event table element, `<office:event-listeners>`, which contains the events assigned to the hyperlink. See section 12.4 for more information on the event table element.

```
<define name="paragraph-content" combine="choice">
    <element name="text:a">
        <ref name="text-a-attlist"/>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </element>
</define>
```

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

- Name
- Link location
- Target frame
- Text styles

Name

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

```
<define name="text-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Title

The `office:title` attribute specifies a short accessible description for hint text.

See appendix E for guidelines how to use this attribute.

```
<define name="text-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:title">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Link Location

The `xlink:href` attribute specifies the URL for the target location of the link.

```
<define name="text-a-attlist" combine="interleave">
    <attribute name="xlink:href">
        <ref name="anyURI"/>
    </attribute>
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <value>simple</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate" a:defaultValue="onRequest">
            <value>onRequest</value>
        </attribute>
    </optional>
</define>
```

Target Frame

The office:target-frame-name attribute specifies the target frame of the link. 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. 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. See [XLink].

```
<define name="text-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:target-frame-name">
            <ref name="targetFrameName"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:show">
            <choice>
                <value>new</value>
                <value>replace</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Text Styles

Every hyperlink has two text styles as follows:

- If the link location of the hyperlink was not visited, the text style specifies by the text:style-name attribute is applied to the text of the hyperlink.
- If the link location of the hyperlink was already visited, the text style specified by the text:visited-style-name attribute is applied to the text of the hyperlink

```
<define name="text-a-attlist" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:visited-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

5.2 Bookmarks and References

5.2.1 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

For every `<text:bookmark-start>` element, there must be a `<text:bookmark-end>` element in the same text flow using the same `text:name` attribute, and vice versa. The `<text:bookmark-start>` element must precede the `<text:bookmark-end>` element.

```
<define name="paragraph-content" combine="choice">
    <choice>
        <element name="text:bookmark">
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </element>
        <element name="text:bookmark-start">
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </element>
        <element name="text:bookmark-end">
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </element>
    </choice>
</define>
```

Example: Bookmarks

```
<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>
```

5.2.2 References

The 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.

Point References

The `<text:reference-mark>` element represents a point reference.

```
<define name="paragraph-content" combine="choice">
    <element name="text:reference-mark">
        <attribute name="text:name">
            <ref name="string"/>
        </attribute>
    </element>
</define>
```

Range References

The `<text:reference-mark-start>` and `<text:reference-mark-end>` elements represent a range reference.

```
<define name="paragraph-content" combine="choice">
    <choice>
        <element name="text:reference-mark-start">
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </element>
        <element name="text:reference-mark-end">
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </element>
    </choice>
</define>
```

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

```
    <text:reference-mark-end text: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.

5.3 Notes

Notes consist of a `<text:note>` element which occurs in the text stream at the position to which the note is anchored. How notes are numbered and rendered is determined by `<text:notes-configuration>` element, which occurs inside the `<office:styles>` section.

5.3.1 Note Element

The note element represents text notes which are attached to a certain text position. A common implementation of this concept are the footnotes and endnotes found in most word processors. A note contains a note citation element and a note body elements, which contains the note's content.

In OpenDocument documents, notes are represented in a similar fashion to footnotes in [XSL]. In XSL, the first child of the note element contains the citation in the form of an `<fo:inline>` element. The OpenDocument schema uses the same structure but introduces a `<text:note-citation>` element. The second child contains the note body, just as in XSL.

Additionally, OpenDocument features `<text:notes-configuration>` elements. To achieve a similar effect to the note configuration in XSL, every note citation element must be formatted appropriately.

```
<define name="paragraph-content" combine="choice">
    <element name="text:note">
        <ref name="text-note-class"/>
        <optional>
            <attribute name="text:id">
                <ref name="string"/>
            </attribute>
        </optional>
        <element name="text:note-citation">
            <optional>
                <attribute name="text:label">
                    <ref name="string"/>
                </attribute>
            </optional>
            <text/>
        </element>
        <element name="text:note-body">
            <zeroOrMore>
                <ref name="text-content"/>
            </zeroOrMore>
        </element>
    </element>
</define>
```

Note Class

Each note belongs to a class which determines how the note is expected to be rendered. Currently, two note classes are supported: Footnotes and endnotes.

```
<define name="text-note-class">
    <attribute name="text:note-class">
        <choice>
            <value>footnote</value>
            <value>endnote</value>
        </choice>
    </attribute>
</define>
```

Footnote Reference ID

The footnote reference ID is used by references to footnotes to identify the footnote that is referenced.

Note Citation Element

The `<text:note-citation>` element contains the formatted note citation element, either as a formatted number or a string.

Note Label

Note citation elements can be labeled or numbered. If they are numbered, the number is chosen and formatted automatically according to the notes configuration element. If they are labeled, the user must supply a label for every note he/she inserts into the document. This label is stored in the `text:label` attribute of the `<text:note-citation>` element.

Note Body

The `<text:note-body>` element contains the actual content of the footnote. It does not have any attributes.

The schema allows for the inclusion of notes into the note body. While this may be reasonable for some future note types, it is not reasonable for footnotes and endnotes. Conforming applications may or may not support such nested notes.

Footnote example

```
<text:p>
    This paragraph contains a footnote
    <text:note text:note-class="footnote" text:id="ftn001">
        <text:note-citation>1</text:note-citation>
        <text:note-body>
            <text:p>
                This footnote has a generated sequence number
            </text:p>
        </text:note-body>
    </text:note>
    .
</text:p>
<text:p>
    This paragraph contains a footnote
    <text:note text:note-class="footnote" text:id="ftn002">
```

```
        <text:note-citation text:label="*">*</text:note-citation>
        <text:note-body>
            <text:p>
                 This footnote has a fixed citation
            </text:p>
        </text:note-body>
    </text:note>
    , too
</text:p>
```

5.4 Ruby

A ruby is additional text that is displayed above or below some base text. The purpose of ruby is to annotate the base text or provide information about its pronunciation.

There are two elements that can be contained in the `<text:ruby>` element:

- Ruby base
- Ruby text

The `<text:ruby-base>` element contains the text that is to be annotated. It contains any paragraph element content, like text spans. The element's `text:style-name` attribute references a ruby style that specifies further formatting attributes of the ruby. See section 14.8.4 for details.

The `<text:ruby-text >` element contains the annotation text. It may contain only plain text. The element's `text:style-name` attribute references a text style that specifies further formatting attributes used for the text.

```
<define name="paragraph-content" combine="choice">
    <element name="text:ruby">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <element name="text:ruby-base">
            <ref name="paragraph-content"/>
        </element>
        <element name="text:ruby-text">
            <optional>
                <attribute name="text:style-name">
                    <ref name="styleNameRef"/>
                </attribute>
            </optional>
            <text/>
        </element>
    </element>
</define>
```

5.5 Text Annotation

The OpenDocument format allows annotation to appear within a paragraph element. See section 12.1 for details on annotations.

```
<define name="paragraph-content" combine="choice">
    <ref name="office-annotation"/>
</define>
```

5.6 Index Marks

Index marks are used to mark text areas for inclusion into text indices. They are similar in structure to bookmarks and references. They are discussed in detail section 7.1, together with text indices.

5.7 Change Tracking and Change Marks

Paragraphs may also contain change tracking marks. These have already been explained in the chapter on change tracking (section 4.6), and are referenced here for completeness.

```
<define name="paragraph-content" combine="choice">
    <ref name="change-marks"/>
</define>
```

5.8 Inline graphics and text-boxes

Within text documents, images, embedded objects and other drawing objects may be anchored to a paragraph, to a character, or as a character. If they are anchored to a paragraph, they appear within a paragraph at an arbitrary position. If they are anchored to or as a character, they appear within a paragraph at exactly the character position they are anchored to or as. See section 9.2 for details on drawing objects, and section 9.2.16 for their anchoring.

```
<define name="paragraph-content" combine="choice">
    <choice>
        <ref name="shape"/>
        <ref name="draw-a"/>
    </choice>
</define>
```

6 Text Fields

OpenDocument text documents or OpenDocument 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 an image and its associated number range field are moved within a document, the fields are automatically updated to reflect the new order.

This section describes how fields are represented in the OpenDocument file format.

6.1 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 would be displayed or printed. 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 6.7.1. 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 6.7.2 for more information on this attribute.
- Formatting style
  Several field types, particularly those representing number, date, or time data, contain a formatting style. In the OpenDocument format, this formatting style is represented by a

style:data-style-name attribute. Since the user can change the presentation style for fields, applications must be able to recompute a new representation of the field content at any time. See section 6.7.7 for more information on this attribute.

6.2 Document Fields

OpenDocument 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 re-evaluated, 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. In the same way, 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
- Page number fields
- Sender and author fields
- Chapter fields
- File name fields
- Document template fields

6.2.1 Date Fields

Date fields display the current date. The date can be adjusted to display a date other than the current date. For example, the date can be changed on a document that was edited late at night so that it displays the date of the following day or several days later.

This element contains the presentation of the date field value, depending on the data style specified. The default date is the current date. The value of this element can be preserved using the text:fixed attribute described in section 6.7.2.

```
<define name="paragraph-content" combine="choice">
    <element name="text:date">
        <ref name="text-date-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated with the <text:date> element are:

- Date value
- Date adjustment
- Fixed (see section 6.7.2)
- Formatting style (see section 6.7.7). The formatting style must be a date data style, see section 14.7 for more information.

```
<define name="text-date-attlist" combine="interleave">
```

```
    <interleave>
        <ref name="common-field-fixed-attlist"/>
        <ref name="common-field-data-style-name-attlist"/>
    </interleave>
</define>
```

Date Value

The `text:date-value` attribute specifies a particular date value. For example, if the date field is marked fixed, this attribute can be used to specify the date on which the field was marked as fixed. This attribute can also be used to specify a future date. Some applications support date and time in addition to date-only values.

The date value should conform with the date formats described in §3.2.7 and §3.2.9 of [xmlschema-2]. If no value is specified, the current date is assumed, even if the field is marked fixed.

```
<define name="text-date-attlist" combine="interleave">
    <optional>
        <attribute name="text:date-value">
            <ref name="dateOrDateTime"/>
        </attribute>
    </optional>
</define>
```

Date Adjustment

The value of a date field can be adjusted by a certain time period, which is specified using the `text:date-adjust` attribute. If the time period is negative, it gets subtracted from the value of the date field, yielding a date before the current date.

The value of this attribute must conform to the time period format described in §3.2.6 of [xmlschema-2]. The value can be preceded by an optional minus sign to indicate a negative time duration.

```
<define name="text-date-attlist" combine="interleave">
    <optional>
        <attribute name="text:date-adjust">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

6.2.2 Time Fields

Time fields display the current time. They are very similar to the date fields described in section 6.2.1, supporting the same attributes except that for time fields, they are called `text:time-value` and `text:time-adjust` attributes.

This element contains the presentation of the time field value, depending on the data style specified. The default time is the current time. The value of this element can be preserved using the `text:fixed` attribute described in section 6.7.2.

```
<define name="paragraph-content" combine="choice">
    <element name="text:time">
        <ref name="text-time-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated with the `<text:time>` element are:

- Time value
- Time adjustment
- Fixed (see section 6.7.2)
- Formatting style (see section 6.7.7). The formatting style must be a time data style, see section 14.7 for more information.

```
<define name="text-time-attlist" combine="interleave">
    <interleave>
        <ref name="common-field-fixed-attlist"/>
        <ref name="common-field-data-style-name-attlist"/>
    </interleave>
</define>
```

Time Value

The `text:time-value` attribute records the time at which the document was last edited.

Some applications support date and time in addition to date-only values.

The value of this attribute must conform with either the “dateTime” or “time” data types described in §3.2.7 and §3.2.8 of [xmlschema-2]. If no value is specified, the current time is assumed, even if the field is marked `fixed`.

```
<define name="text-time-attlist" combine="interleave">
    <optional>
        <attribute name="text:time-value">
            <ref name="timeOrDateTime"/>
        </attribute>
    </optional>
</define>
```

Time Adjustment

The value of a time field can be adjusted by a certain time period, which is specified using the `text:time-adjust` attribute.

The value of this attribute must conform to the time period format described in §3.2.6 of [xmlschema-2]. 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.

```
<define name="text-time-attlist" combine="interleave">
    <optional>
        <attribute name="text:time-adjust">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

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.

6.2.3 Page Number Fields

Page number fields display the current page number. These fields are particularly useful in headers and footers. E.g., if a page number field is inserted into a footer, the current page number is displayed on every page on which the footer appears.

The attributes that may be associated with the `<text:page-number>` element are:

- Page adjustment
- Display previous or following page numbers
- Fixed (see section 6.7.2)
- Formatting style (see section 6.7.8)
  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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:page-number">
        <ref name="text-page-number-attlist"/>
        <text/>
    </element>
</define>
<define name="text-page-number-attlist" combine="interleave">
    <interleave>
        <ref name="common-field-num-format-attlist"/>
        <ref name="common-field-fixed-attlist"/>
    </interleave>
</define>
```

Note: To display the total number of pages in a document, use the `<text:page-count/>` field described in section 6.4.17.

Page Adjustment

The value of a page number field can be adjusted by a specified number, allowing the display of page numbers of following or preceding pages. The adjustment amount is specified using the `text:page-adjust` attribute. When this attribute is used, 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.

```
<define name="text-page-number-attlist" combine="interleave">
    <optional>
        <attribute name="text:page-adjust">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Display Previous or Following Page Numbers

The `text:select-page` attribute is used to display the number of the previous or the following page rather than the number of the current page.

```
<define name="text-page-number-attlist" combine="interleave">
    <optional>
        <attribute name="text:select-page">
            <choice>
                <value>previous</value>
                <value>current</value>
                <value>next</value>
            </choice>
        </attribute>
    </optional>
</define>
```

> **Note:** To display the current page number on all pages except the first or last page, 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"
                    style:num-format="1"/>
```

6.2.4 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:page-continuation">
        <ref name="text-page-continuation-attlist"/>
        <text/>
    </element>
</define>
```

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.

```
<define name="text-page-continuation-attlist" combine="interleave">
    <attribute name="text:select-page">
        <choice>
            <value>previous</value>
            <value>next</value>
        </choice>
    </attribute>
</define>
```

String Value

This attribute specifies the continuation text to display. If this attribute is omitted, the element content is used.

```
<define name="text-page-continuation-attlist" combine="interleave">
    <optional>
        <attribute name="text:string-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

6.2.5 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 OpenDocument 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. (cf. section 6.7.2) 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-firstname">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Last Name

This element represents the last name of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-lastname">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Initials

This element represents the initials of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-initials">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Title

This element represents the title of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-title">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Position

This element represents the position of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-position">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Email Address

This element represents the email address of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-email">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Private Telephone Number

This element represents the private telephone number of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-phone-private">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Fax Number

This element represents the facsimile number of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-fax">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Company Name

This element represents the name of the company that employs the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-company">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Office Telephone Number

This element represents the office telephone number of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-phone-work">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Street

This element represents the street name of the address of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-street">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

City

This element represents the city name of the address of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-city">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Postal Code

This element represents the postal code of the address of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-postal-code">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Country

This element represents the country of the address of the sender.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-country">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

State or Province

This element represents the state or province of the address of the sender, if applicable.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sender-state-or-province">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.2.6 Author Fields

There are two elements available to display the author of a document. One element displays the full name of the author and the other element displays the initials of the author.

The value of 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:author-name">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

Initials of the Author

This element represents the initials of the author.

```
<define name="paragraph-content" combine="choice">
    <element name="text:author-initials">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.2.7 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:chapter">
        <ref name="text-chapter-attlist"/>
        <text/>
    </element>
```

```
</define>
```

The attributes that may be associated with the `<text:chapter>` element are:

- Display
- Outline level

Display

The `text:display` attribute specifies the information that the chapter field should display.

```
<define name="text-chapter-attlist" combine="interleave">
    <attribute name="text:display">
        <choice>
            <value>name</value>
            <value>number</value>
            <value>number-and-name</value>
            <value>plain-number-and-name</value>
            <value>plain-number</value>
        </choice>
    </attribute>
</define>
```

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 is used to specify the outline level to use. The chapter field displays the chapter number or title up to the specified outline level.

```
<define name="text-chapter-attlist" combine="interleave">
    <attribute name="text:outline-level">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

6.2.8 File Name Fields

File name fields display the name of the file that is currently being edited.

The attributes that may be associated with the `<text:file-name>` element are:

- Display
- Fixed

```
<define name="paragraph-content" combine="choice">
    <element name="text:file-name">
        <ref name="text-file-name-attlist"/>
        <text/>
    </element>
</define>
```

Display

The `text:display` attribute specifies how much of the file name to display. The following display options are allowed:

- 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 filename might be an IRI, either because an IRI has been used to retrieve the file, or the application internally uses IRIs and therefore converts even system specific paths into an IRI. If this is the case, and if the path, the name or the extension cannot be evaluated from the IRI, then the IRI should be displayed unmodified.

```
<define name="text-file-name-attlist" combine="interleave">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>full</value>
                <value>path</value>
                <value>name</value>
                <value>name-and-extension</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Fixed File Name Fields

If a file name field is fixed, its value does not change when the file is edited.

```
<define name="text-file-name-attlist" combine="interleave">
    <ref name="common-field-fixed-attlist"/>
</define>
```

6.2.9 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.

The only attribute that may be associated with the `<text:template-name>` element is:

- Display

```
<define name="paragraph-content" combine="choice">
    <element name="text:template-name">
        <ref name="text-template-name-attlist"/>
        <text/>
    </element>
```

```
</define>
```

Display

This attribute specifies which information about the document template to display. The following display options are allowed:

- 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 can be used for template dialogs. The values are a superset of the display values available for the `<text:file-name>` element.

```
<define name="text-template-name-attlist">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>full</value>
                <value>path</value>
                <value>name</value>
                <value>name-and-extension</value>
                <value>area</value>
                <value>title</value>
            </choice>
        </attribute>
    </optional>
</define>
```

6.2.10 Sheet Name Fields

For Spreadsheet documents, sheet name fields display the name of the sheet that is currently being edited.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sheet-name">
        <text/>
    </element>
</define>
```

6.3 Variable Fields

OpenDocument text 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 OpenDocument file format, a variable must be declared at the beginning of a document.

There are three types of variables:

- **Simple variables**

Simple variables, usually called variables, can take different values at different positions throughout a document. Simple variables can be 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 office application user interface, a user variable can be set at any occurrence of a user field, or by using user variable input fields. In the OpenDocument file format, the value of the user variable can only be set after the variable is declared.

- **Sequence variables**

  Sequence variables are used to number certain items in an OpenDocument text 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.

Variables must be declared before they can be used. The variable declarations are collected in container elements for the particular variable type. The OpenDocument code for declaring variables is described in sections 6.3.1, 6.3.5 and 6.3.8.

6.3.1 Declaring Simple Variables

Simple variables are declared using `<text:variable-decl>` elements. The declaration specifies the name and the value type of the variable.

To specify the name and value type of the simple variable, the following attributes are attached 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 6.7.3 for information on using this attribute.

- `office:value-type`

  See section 6.7.1 for information on using this attribute.

```
<define name="text-variable-decl">
    <element name="text:variable-decl">
        <ref name="common-field-name-attlist"/>
        <ref name="common-value-type-attlist"/>
    </element>
</define>
```

6.3.2 Setting Simple Variables

Simple variables can be set using variable setter elements. This element contains the presentation of the value of the variable, which can be empty if the `text:display` attribute is set to `none`.

The attributes that may be associated with 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 6.7.3 for information on using this attribute.

- `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, this attribute can be omitted. See section 6.7.6 for information on using this attribute.

- `office:value-type` and the appropriate value attribute

  See section 6.7.1 for information on using these attributes.

  Note: A simple variable should not contain different value types at different places in a document. However, an implementation may allow 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.

- `text:display`

  This attribute can be used 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 6.7.5 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:variable-set">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-formula-attlist"/>
            <ref name="common-value-and-type-attlist"/>
            <ref name="common-field-display-value-none-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.3 Displaying Simple Variables

The `<text:variable-get>` element reads and displays the value of a simple variable. 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.

The attributes that may be associated with 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-del>` element. See section 6.7.3 for information on using this attribute.

- `text:display`

  This attribute can be used to specify whether to display the formula for a simple variable or the computed value of the variable. See section 6.7.5 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:variable-get">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-display-value-formula-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.4 Simple Variable Input Fields

As an alternative to setting simple variables using formulas in variable setter elements, the user can be prompted for variable values. To do this, use the `<text:variable-input>` element. This element contains the presentation of the variable's value according to the chosen formatting style. The presentation can be empty if the `text:display` attribute is set to `none`.

The attributes that may be associated with 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 6.7.3 for information on using this attribute.

- `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 6.7.4 for information on using this attribute.

- `office:value-type` and the appropriate value attribute

  See section 6.7.1 for information on using these attributes.

- `text:display`

  This attribute can be used to specify whether to display or hide the value of the variable through the variable input field. See section 6.7.5 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:variable-input">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-description-attlist"/>
            <ref name="common-value-type-attlist"/>
            <ref name="common-field-display-value-none-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.5 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. For this reason, the value of user variables is stored in the variable declaration itself.

The attributes that may be associated with the `<text:user-field-del>` element are:

- `text:name`

  This attribute specifies the name of the variable to be declared. The name must be unique. It cannot already be used for any other type of variable including simple and sequence variables. See section 6.7.3 for information on using this attribute.

- `text:formula`

  This attribute contains the formula to compute the value of the user variable field. If the formula is the same as the content of the field element, this attribute can be omitted. See section 6.7.6 for information on using this attribute.

- `office:value-type` and the appropriate value attribute

  See section 6.7.1 for information on using these attributes.

```
<define name="text-user-field-decl">
    <element name="text:user-field-decl">
        <ref name="common-field-name-attlist"/>
        <optional>
            <ref name="common-field-formula-attlist"/>
        </optional>
        <ref name="common-value-and-type-attlist"/>
    </element>
</define>
```

6.3.6 Displaying User Variables

The content of user variables can be displayed using `<text:user-field-get>` elements.

The attributes that may be associated with 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-del>` element. See section 6.7.3 for information on using this attribute.

- `text:display`

This attribute can be used 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 6.7.5 for information on using this attribute.

Note: Since the office application user interfaces usually allow 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.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-field-get">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-display-value-formula-none-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.7 User Variable Input Fields

An alternative method of setting user variables is to use input fields, similar to the input fields for simple variables. A user variable can be set 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-del>` element, the <`text:user-field-input>` element does not contain the value and value type attributes from the <`text:variable-input>` field.

The presentation can be empty if the `text:display` attribute is set to `none`.

The attributes that may be associated with 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 6.7.3 for information on using this attribute.

- `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 6.7.4 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-field-input">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-description-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.8 Declaring Sequence Variables

Sequence variables are used to number items within an OpenDocument text document. Sequence variables are most commonly used for sequential numbering. However, expression formulas can be included in sequence fields to support more advanced sequences. See section 6.3.9 for more information on Using Sequence Fields and their uses.

Sequence variables are declared using the `<text:sequence-del>` element.

To facilitate chapter-specific numbering, attributes can be attached to a sequence variable to specify a chapter level and a separation character. The attributes that may be associated with the `<text:sequence-del>` element are:

- `text:name`

  This attribute specifies the name of the variable to be declared. The name must be unique. It cannot already be used for any other type of variable including simple and user variables. See section 6.7.3 for information on using this attribute.

- `text:display-outline-level`

  See section 6.3.8:Outline Level for information about this attribute.

- `text:separation-character`

  See section 6.3.8:Separation Character for information about this attribute.

```
<define name="text-sequence-decl">
    <element name="text:sequence-decl">
        <ref name="text-sequence-decl-attlist"/>
    </element>
</define>
<define name="text-sequence-decl-attlist" combine="interleave">
    <ref name="common-field-name-attlist"/>
</define>
```

Outline Level

Sequences can be numbered 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 prefixed to the sequence number. Choosing an outline level of zero results in a straight sequence of all sequence elements for that sequence variable.

```
<define name="text-sequence-decl-attlist" combine="interleave">
    <attribute name="text:display-outline-level">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

Separation Character

If sequences are numbered by chapter, this attribute is used to choose a character to separate the chapter number from the sequence number.

If the value of the `text:display-outline-level` attribute is a non-zero value, a separation character may be specified. The default separation character is `"."`.Otherwise, if the value of `text:display-outline-level` is zero, this attribute must be omitted.

```
<define name="text-sequence-decl-attlist" combine="interleave">
    <optional>
        <attribute name="text:separation-character">
            <ref name="character"/>
        </attribute>
    </optional>
</define>
```

Example: Sequence variable

The sequence variable 3.7.36#5 with a value of `5` is declared using:

Attribute	***Value***
`text:display-outline-level`	`3`
`text:separation-character`	`#`

6.3.9 Using Sequence Fields

Once a sequence variable is declared, it can be used 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, the office application 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.

The attributes that may be associated with 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 6.7.3 for information on using this attribute.

- `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 6.7.6 for information on using this attribute.

- `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 6.7.8 for information on these attributes.

- `text:ref-name`

  See the section 6.3.9:Reference Name for more information about this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sequence">
        <interleave>
            <ref name="common-field-name-attlist"/>
            <ref name="common-field-formula-attlist"/>
            <ref name="common-field-num-format-attlist"/>
            <ref name="text-sequence-ref-name"/>
        </interleave>
        <text/>
    </element>
</define>
```

Reference Name

Sequence fields can be the target of references, as implemented using reference fields. See section 6.6.5 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.

If the sequence field is not the target of a reference, this attribute can be omitted.

```
<define name="text-sequence-ref-name">
    <optional>
        <attribute name="text:ref-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

6.3.10 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.

The attributes that may be associated with the `<text:expression>` element are:

- `text:formula`

  This attribute contains the actual expression used to compute the value of the expression field. See section 6.7.6 for information on using this attribute.

- `office:value-type` and the appropriate value attribute

  See section 6.7.1 for information on using these attributes.

- `text:display`

Use this attribute to specify one of the following:

– To display the value of the field.

– To display the formula used to compute the value.

See section 6.7.5 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:expression">
        <interleave>
            <ref name="common-field-formula-attlist"/>
            <optional>
                <ref name="common-value-and-type-attlist"/>
            </optional>
            <ref name="common-field-display-value-formula-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.11 Text Input Fields

A text input field is a variable field. From the point of view of the 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.

The only attribute that may be associated with 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 6.7.4 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:text-input">
        <ref name="common-field-description-attlist"/>
        <text/>
    </element>
</define>
```

6.4 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 Chapter 3.

All metadata field elements can be marked as fixed using the `text:fixed` attribute. (Cf. section 6.7.2)

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 6.7.1 for more information on these attributes.

6.4.1 Initial Creator

This element represents the name of the author who created the original document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:initial-creator">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.2 Document Creation Date

This element represents the date on which the document was created.

```
<define name="paragraph-content" combine="choice">
    <element name="text:creation-date">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:date-value">
                    <ref name="dateOrDateTime"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.3 Document Creation Time

This element represents the time at which the document was created.

```
<define name="paragraph-content" combine="choice">
    <element name="text:creation-time">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:time-value">
                    <ref name="timeOrDateTime"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.4 Document Description

This element contains a brief description of the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:description">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.5 User-Defined Document Information

This element contains user-defined information about the document. It displays the information provided within a `<meta:user-defined>` element that has the same name.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-defined">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="office:value">
                    <ref name="double"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="office:date-value">
                    <ref name="dateOrDateTime"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="office:time-value">
                    <ref name="duration"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="office:boolean-value">
                    <ref name="boolean"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="office:string-value">
                    <ref name="string"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.6 Print Time

This element represents the time at which the document was last printed.

```
<define name="paragraph-content" combine="choice">
    <element name="text:print-time">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:time-value">
                    <ref name="time"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.7 Print Date

This element represents the date on which the document was last printed.

```
<define name="paragraph-content" combine="choice">
    <element name="text:print-date">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:date-value">
                    <ref name="date"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.8 Printed By

This element represents name of the last person who printed the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:printed-by">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.9 Document Title

This element represents the title of the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:title">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.10 Document Subject

This element represents the subject of the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:subject">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.11 Document Keywords

This element contains a list of keywords used to describe the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:keywords">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.12 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:editing-cycles">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

> **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.

6.4.13 Document Edit Duration

Every time a document is edited, the office application 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:editing-duration">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:duration">
                    <ref name="duration"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.14 Document Modification Time

This element represents the time at which the document was last modified.

This element displays the information from the `<dc:date>` element. The name was chosen to avoid confusion with `<text:date>` fields.

```
<define name="paragraph-content" combine="choice">
    <element name="text:modification-time">
        <interleave>
```

```
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:time-value">
                    <ref name="time"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.15 Document Modification Date

This element represents the date on which the document was last modified.

This element displays the information from the `<dc:date>` element. The name was chosen to avoid confusion with `<text:date>` fields.

```
<define name="paragraph-content" combine="choice">
    <element name="text:modification-date">
        <interleave>
            <ref name="common-field-fixed-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
            <optional>
                <attribute name="text:date-value">
                    <ref name="date"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.16 Document Modified By

This element represents the name of the person who last modified the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:creator">
        <ref name="common-field-fixed-attlist"/>
        <text/>
    </element>
</define>
```

6.4.17 Document Statistics Fields

These fields display how many objects of a certain type a document contains. They can be used to display the number of

- pages,
- paragraphs,
- words,
- characters,
- tables,

- images, or
- embedded objects.

```
<define name="paragraph-content" combine="choice">
    <element>
        <choice>
            <name>text:page-count</name>
            <name>text:paragraph-count</name>
            <name>text:word-count</name>
            <name>text:character-count</name>
            <name>text:table-count</name>
            <name>text:image-count</name>
            <name>text:object-count</name>
        </choice>
        <ref name="common-field-num-format-attlist"/>
        <text/>
    </element>
</define>
```

6.5 Database Fields

Documents can reference databases and display database information as text content. To display database information, the OpenDocument schema uses a group of text fields, collectively called database fields. Office applications may use database tables from SQL servers, therefore database fields can be used to access any SQL database, provided that the appropriate drivers are available.

A database may contain the following components:

- 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 text content, therefore they are not discussed in this chapter. From the point of view of embedding database information in OpenDocument 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.

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 office application user interface, database rows can be added in one of the following ways:

- Manually, using a data source browser and the data to fields function.
- Using the Form Letter menu item on the File menu. This menu item adds 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>` element. The `<text:database-select>` and `<text:database-next>` elements can be used to determine which row within the current selection should be displayed. The current row number for a particular table can be displayed using the `<text:database-row-number>` element. Finally, the `<text:database-name>` field displays the name of the most recently used database, which is the address book file database by default.

6.5.1 Database Field Data Source

A database field's source can either be the name of a database, or an IRI containing database connection resource data. If the source is a database name, then this name is used by all of the office application components to identify a database. All database fields contain a database name or connection resource, and most database fields also contain the name of a database table, which must be stored in the database. An additional attribute determines whether the database table refers to an SQL table, an OpenDocument query, or the result of a SQL command.

```
<define name="common-field-database-table">
    <ref name="common-field-database-table-attlist"/>
    <ref name="common-field-database-name"/>
</define>
```

Database Name

The `text:database-name` attribute specifies the source database by its name.

```
<define name="common-field-database-name" combine="choice">
    <optional>
        <attribute name="text:database-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Connection Resource

The `<form:connection-resource>` element specifies the source database by an [XLink]. Its `xlink:href` attribute either references a file containing a database, or it contains information on how to make a connection to a database, for instance a [JDBC] URL. See also section 11.1.20.

```
<define name="common-field-database-name" combine="choice">
    <ref name="form-connection-resource"/>
</define>
```

Database Table Name

The `text:table-name` attribute specifies a table within the source database.

```
<define name="common-field-database-table-attlist" combine="interleave">
    <attribute name="text:table-name">
        <ref name="string"/>
    </attribute>
</define>
```

Database Type

The `text:table-type` attribute determines whether the database table refers to an SQL table, an OpenDocument query, or the result of a SQL command.

```
<define name="common-field-database-table-attlist" combine="interleave">
    <optional>
        <attribute name="text:table-type">
            <choice>
                <value>table</value>
                <value>query</value>
                <value>command</value>
            </choice>
        </attribute>
    </optional>
</define>
```

6.5.2 Displaying Database Content

The `<text:database-display>` element displays data from a database. When a new data set is added to a document, all fields that display data from that database table update their content.

The attributes that may be associated with the `<text:database-display>` element are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes specify the database and database table that this field uses.

- `text:database-column-name`

  See section 6.5.2:Column Name for information about this attribute.

- `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 is used. See section 6.7.7 for more information about using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:database-display">
        <ref name="text-database-display-attlist"/>
        <text/>
    </element>
</define>
<define name="text-database-display-attlist" combine="interleave">
    <ref name="common-field-database-table"/>
</define>
<define name="text-database-display-attlist" combine="interleave">
    <ref name="common-field-data-style-name-attlist"/>
</define>
```

Column Name

The `text:column-name` attribute specifies the column from which to display the data. The value of this attribute must be a column contained in the specified database.

```
<define name="text-database-display-attlist" combine="interleave">
    <attribute name="text:column-name">
        <ref name="string"/>
    </attribute>
</define>
```

6.5.3 Selecting the Next Database Row

The `<text:database-next>` element changes the row in the current selection which is used for display in all following <`text:database-display>` fields. The next row from the current 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`.

The attributes that may be associated with the `<text:database-next>` are:

- `text:database-name`, `text:table-name` and `text:table-type`

These attributes specify the database and the database table that this field uses.

- `text:condition`

  See section 6.5.3:Condition for information about this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:database-next">
        <ref name="text-database-next-attlist"/>
    </element>
</define>
<define name="text-database-next-attlist" combine="interleave">
    <ref name="common-field-database-table"/>
</define>
```

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. Database field values can be used 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, it is assumes that the formula `true`, meaning that the next row is selected unconditionally.

```
<define name="text-database-next-attlist" combine="interleave">
    <optional>
        <attribute name="text:condition">
            <ref name="formula"/>
        </attribute>
    </optional>
</define>
```

Example:

```
text:formula='ooo-w:[address book file.address.FIRSTNAME] == "Julie"'
```

This example specifies a condition that is true if the current row from an address book database table 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 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 6.7.6 for more information on the formula syntax of a `text:condition` attribute, which is the same as that of the `text:formula` attribute.

6.5.4 Selecting a Row Number

The `<text:database-row-select>` element selects a specific row from the current selection. As with the `<text:database-row-next>` element, a condition can be specified so that the given row is only selected if the condition is `true`.

The attributes that may be associated with the `<text:database-row-select>` are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

- `text:condition`

  This attribute specifies the condition expression. See section 6.5.3 for a full explanation of how to use this attribute.

- `text:row-number`

  See the following section 6.5.4:Selecting the Row Number about this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:database-row-select">
        <ref name="text-database-row-select-attlist"/>
    </element>
</define>
<define name="text-database-row-select-attlist" combine="interleave">
    <ref name="common-field-database-table"/>
</define>
<define name="text-database-row-select-attlist" combine="interleave">
    <optional>
        <attribute name="text:condition">
            <ref name="formula"/>
        </attribute>
    </optional>
</define>
```

Selecting the Row Number

This attribute specifies the row number to select when a condition is `true`.

```
<define name="text-database-row-select-attlist" combine="interleave">
    <optional>
        <attribute name="text:row-number">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

6.5.5 Displaying the Row Number

The `<text:database-row-number>` element displays the current row number for a given table. 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.

The attributes that may be associated with the `<text:database-row-number>` are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

- `style:num-format` and `style:num-letter-sync`

  These attributes determine how the number should be formatted. See section 6.7.8 for more information on how to use these attributes.

- `text:value`

This attribute specifies the current row number. The number changes when new data is added to the current document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:database-row-number">
        <interleave>
            <ref name="common-field-database-table"/>
            <ref name="common-field-num-format-attlist"/>
            <optional>
                <attribute name="text:value">
                    <ref name="nonNegativeInteger"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.5.6 Display Current Database and Table

Office applications may 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. The `<text:database-name>` element displays the database and table name of the most recently used table.

The attributes that may be associated with the `<text:database-name>` element are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

```
<define name="paragraph-content" combine="choice">
    <element name="text:database-name">
        <ref name="common-field-database-table"/>
        <text/>
    </element>
</define>
```

6.6 More Fields

6.6.1 Page Variable Fields

Page variables allow an alternative page numbering scheme to be defined. 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, use the `<text:variable-page-set>` element.

```
<define name="paragraph-content" combine="choice">
    <element name="text:page-variable-set">
        <ref name="text-set-page-variable-attlist"/>
        <text/>
    </element>
</define>
```

Turning Page Variables On or Off

At the beginning of a document, the page variable is inactive. The `text:active` attribute can be used to disable a page variable after it was used in the document.

```
<define name="text-set-page-variable-attlist" combine="interleave">
    <optional>
        <attribute name="text:active">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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.

```
<define name="text-set-page-variable-attlist" combine="interleave">
    <optional>
        <attribute name="text:page-adjust">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Displaying Page Variable Fields

The `<text:variable-page-get>` element displays the value of the page variable. The field can be formatted in the same way as regular page number fields.

```
<define name="paragraph-content" combine="choice">
    <element name="text:page-variable-get">
        <ref name="text-get-page-variable-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated with the `<text:get-page-variable>` element are:

- `style:num-format` and `style:num-letter-sync`

  These attributes determine how the number should be formatted. See section 6.7.8 for more information on how to use these attributes.

```
<define name="text-get-page-variable-attlist" combine="interleave">
    <ref name="common-field-num-format-attlist"/>
</define>
```

6.6.2 Placeholders

The OpenDocument format 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 can be 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.

This element contains some brief text which is displayed with the placeholder.

```
<define name="paragraph-content" combine="choice">
    <element name="text:placeholder">
        <ref name="text-placeholder-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated 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 `text:placeholder-type` attribute represents the content type. 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`.

```
<define name="text-placeholder-attlist" combine="interleave">
    <attribute name="text:placeholder-type">
        <choice>
            <value>text</value>
            <value>table</value>
            <value>text-box</value>
            <value>image</value>
            <value>object</value>
        </choice>
    </attribute>
</define>
```

Placeholder Description

In addition to the brief text stored in the element content, may be associated 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 6.7.4 for information on using the `text:description` attribute.

```
<define name="text-placeholder-attlist" combine="interleave">
    <ref name="common-field-description-attlist"/>
</define>
```

6.6.3 Conditional Text Fields

Text fields can be used to display one text or another, depending on a 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:conditional-text">
        <ref name="text-conditional-text-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated with the `<text:conditional-text>` element are:

- Condition

- Text to display if the condition is true
- Text to display if the condition is false
- Current condition

The `text: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.

```
<define name="text-conditional-text-attlist" combine="interleave">
    <attribute name="text:condition">
        <ref name="formula"/>
    </attribute>
</define>
```

Text to Display if the Condition is True

The `text:string-value-if-true` attribute contains the text string to display if the condition is `true`.

```
<define name="text-conditional-text-attlist" combine="interleave">
    <attribute name="text:string-value-if-true">
        <ref name="string"/>
    </attribute>
</define>
```

Text to Display if the Condition is False

The `text:string-value-if-false` attribute contains the text string to display if the condition is `false`.

```
<define name="text-conditional-text-attlist" combine="interleave">
    <attribute name="text:string-value-if-false">
        <ref name="string"/>
    </attribute>
</define>
```

Current Value and Condition

The `text:current-value` attribute contains the evaluation result of the condition given by the expression in the `text:condition` attribute. Explicitly giving the result allows applications to delay evaluating the result until necessary. This attribute is valuable for the following reasons:

- If the expression is costly to evaluate, for example, the expression contains references to several databases.
- To allow transformations to correctly display the state of the document without having to parse and evaluate the condition.

```
<define name="text-conditional-text-attlist" combine="interleave">
    <optional>
        <attribute name="text:current-value">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Note: The value of this attribute is overwritten with a new value as soon as the application evaluates the expression. This attribute has no function other than to ease transformation or initially display the document.

6.6.4 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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:hidden-text">
        <ref name="text-hidden-text-attlist"/>
        <text/>
    </element>
</define>
```

The attributes that may be associated with the `<text:hidden-text>` element are:

- Condition
- Text
- Is hidden

Condition

The `text:condition` attribute contains a Boolean expression. If the expression evaluates to `true`, the text is hidden.

```
<define name="text-hidden-text-attlist" combine="interleave">
    <attribute name="text:condition">
        <ref name="formula"/>
    </attribute>
</define>
```

Text

The `text:string-value` attribute specifies the text to display if the condition is `false`.

```
<define name="text-hidden-text-attlist" combine="interleave">
    <attribute name="text:string-value">
        <ref name="string"/>
    </attribute>
</define>
```

Is Hidden

The `text:is-hidden` attribute specifies whether or not the field is currently visible. The purpose of this attribute is similar to that of the `text:current-value` attribute in the `text:condition` field. Recording the result allows transformations to correctly represent the document without having to parse the condition expression or evaluate the condition when loading the document.

```
<define name="text-hidden-text-attlist" combine="interleave">
    <optional>
        <attribute name="text:is-hidden">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Note: The value of this attribute is overwritten with a new value as soon as the application evaluates the expression. This attribute has no function other than to ease transformation or initially display the document.

6.6.5 Reference Fields

The OpenDocument format uses four 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; notes, 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 are assigned names by the application used to create the OpenDocument file format automatically.

```
<define name="paragraph-content" combine="choice">
    <element>
        <choice>
            <name>text:reference-ref</name>
            <name>text:bookmark-ref</name>
        </choice>
        <interleave>
            <ref name="text-common-ref-content"/>
            <ref name="text-ref-content"/>
        </interleave>
    </element>
</define>
<define name="paragraph-content" combine="choice">
    <element name="text:note-ref">
        <interleave>
            <ref name="text-common-ref-content"/>
            <ref name="text-note-ref-content"/>
            <ref name="text-ref-content"/>
        </interleave>
    </element>
</define>
<define name="paragraph-content" combine="choice">
    <element name="text:sequence-ref">
        <interleave>
            <ref name="text-common-ref-content"/>
            <ref name="text-sequence-ref-content"/>
        </interleave>
    </element>
</define>
<define name="text-common-ref-content" combine="interleave">
    <text/>
</define>
```

The attributes that may be associated 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 are identified by a name that is usually generated automatically.

```
<define name="text-common-ref-content" combine="interleave">
    <optional>
        <attribute name="text:ref-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Note Class

For `<text:note-ref>` elements, the `text:note-class` attribute determines whether the field references a foot- or an endnote.

```
<define name="text-note-ref-content" combine="interleave">
    <ref name="text-note-class"/>
</define>
```

Reference Format

The `text:reference-format` attribute determines what information about the reference is displayed. If the reference format is not specified, the page format is used as the default.

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.

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.

```
<define name="text-ref-content" combine="interleave">
    <optional>
        <attribute name="text:reference-format">
            <choice>
                <value>page</value>
                <value>chapter</value>
                <value>direction</value>
                <value>text</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="text-sequence-ref-content" combine="interleave">
    <optional>
```

```
        <attribute name="text:reference-format">
            <choice>
                <value>page</value>
                <value>chapter</value>
                <value>direction</value>
                <value>text</value>
                <value>category-and-value</value>
                <value>caption</value>
                <value>value</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Example: Different reference formats and displays

The following table shows all possible reference formats and the resulting reference display that can be used to refer to the table itself. The left column lists the value of the `text:reference-format` attribute and the right column

Reference format	***Reference display***
`page`	138
`chapter`	3.7.27
`text`	Table 2: Examples of reference formats
`direction`	above
`category-and-value`	Table 1
`caption`	Examples of reference formats
`value`	1

6.6.6 Script Fields

A script field stores scripts or sections of scripts. The field can be used to store and edit scripts that are attached to the document. The primary purpose of this field is to provide an equivalent to the `<script>` element in [HTML4], so that the content of a `<script>` element in HTML can be imported, edited, and exported using an office application software.

The source code for the script can be stored in one of the following ways:

- The `<text:script>` element contains the source code.
- The source code is stored in an external file. Use the `xlink:href` attribute to specify the location of the source file.

The element should have either a `xlink:href` attribute or content, but not both.

```
<define name="paragraph-content" combine="choice">
    <element name="text:script">
        <interleave>
            <choice>
                <group>
                    <attribute name="xlink:href">
                        <ref name="anyURI"/>
                    </attribute>
                    <optional>
                        <attribute name="xlink:type" a:defaultValue="simple">
                            <value>simple</value>
                        </attribute>
                    </optional>
                </group>
                <text/>
            </choice>
            <optional>
                <attribute name="script:language">
                    <ref name="string"/>
                </attribute>
            </optional>
        </interleave>
    </element>
</define>
```

Script URL

The `xlink:href` attribute specifies the location of the file that contains the script source code. The script field should have either an URL attribute or content, but not both.

Script Language

The `script:language` attribute specifies the language in which the script source code is written, for example, JavaScript.

6.6.7 Macro Fields

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.

The only attribute that may be associated with the `<text:execute-macro>` element is:

- Macro name

```
<define name="paragraph-content" combine="choice">
    <element name="text:execute-macro">
        <optional>
            <attribute name="text:name">
                <ref name="string"/>
            </attribute>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <text/>
    </element>
</define>
```

Macro Name

The `text:name` attribute specifies the macro to invoke when the field is activated.

6.6.8 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 a paragraph of formatted text to be hidden or displayed 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, the conditions associated with several hidden paragraph fields can be combined into a single condition for a single field using logical operations on the conditions.

> **Note**: Unlike most fields, this field does not display text, but it affects the entire paragraph in which it is contained.

The attributes that may be associated with the `<text:hidden-paragraph>` element are:

- Condition
- Is hidden

```
<define name="paragraph-content" combine="choice">
    <element name="text:hidden-paragraph">
        <ref name="text-hidden-paragraph-attlist"/>
        <text/>
    </element>
</define>
```

Condition

The `text:condition` attribute contains a Boolean expression. If the condition is `true`, the paragraph is hidden. If the condition is `false`, the paragraph is displayed.

```
<define name="text-hidden-paragraph-attlist" combine="interleave">
    <attribute name="text:condition">
        <ref name="formula"/>
    </attribute>
</define>
```

Is Hidden

The `text:is-hidden` attribute records whether the paragraph is currently visible or not. It has the same purpose as the corresponding attribute of the hidden text field, namely to allow correct display of the paragraph without having to evaluate the condition first. The value of this attribute is overwritten with a new value as soon as the application evaluates the expression.

> **Note**: This attribute has no function other than to ease transformation or initially display the document.

```
<define name="text-hidden-paragraph-attlist" combine="interleave">
    <optional>
        <attribute name="text:is-hidden">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

6.6.9 DDE Connection Fields

A DDE field allows information from a DDE connection to be displayed. 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.

The field element contains the content of the most recent data that was received from the DDE connection. This may be used to render the document if the DDE connection cannot be accessed.

See section 12.6 for the use of DDE connections.

```
<define name="paragraph-content" combine="choice">
    <element name="text:dde-connection">
        <attribute name="text:connection-name">
            <ref name="string"/>
        </attribute>
        <text/>
    </element>
</define>
```

The only attribute that may be associated 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.

6.6.10 Measure Fields

Within the text contained in measure drawing objects (see section 9.2.11), a `<text:measure>` field displays the current measure. The `draw:kind` attribute specifies which part of the measure is displayed. It my have one of the following values:

- `value`: The measure's value is displayed, for instance “12”
- `unit`: The measure's unit is displayed, for instance “inch”
- `gap`: A gap or blank is displayed if and only if the measure text's writing direction is perpendicular to the measure line. The purpose of this value is add some space between the measure line and the text if the text is displayed perpendicular to the measure line.

```
<define name="paragraph-content" combine="choice">
    <element name="text:measure">
        <attribute name="text:kind">
            <choice>
                <value>value</value>
                <value>unit</value>
                <value>gap</value>
            </choice>
        </attribute>
        <text/>
    </element>
</define>
```

6.6.11 Table Formula Field

The table formula field is a legacy from previous versions of current office applications. It should not be used in new documents. It stores a formula to be used in tables, a function that is better performed by the table:formula attribute of the table cell.

> **Note**: This element should not be used in new documents.

The table formula field can take the following attributes:

- `text:formula`

  This attribute contains the actual expression used to compute the value of the table formula field. See section 6.7.6 for information on using this attribute.

- `text:display`

  Use this attribute to specify one of the following:

  – To display the value of the field.

  – To display the formula used to compute the value.

  See section 6.7.5 for information on using this attribute.

- `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 6.7.7 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:table-formula">
        <interleave>
            <ref name="common-field-formula-attlist"/>
            <ref name="common-field-display-value-formula-attlist"/>
            <ref name="common-field-data-style-name-attlist"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.7 Common Field Attributes

The attributes described in this section can be used with several field elements.

6.7.1 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 `office:value-type` attribute.

```
<define name="common-value-type-attlist">
    <attribute name="office:value-type">
        <ref name="valueType"/>
    </attribute>
</define>
```

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 Type	***Value Attribute(s)***	***Encoded as...***	***Example***
`float`	`office:value`	Numeric value	`"12.345"`
`percentage`	`office:value`	Numeric value	`"0.50"`
`currency`	`office:value` and `office:currency`	Numeric value and currency symbol	`"100"` `"USD"`
`date`	`office:date-value`	Date value as specified in §3.2.9 of [xmlschema-2], or date and time value as specified in §3.2.7 of [xmlschema-2]	`"2003-04-17"`
`time`	`office:time-value`	Duration, as specified in §3.2.6 of [xmlschema-2]	`"PT03H30M00S"`
`boolean`	`office:boolean-value`	`true` or `false`	`"true"`
`string`	`office:string-value`	Strings	`"abc def"`

The OpenDocument concept of field values and value types and their encoding in XML is modeled on the corresponding XML for table cell attributes. See section 8.1.3 for information on table cells and their attributes.

The definition of the entity `%value-attlist;` is as follows:

```
<define name="common-value-and-type-attlist">
    <choice>
        <group>
            <attribute name="office:value-type">
                <value>float</value>
            </attribute>
            <attribute name="office:value">
                <ref name="double"/>
            </attribute>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>percentage</value>
            </attribute>
            <attribute name="office:value">
                <ref name="double"/>
            </attribute>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>currency</value>
            </attribute>
            <attribute name="office:value">
                <ref name="double"/>
            </attribute>
            <optional>
                <attribute name="office:currency">
                    <ref name="string"/>
                </attribute>
            </optional>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>date</value>
            </attribute>
            <attribute name="office:date-value">
                <ref name="dateOrDateTime"/>
            </attribute>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>time</value>
            </attribute>
            <attribute name="office:time-value">
                <ref name="duration"/>
            </attribute>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>boolean</value>
            </attribute>
            <attribute name="office:boolean-value">
                <ref name="boolean"/>
            </attribute>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>string</value>
            </attribute>
            <optional>
                <attribute name="office:string-value">
                    <ref name="string"/>
                </attribute>
            </optional>
        </group>
    </choice>
</define>
```

6.7.2 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 value of the field element to which this attribute is attached is preserved in all future edits of the document. If the value of the field is not fixed, the value of the field may be 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

```
<define name="common-field-fixed-attlist">
    <optional>
        <attribute name="text:fixed">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

6.7.3 Variable Name

Use the `text:name` attribute to specify the name of a variable when it is being declared, set, or displayed a variable. This attribute can be used with any of the following elements:

- `<text:variable-del>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-del>`
- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:sequence-del>`
- `<text:sequence>`

When this attribute is being used to specify the name of a variable to display, a variable of the appropriate type with the same name must already have been declared.

```
<define name="common-field-name-attlist">
    <attribute name="text:name">
        <ref name="variableName"/>
    </attribute>
</define>
```

6.7.4 Description

The `text:description` attribute contains a brief message that is displayed when users are prompted for input. This attribute can be used with any of the following elements:

- `<text:placeholder>`
- `<text:variable-input>`
- `<text:user-field-input>`
- `<text:text-input>`

```
<define name="common-field-description-attlist">
    <optional>
        <attribute name="text:description">
            <text/>
        </attribute>
    </optional>
</define>
```

6.7.5 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 the display of 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 variables to be set in one part of the document and displayed in another part of the document.

This attribute can be used with any of the following elements:

- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-get>`
- `<text:expression>`

```
<define name="common-field-display-value-none-attlist">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>value</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="common-field-display-value-formula-none-attlist">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>value</value>
                <value>formula</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="common-field-display-value-formula-attlist">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>value</value>
                <value>formula</value>
            </choice>
        </attribute>
    </optional>
</define>
```

6.7.6 Formula

The `text:formula` attribute contains the formula or expression used to compute the value of the field. This attribute can be used with any of the following elements:

- `<text:variable-set>`
- `<text:user-field-del>`

- `<text:sequence>`
- `<text:expression>`

The formula should start with a namespace prefix that indicates the syntax and semantic used within the formula.

```
<define name="common-field-formula-attlist">
    <optional>
        <attribute name="text:formula">
            <ref name="formula"/>
        </attribute>
    </optional>
</define>
```

6.7.7 Formatting Style

The `style:data-style-name` attribute refers to the data style used to format the numeric value. For general information about styles, see Chapter 14. For more information about data styles, see section 14.7.

For string variables this attribute must be omitted. Otherwise, this attribute is required.

The name must match the name of a data style.

This attribute can be used 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>`

```
<define name="common-field-data-style-name-attlist">
    <optional>
        <attribute name="style:data-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

6.7.8 Number Formatting Style

Numbers that are used for number sequences such as page numbers or sequence fields can be formatted according to the number styles described in section 12.2. 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,...

  Note: The value of this attribute can be any of the [XSLT] number format keys `1`, `i`, `I`, `a`, or `A`.

Alphabetic number styles need an additional attribute to determine how to display numbers that cannot be represented by a single letter. The OpenDocument format 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 12.2 for more information.

```
<define name="common-field-num-format-attlist">
    <optional>
        <ref name="common-num-format-attlist"/>
    </optional>
</define>
```

7 Text Indices

OpenDocument text documents may contain automatically generated indices. An index generally contains a sorted list of all items of a certain types, where the sorting (document position, alphabetical, etc.) and the type of items (chapter headings, tables, etc.) are determined by the specific type of index.

7.1 Index Marks

There are three types of index marks that correspond to the three types of index that make 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.

7.1.1 Table of Content Index Marks

The `<text:toc-mark-start>` element marks the start of a table of content index entry. The ID specified by the `text:id` attribute must be unique except for the matching index mark end element. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
    <element name="text:toc-mark-start">
        <ref name="text-toc-mark-start-attrs"/>
    </element>
</define>
```

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.

```
<define name="text-toc-mark-start-attrs">
    <ref name="text-id"/>
    <ref name="text-outline-level"/>
```

```
</define>
<define name="text-outline-level">
    <optional>
        <attribute name="text:outline-level">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
<define name="text-id">
    <attribute name="text:id">
        <ref name="string"/>
    </attribute>
</define>
```

The `<text:toc-mark-end>` element marks the end of a table of contents index entry. There must be a start element with the same `text:id` value to match the end element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
    <element name="text:toc-mark-end">
        <ref name="text-id"/>
    </element>
</define>
```

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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:toc-mark">
        <attribute name="text:string-value">
            <ref name="string"/>
        </attribute>
        <ref name="text-outline-level"/>
    </element>
</define>
```

7.1.2 User-Defined Index Marks

The `<text:user-index-mark-start>` element marks the start of a user-defined index entry. The ID specified by the `text:id` attribute must be unique except for the matching index mark end element. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-index-mark-start">
        <ref name="text-id"/>
        <ref name="text-outline-level"/>
        <ref name="text-index-name"/>
    </element>
</define>
```

The `<text:user-index-mark-end>` element marks the end of the user-defined index entry. There must be a start element with the same `text:id` value to match the end element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-index-mark-end">
        <ref name="text-id"/>
        <ref name="text-outline-level"/>
    </element>
</define>
```

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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-index-mark">
        <attribute name="text:string-value">
            <ref name="string"/>
        </attribute>
        <ref name="text-outline-level"/>
        <ref name="text-index-name"/>
    </element>
</define>
```

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.

```
<define name="text-index-name">
    <attribute name="text:index-name">
        <ref name="string"/>
    </attribute>
</define>
```

7.1.3 Alphabetical Index Mark

The `<text:alpha-index-mark-start>` element marks the start of an alphabetical index entry. There are two optional attributes that may contain keys for alphabetical entries, which allows structuring of entries. There is also a Boolean attribute that determines if this entry is intended to be the main entry, if there are several equal entries.

The ID specified by the text:id attribute must be unique except for the matching index mark end element. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
    <element name="text:alphabetical-index-mark-start">
        <ref name="text-id"/>
        <ref name="text-alphabetical-index-mark-attrs"/>
    </element>
</define>
```

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. There must be a start element with the same text:id value to match the end element located in the same paragraph, with the start element appearing first.

```
<define name="paragraph-content" combine="choice">
```

```
    <element name="text:alphabetical-index-mark-end">
        <ref name="text-id"/>
    </element>
</define>
```

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.

```
<define name="paragraph-content" combine="choice">
    <element name="text:alphabetical-index-mark">
        <attribute name="text:string-value">
            <ref name="string"/>
        </attribute>
        <ref name="text-alphabetical-index-mark-attrs"/>
    </element>
</define>
```

Additional Keys

The `text:key1` and `text:key2` attributes specify additional keys for the alphabetical index mark. If only one key is used, it must be contained in the `text:key1` attribute.

```
<define name="text-alphabetical-index-mark-attrs" combine="interleave">
    <optional>
        <attribute name="text:key1">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:key2">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Phonetic Keys

For ideographic languages, there sometimes is no obvious or common sorting of the language's characters. One common scheme to facilitate an alphabetical index in such languages is to sort according to a phonetic description of the search time. To achieve this in the OpenDocument file format, there are additional attributes for the string value and the two keys for phonetic descriptions. The original value and key attributes are for display, but if phonetic variants are present, they should be used for sorting the index.

```
<define name="text-alphabetical-index-mark-attrs" combine="interleave">
    <optional>
        <attribute name="text:string-value-phonetic">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:key1-phonetic">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:key2-phonetic">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Main Entry

If there are several index marks for the same entry, one of these entries may be declared as the main entry using the `text:main-entry` attribute.

```
<define name="text-alphabetical-index-mark-attrs" combine="interleave">
    <optional>
        <attribute name="text:main-entry" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

7.1.4 Bibliography Index Mark

The `<text:bibliography-mark>` element contains the text and information for a bibliography index entry. It supports attributes for each type of bibliographical data that a bibliography index may contain.

```
<define name="paragraph-content" combine="choice">
    <element name="text:bibliography-mark">
        <attribute name="text:bibliography-type">
            <ref name="text-bibliography-types"/>
        </attribute>
        <zeroOrMore>
            <attribute>
                <choice>
                    <name>text:identifier</name>
                    <name>text:address</name>
                    <name>text:annote</name>
                    <name>text:author</name>
                    <name>text:booktitle</name>
                    <name>text:chapter</name>
                    <name>text:edition</name>
                    <name>text:editor</name>
                    <name>text:howpublished</name>
                    <name>text:institution</name>
                    <name>text:journal</name>
                    <name>text:month</name>
                    <name>text:note</name>
                    <name>text:number</name>
                    <name>text:organizations</name>
                    <name>text:pages</name>
                    <name>text:publisher</name>
                    <name>text:school</name>
                    <name>text:series</name>
                    <name>text:title</name>
                    <name>text:report-type</name>
                    <name>text:volume</name>
                    <name>text:year</name>
                    <name>text:url</name>
                    <name>text:custom1</name>
                    <name>text:custom2</name>
                    <name>text:custom3</name>
                    <name>text:custom4</name>
                    <name>text:custom5</name>
                    <name>text:isbn</name>
                    <name>text:issn</name>
                </choice>
                <ref name="string"/>
            </attribute>
        </zeroOrMore>
        <text/>
    </element>
</define>
<define name="text-bibliography-types">
    <choice>
        <value>article</value>
        <value>book</value>
        <value>booklet</value>
        <value>conference</value>
        <value>custom1</value>
        <value>custom2</value>
        <value>custom3</value>
        <value>custom4</value>
        <value>custom5</value>
        <value>email</value>
        <value>inbook</value>
        <value>incollection</value>
        <value>inproceedings</value>
        <value>journal</value>
        <value>manual</value>
        <value>mastersthesis</value>
        <value>misc</value>
        <value>phdthesis</value>
        <value>proceedings</value>
        <value>techreport</value>
        <value>unpublished</value>
        <value>www</value>
    </choice>
</define>
```

7.2 Index Structure

An index consists of two parts: The index source, and the index body. Both of these are contained in an element of their own, which in turn form the two child elements for the index element itself.

The index source is specific to the type of index it is being used for. It contains the information necessary to generate the index content. An index source has no graphical rendition.

The index body is the same for all types of indices. It contains the text generated from the information in the index source. The text contained in an index body is in no way special or different from text used elsewhere in this specification.

The content of the index body can be regenerated at any time from the information contained in the index source and the remainder of the document. One could say that the index source contains all the logical information about an index, while the index body contains the rendition of the index. A tool extracting structure information about a document might look only at the index source, while a rendering program might look only at an index body.

7.2.1 Index Source

An index source element contains the information necessary to generate the index body. In addition to a set of flags that determine which information to include in an index, the index source contains a set of index templates. Such a template determines how an item to be contained in the index is to be rendered.

For example, a table of content might look as follows:

```
1 Introduction.......................................................................... .........................7
1.1 Namespaces........................................................................ .................7
1.2 Relax-NG Schema Prefix.................................................... ..................8
```

An index source for this index would contain flags indicating that chapter headers at least up to level 2 are to be included. The contained index templates would define that an entry consists of the chapter number, a space, the chapter name, a tab (with a '.' leader) and the page number.

The various index templates are described together with their index elements. The index templates elements in use are described in section 7.12.

The different index source elements are described together with their corresponding index elements.

7.2.2 Index Body Section

The index body contains the current textual rendition of the index. The format is the same as for regular text within this specification, e.g., text sections, except that it also allows index title sections.

```
<define name="text-index-body">
    <element name="text:index-body">
        <zeroOrMore>
            <ref name="index-content-main"/>
        </zeroOrMore>
    </element>
</define>
<define name="index-content-main">
    <choice>
        <ref name="text-content"/>
        <ref name="text-index-title"/>
    </choice>
</define>
```

7.2.3 Index Title Section

The index title is usually contained in a section of its own. The reason for this enclosure is to enable the popular layout of having an index title across the entire page, but having the index itself in a two column layout.

```
<define name="text-index-title">
    <element name="text:index-title">
        <ref name="sectionAttr"/>
        <zeroOrMore>
            <ref name="index-content-main"/>
        </zeroOrMore>
    </element>
</define>
```

7.3 Table Of Content

A table of contents provides the user with a guide through the content of the document. It is typically found at the beginning of a document, contains the chapter headings with their respective page numbers. An example for a table of content may be found at the beginning of this document.

The items that can be listed in a table of content are:

- Headers (as defined by the outline structure of the document), up to a selectable level
- Table of content index marks
- Paragraphs formatted with a set of selectable paragraph styles

The table of contents is represented by the `<text:table-of-content>` element. The `<text:table-of-content>` element supports the same style (and class) attributes as a text section (see section 4.4).

```
<define name="text-table-of-content">
    <element name="text:table-of-content">
        <ref name="sectionAttr"/>
        <ref name="text-table-of-content-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.3.1 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.

The `<text:table-of-content-source>` element contains

- an optional template for the index title
- optional templates for index entries, one per level
- optionally a list of styles to be used for gathering index entries

```
<define name="text-table-of-content-source">
    <element name="text:table-of-content-source">
        <ref name="text-table-of-content-source-attlist"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <zeroOrMore>
            <ref name="text-table-of-content-entry-template"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="text-index-source-styles"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<text:table-of-content-source>` element are:

- Outline level
- Use outline
- Use index marks
- Use index source styles
- Index source
- Relative tab stop position

Outline Level

The `text:outline-level` attribute specifies which outline levels are used when generating the table of contents.

The value of this attribute must be an integer greater than zero. If this attribute is omitted, all outline levels are used by default.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:outline-level">
            <choice>
                <ref name="positiveInteger"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Use Outline

The `text:use-outline-level` attribute determines whether headings are used to generate index entries. If the value is `true`, the table of contents includes entries generated from headings. The `text:outline-level` attribute specifies up to which level headings are being included. See section 7.1 for more information on index marks.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:use-outline-level" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Use Index Marks

The `text:use-index-marks` attribute determines whether or not index marks are used to generate index entries. If the value is `true`, the table of contents includes entries generated from table of content index marks. The `text:outline-level` attribute specifies up to which level index marks are being included. See section 7.1 for more information on index marks.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:use-index-marks">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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. 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. The `text:outline-level` attribute specifies up to which level index source styles are being included.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:use-index-source-styles">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Index Scope

The `text:index-scope` attribute determines whether the table-of-content is generated for the whole document, or only for the current chapter.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:index-scope">
            <choice>
                <value>document</value>
                <value>chapter</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Relative Tab-Stop Position

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 for copying 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.

```
<define name="text-table-of-content-source-attlist" combine="interleave">
    <optional>
        <attribute name="text:relative-tab-stop-position">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

7.3.2 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. For each table of content, there must not be more than one element for any outline level. (See below.)

```
<define name="text-table-of-content-entry-template">
    <element name="text:table-of-content-entry-template">
        <ref name="text-table-of-content-entry-template-attlist"/>
        <zeroOrMore>
            <ref name="text-table-of-content-children"/>

        </zeroOrMore>
    </element>
</define>
```

A table of content entry template supports the following kinds of text elements:

- Chapter and Page Number
- Reference Text
- Text Span

- Tab
- Hyperlink start and end

```
<define name="text-table-of-content-children">
    <choice>
        <ref name="text-index-entry-chapter"/>
        <ref name="text-index-entry-page-number"/>
        <ref name="text-index-entry-text"/>
        <ref name="text-index-entry-span"/>
        <ref name="text-index-entry-tab-stop"/>
        <ref name="text-index-entry-link-start"/>
        <ref name="text-index-entry-link-end"/>
    </choice>
</define>
```

The attributes that may be associated 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. Outline levels must be unique for the template elements in one index source.

```
<define name="text-table-of-content-entry-template-attlist"
        combine="interleave">
    <attribute name="text:outline-level">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

```
<define name="text-table-of-content-entry-template-attlist"
        combine="interleave">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

7.4 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.

The attribute that may be attached to the `<text:illustration-index>` element is:

- `text:style-name`

  This attribute specifies the section style to use for the index of illustrations.

```
<define name="text-illustration-index">
    <element name="text:illustration-index">
        <ref name="sectionAttr"/>
        <ref name="text-illustration-index-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.4.1 Index of Illustration Source

The `<text:illustration-index-source>` element specifies how the index of illustrations is generated.

```
<define name="text-illustration-index-source">
    <element name="text:illustration-index-source">
        <ref name="text-illustration-index-source-attrs"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <optional>
            <ref name="text-illustration-index-entry-template"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated 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.
- `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.

```
<define name="text-illustration-index-source-attrs" combine="interleave">
    <ref name="text-index-scope-attr"/>
</define>
<define name="text-index-scope-attr">
    <optional>
        <attribute name="text:index-scope" a:defaultValue="document">
            <choice>
                <value>document</value>
                <value>chapter</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="text-illustration-index-source-attrs" combine="interleave">
    <ref name="text-relative-tab-stop-position-attr"/>
</define>
<define name="text-relative-tab-stop-position-attr">
    <optional>
        <attribute name="text:relative-tab-stop-position"
                   a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Use Caption

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.

```
<define name="text-illustration-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-caption" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Caption Sequence Name

Captions are associated with a sequence name. If the `text:use-caption` attribute is set to `true`, this attribute must be used to specify the sequence with which the captions are associated.

If this attribute is omitted, the default sequence for the object type is used, for example the sequence “Illustration” is used for illustrations.

```
<define name="text-illustration-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:caption-sequence-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Caption Sequence Format

If the entries for the index of illustrations are obtained from the image captions, this attribute must be used to specify the format for the entries.

```
<define name="text-illustration-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:caption-sequence-format">
            <choice>
                <value>text</value>
                <value>category-and-value</value>
                <value>caption</value>
            </choice>
        </attribute>
    </optional>
</define>
```

7.4.2 Illustration Index Entry Template

The illustration index entry template element determines the format of an index entry for a particular outline level.

```
<define name="text-illustration-index-entry-template">
    <element name="text:illustration-index-entry-template">
        <ref name="text-illustration-index-entry-content"/>
    </element>
</define>
```

```
<define name="text-illustration-index-entry-content">
    <ref name="text-illustration-index-entry-template-attrs"/>
    <zeroOrMore>
        <choice>
            <ref name="text-index-entry-page-number"/>
            <ref name="text-index-entry-text"/>
            <ref name="text-index-entry-span"/>
            <ref name="text-index-entry-tab-stop"/>
        </choice>
    </zeroOrMore>
</define>
```

The attribute that may be associated with the `<text:illustration-index-entry-template>` element is:

- Paragraph style

Paragraph Style

This attribute identifies the paragraph style to use for this template.

```
<define name="text-illustration-index-entry-template-attrs">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

7.5 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.

```
<define name="text-table-index">
    <element name="text:table-index">
        <ref name="sectionAttr"/>
        <ref name="text-table-index-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.5.1 Table Index Source

The `<text:table-index-source>` element specifies how the index of tables is generated.

The attributes that may be associated with this element are the same as those that can be associated with the `<text:illustration-index-source>` element. See section 7.4.1 for detailed information about these attributes.

```
<define name="text-table-index-source">
    <element name="text:table-index-source">
        <ref name="text-illustration-index-source-attrs"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <optional>
            <ref name="text-table-index-entry-template"/>
        </optional>
    </element>
</define>
```

7.5.2 Table Index Entry Template

The table index entry template element determines the format of an index entry for a particular outline level.

The attributes that may be associated with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See section 7.4.2 for detailed information about these attributes.

```
<define name="text-table-index-entry-template">
    <element name="text:table-index-entry-template">
        <ref name="text-illustration-index-entry-content"/>
    </element>
</define>
```

7.6 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.

```
<define name="text-object-index">
    <element name="text:object-index">
        <ref name="sectionAttr"/>
        <ref name="text-object-index-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.6.1 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.

```
<define name="text-object-index-source">
    <element name="text:object-index-source">
        <ref name="text-object-index-source-attrs"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <optional>
            <ref name="text-object-index-entry-template"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<text:object-index-source>` element are:

- Use attributes, `text:use-*-objects`
- Index scope (see section 7.4.1)

  This attribute specifies whether the index applies to the entire document or only the the current chapter.
- Relative tab stop position (see section 7.4.1)

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

```
<define name="text-object-index-source-attrs" combine="interleave">
    <ref name="text-index-scope-attr"/>
```

```
</define>
<define name="text-object-index-source-attrs" combine="interleave">
    <ref name="text-relative-tab-stop-position-attr"/>
</define>
```

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 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`

```
<define name="text-object-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-spreadsheet-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
<define name="text-object-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-math-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
<define name="text-object-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-draw-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
<define name="text-object-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-chart-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
<define name="text-object-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-other-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

7.6.2 Object Index Entry Template

The object index entry template element determines the format of an index entry for a particular outline level.

```
<define name="text-object-index-entry-template">
    <element name="text:object-index-entry-template">
        <ref name="text-illustration-index-entry-content"/>
    </element>
</define>
```

The attributes that may be associated with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See section 7.4.2 for detailed information about these attributes.

7.7 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.

```
<define name="text-user-index">
    <element name="text:user-index">
        <ref name="sectionAttr"/>
        <ref name="text-user-index-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.7.1 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.

```
<define name="text-user-index-source">
    <element name="text:user-index-source">
        <ref name="text-user-index-source-attr"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <zeroOrMore>
            <ref name="text-user-index-entry-template"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="text-index-source-styles"/>
        </zeroOrMore>
    </element>
```

```
</define>
```

The attributes that may be associated with `<text:user-index-source>` elements are:

- Use attributes, `text:use-*`
- Copy outline level
- Index scope (see section 7.4.1)

  This attribute specifies whether the index applies to the entire document or only to the current chapter.
- Index name

  In order to support several user-defined indexes with different contents, user index marks have a `text:index-name` attribute. The same attribute can be used with a `<text:user-index-source>` element to specify which index marks apply to the current index.
- Relative tab stop position (see section 7.4.1)

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

```
<define name="text-user-index-source-attr" combine="interleave">
    <ref name="text-index-scope-attr"/>
    <ref name="text-relative-tab-stop-position-attr"/>
    <attribute name="text:index-name">
        <ref name="string"/>
    </attribute>
</define>
```

Use Attributes

The `text:use-*` attributes specify which entries to include in the user-defined index. The following attributes exist:

- `text:use-index-marks`
- `text:use-graphics`
- `text:use-tables`
- `text:use-floating-frames`
- `text:use-objects`

```
<define name="text-user-index-source-attr" combine="interleave">
    <optional>
        <attribute name="text:use-index-marks" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:use-graphics" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:use-tables" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:use-floating-frames"
                     a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:use-objects" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Copy Outline Levels

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.

```
<define name="text-user-index-source-attr" combine="interleave">
    <optional>
        <attribute name="text:copy-outline-levels"
                     a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

7.7.2 User-Defined Index Entry Template

User index entry templates support entry elements for chapter number, page number, entry text, text spans, and tab stops.

```
<define name="text-user-index-entry-template">
    <element name="text:user-index-entry-template">
        <ref name="text-user-index-entry-template-attrs"/>
        <zeroOrMore>
            <choice>
                <ref name="text-index-entry-chapter"/>
                <ref name="text-index-entry-page-number"/>
                <ref name="text-index-entry-text"/>
                <ref name="text-index-entry-span"/>
                <ref name="text-index-entry-tab-stop"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated 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.

All `<text:outline-level>` elements that are contained in the same parent element must specify different outline levels.

```
<define name="text-user-index-entry-template-attrs" combine="interleave">
    <attribute name="text:outline-level">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

```
<define name="text-user-index-entry-template-attrs" combine="interleave">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

7.8 Alphabetical Index

An alphabetical index gathers its entries solely from index marks.

```
<define name="text-alphabetical-index">
    <element name="text:alphabetical-index">
        <ref name="sectionAttr"/>
        <ref name="text-alphabetical-index-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.8.1 Alphabetical Index Source

The `<text:alphabetical-index-source>` element specifies how the alphabetical index is generated.

```
<define name="text-alphabetical-index-source">
    <element name="text:alphabetical-index-source">
        <ref name="text-alphabetical-index-source-attrs"/>
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <zeroOrMore>
            <ref name="text-alphabetical-index-entry-template"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated 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
- Sort language, country and algorithm
- Index scope (see section 7.4.1)

  This attribute specifies whether the index applies to the entire document or only to the current chapter.
- Relative tab stop position (see section 7.4.1)

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <ref name="text-index-scope-attr"/>
    <ref name="text-relative-tab-stop-position-attr"/>
</define>
```

Ignore Case

The `text:ignore-case` attribute determines whether or not the capitalization of words is ignored. 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.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:ignore-case" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:main-entry-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

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.

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.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:alphabetical-separators" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Combining Entries

There are several options for dealing with the common situation where there are 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.
- The pages referenced by a combined entry can be formatted as:
  - 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

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:combine-entries" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:combine-entries-with-dash"
                   a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:combine-entries-with-pp" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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. If the value of this attribute is `true`, the keys are used as additional entries. If the value of this attribute is `false`, the keys are used as sub entries.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:use-keys-as-entries" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Capitalize Entries

The `text:capitalize-entries` attribute determines whether or not the entries in the index are to be capitalized.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:capitalize-entries" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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, multiple entries can be listed on a single line separated by a comma. 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.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:comma-separated" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Sort country, Language, and Algorithm

If index entries are to be sorted, these attributes can be used to specify the sorting. The attributes country and language specify the sorting locale. For some locales, there are multiple sorting algorithms in use. In this case, the algorithm attribute can be used to specify an algorithm by name.

```
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="fo:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
</define>
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="fo:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
<define name="text-alphabetical-index-source-attrs" combine="interleave">
    <optional>
        <attribute name="text:sort-algorithm">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

7.8.2 Auto Mark File

The alphabetical index supports a so-called auto mark file. Such a file contains a list of terms, and each occurrence of such a term is to be included in the alphabetical index. The alphabetical index mark file is declared as part of the text declarations (see section 4.8). The declaration element in an XLink, which points to the resource containing the list of terms.

```
<define name="text-alphabetical-index-auto-mark-file">
    <element name="text:alphabetical-index-auto-mark-file">
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <value>simple</value>
            </attribute>
        </optional>
    </element>
</define>
```

7.8.3 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.

```
<define name="text-alphabetical-index-entry-template">
    <element name="text:alphabetical-index-entry-template">
        <ref name="text-alphabetical-index-entry-template-attrs"/>
        <zeroOrMore>
            <choice>
                <ref name="text-index-entry-chapter"/>
                <ref name="text-index-entry-page-number"/>
                <ref name="text-index-entry-text"/>
                <ref name="text-index-entry-span"/>
                <ref name="text-index-entry-tab-stop"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated 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

```
<define name="text-alphabetical-index-entry-template-attrs"
        combine="interleave">
    <attribute name="text:outline-level">
        <choice>
            <value>1</value>
            <value>2</value>
            <value>3</value>
            <value>separator</value>
        </choice>
    </attribute>
</define>
```

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

```
<define name="text-alphabetical-index-entry-template-attrs"
        combine="interleave">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

7.9 Bibliography

A bibliography index gathers its entries from bibliography index marks. The `<text:bibliography>` element represents a bibliography.

```
<define name="text-bibliography">
    <element name="text:bibliography">
        <ref name="sectionAttr"/>
        <ref name="text-bibliography-source"/>
        <ref name="text-index-body"/>
    </element>
</define>
```

7.9.1 Bibliography Index Source

The `<text:bibliography-source>` element specifies how the bibliography is generated.

```
<define name="text-bibliography-source">
    <element name="text:bibliography-source">
        <optional>
            <ref name="text-index-title-template"/>
        </optional>
        <zeroOrMore>
```

```
            <ref name="text-bibliography-entry-template"/>
        </zeroOrMore>
    </element>
</define>
```

7.9.2 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.

```
<define name="text-bibliography-entry-template">
    <element name="text:bibliography-entry-template">
        <ref name="text-bibliography-entry-template-attrs"/>
        <zeroOrMore>
            <choice>
                <ref name="text-index-entry-span"/>
                <ref name="text-index-entry-tab-stop"/>
                <ref name="text-index-entry-bibliography"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<text:bibliography-entry-template>` elements are:

- Bibliography type
- Paragraph style

Bibliography Type

This attribute specifies to which type of bibliographical entry the template applies. This attribute must be unique among all `<text:bibliography-type>` elements within the same parent element.

```
<define name="text-bibliography-entry-template-attrs" combine="interleave">
    <attribute name="text:bibliography-type">
        <ref name="text-bibliography-types"/>
    </attribute>
</define>
```

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

```
<define name="text-bibliography-entry-template-attrs" combine="interleave">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

7.10 index source styles

Some indices 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. The `text:outline-levels` attribute determines at which outline level to list the index entries gathered from the respective paragraph styles. There can only be one `<text:index-source-style>` element for each outline level.

```
<define name="text-index-source-styles">
    <element name="text:index-source-styles">
        <attribute name="text:outline-level">
            <ref name="positiveInteger"/>
        </attribute>
        <zeroOrMore>
            <ref name="text-index-source-style"/>
        </zeroOrMore>
    </element>
</define>
```

7.10.1 Index source style

All paragraphs formatted using the style or class specified in the `<text:index-source-style>` element are included in the index.

```
<define name="text-index-source-style">
    <element name="text:index-source-style">
        <attribute name="text:style-name">
            <ref name="styleName"/>
        </attribute>
        <empty/>
    </element>
</define>
```

7.11 Index title template

The `<text:index-title-template>` element determines the style and content of the index title. There can only be one `<text:index-title-template>` element contained in a `<text:table-of-content-source>` element.

```
<define name="text-index-title-template">
    <element name="text:index-title-template">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <text/>
    </element>
</define>
```

7.12 Index Template Entries

There are eight types of index entries, as follows:

- Chapter information
- Entry text
- Page number
- Fixed string
- Bibliography information
- Tab stop
- Hyperlink start and end

7.12.1 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.

```
<define name="text-index-entry-chapter">
    <element name="text:index-entry-chapter">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <ref name="text-index-entry-chapter-attrs"/>
    </element>
</define>
```

Note: This element can only display the chapter number. To display the chapter name, the `<text:index-entry-text>` elements must be used.

Display Chapter Format

The `text:display` attribute displays either the chapter number, the chapter name, or both.

```
<define name="text-index-entry-chapter-attrs">
    <optional>
        <attribute name="text:display" a:defaultValue="number">
            <choice>
                <value>name</value>
                <value>number</value>
                <value>number-and-name</value>
            </choice>
        </attribute>
    </optional>
</define>
```

7.12.2 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.

```
<define name="text-index-entry-text">
    <element name="text:index-entry-text">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
    </element>
</define>
```

7.12.3 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.

```
<define name="text-index-entry-page-number">
    <element name="text:index-entry-page-number">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
    </element>
</define>
```

7.12.4 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. Unlike the `<text:span>` element, the `<text:index-entry-span>` element does not have any child elements.

```
<define name="text-index-entry-span">
    <element name="text:index-entry-span">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <text/>
    </element>
</define>
```

7.12.5 Bibliography Information

The `<text:index-entry-bibliography>` element introduces bibliography data into index entry templates.

```
<define name="text-index-entry-bibliography">
    <element name="text:index-entry-bibliography">
        <ref name="text-index-entry-bibliography-attrs"/>
    </element>
</define>
```

The attributes that may be associated with the `<text:index-entry-bibliography>` element are:

- `text:style-name` attribute
- `text:bibliography-data-field` attribute

Text Style Name

The `text:style-name` attribute determines the style for display of the entry.

```
<define name="text-index-entry-bibliography-attrs" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Bibliography Data Field Identifier

The `text:bibliography-data-field` attribute determines which part of the bibliography data field will be displayed.

```
<define name="text-index-entry-bibliography-attrs" combine="interleave">
    <attribute name="text:bibliography-data-field">
        <choice>
            <value>address</value>
            <value>annote</value>
            <value>author</value>
            <value>bibliography-type</value>
            <value>booktitle</value>
            <value>chapter</value>
            <value>custom1</value>
            <value>custom2</value>
            <value>custom3</value>
            <value>custom4</value>
            <value>custom5</value>
            <value>edition</value>
            <value>editor</value>
            <value>howpublished</value>
            <value>identifier</value>
            <value>institution</value>
            <value>isbn</value>
            <value>issn</value>
            <value>journal</value>
            <value>month</value>
            <value>note</value>
            <value>number</value>
            <value>organizations</value>
            <value>pages</value>
            <value>publisher</value>
            <value>report-type</value>
            <value>school</value>
            <value>series</value>
            <value>title</value>
            <value>url</value>
            <value>volume</value>
            <value>year</value>
        </choice>
    </attribute>
</define>
```

7.12.6 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.

```
<define name="text-index-entry-tab-stop">
    <element name="text:index-entry-tab-stop">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
        <ref name="text-index-entry-tab-stop-attrs"/>
    </element>
</define>
```

The attributes that may be associated 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.

```
<define name="text-index-entry-tab-stop-attrs" combine="interleave">
    <optional>
        <attribute name="style:leader-char">
            <ref name="character"/>
        </attribute>
    </optional>
</define>
```

Tab Type and Position

The style:type attribute specifies the tab stop type. 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. The style:position attribute specifies the position of the tab. 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.

```
<define name="text-index-entry-tab-stop-attrs" combine="interleave">
    <choice>
        <attribute name="style:type">
            <value>right</value>
        </attribute>
        <group>
            <attribute name="style:type">
                <value>left</value>
            </attribute>
            <attribute name="style:position">
                <ref name="length"/>
            </attribute>
        </group>
    </choice>
</define>
```

7.12.7 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.

```
<define name="text-index-entry-link-start">
    <element name="text:index-entry-link-start">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
    </element>
```

```
</define>
<define name="text-index-entry-link-end">
    <element name="text:index-entry-link-end">
        <optional>
            <attribute name="text:style-name">
                <ref name="styleNameRef"/>
            </attribute>
        </optional>
    </element>
</define>
```

7.12.8 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:outline-level="1"
            text:style-name="Contents 1">
            <text:index-entry-chapter text:display="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:outline-level="2"
            text:style-name="Contents 2">
            <text:index-entry-chapter text:display="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 stylename="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>
```

8 Tables

This chapter describes the table structure that is used for tables that are embedded within text documents and for spreadsheets.

8.1 Basic Table Model

The structure of OpenDocument tables is similar to the structure of [HTML4] or [XSL] tables, and like these tables, they can be nested.

The representation of tables is based on a grid of rows and columns. Rows take precedence over columns. The table is divided into rows and the rows are divided into cells. Each column includes a column description, but this description does not contain any cells.

Table rows may be empty, and different rows might contain a different number of table cells. This is not an error, but applications might resolve this in different ways. Spreadsheet applications typically operate on large tables that have a fixed application dependent row and column number, but may have an unused area. Only the used area of the table is saved in files. When loading a table with empty or incomplete rows into a spreadsheet application, empty rows typically introduce a default row (just as in an empty sheet), and incomplete rows are filled with empty cells (just like in an empty sheet).

All other applications typically have fixed size tables. Incomplete rows are basically rendered as if they had the necessary number of empty cells, and the same applies to empty rows. Empty cells typically occupy the space of an empty paragraph.

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.

8.1.1 Table Element

The table element is the root element for tables.

```
<define name="table-table">
    <element name="table:table">
        <ref name="table-table-attlist"/>
        <optional>
            <ref name="table-table-source"/>
        </optional>
        <optional>
            <ref name="office-dde-source"/>
        </optional>
        <optional>
            <ref name="table-scenario"/>
        </optional>
        <optional>
            <ref name="office-forms"/>
        </optional>
        <optional>
            <ref name="table-shapes"/>
        </optional>
        <ref name="table-columns-and-groups"/>
        <ref name="table-rows-and-groups"/>
    </element>
</define>
```

The content models for tables is rather complex. The details are explained in the section 8.2. For the moment, it can be assumed that table element's content are columns and row elements.

```
<define name="table-columns-and-groups">
    <oneOrMore>
        <choice>
            <ref name="table-table-column-group"/>
            <ref name="table-columns-no-group"/>
        </choice>
    </oneOrMore>
</define>

<define name="table-columns-no-group">
    <choice>
        <group>
            <ref name="table-columns"/>
            <optional>
                <ref name="table-table-header-columns"/>
                <optional>
                    <ref name="table-columns"/>
                </optional>
            </optional>
        </group>
        <group>
            <ref name="table-table-header-columns"/>
            <optional>
                <ref name="table-columns"/>
            </optional>
        </group>
    </choice>
</define>

<define name="table-columns">
    <choice>
        <ref name="table-table-columns"/>
        <oneOrMore>
                <ref name="table-table-column"/>
        </oneOrMore>
    </choice>
</define>

<define name="table-rows-and-groups">
    <oneOrMore>
        <choice>
            <ref name="table-table-row-group"/>
            <ref name="table-rows-no-group"/>
        </choice>
    </oneOrMore>
</define>

<define name="table-rows-no-group">
    <choice>
        <group>
            <ref name="table-rows"/>
            <optional>
                <ref name="table-table-header-rows"/>
                <optional>
                    <ref name="table-rows"/>
                </optional>
            </optional>
        </group>
        <group>
```

```
            <ref name="table-table-header-rows"/>
            <optional>
                <ref name="table-rows"/>
            </optional>
        </group>
    </choice>
</define>

<define name="table-rows">
    <choice>
        <ref name="table-table-rows"/>
        <oneOrMore>
            <optional>
                <ref name="text-soft-page-break"/>
            </optional>
            <ref name="table-table-row"/>
        </oneOrMore>
    </choice>
</define>
```

Table Name

The `table:name` attribute specifies the name of a table.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Table Style

The `table:style-name` attribute references a table style, i.e., an `<style:style>` element of type “table”. The table style describes the formatting properties of the table, such as width and background color. The table style can be either an automatic or common style.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Example: Table Style

```
<style:style style:name="Table 1" style:family="table">
    <style:table-properties style:width="12cm"
     fo:background-color="light-grey"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
    ...
</table:table>
```

Protected

The `table:protected` attribute specifies whether or not a table is protected from editing. If the table is protected, the `table:protection-key` attribute can specify a password to prevent a user from resetting the protection flag to enable editing. 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.

To avoid saving the password directly into the XML file, only a hash value of the password is stored within the `table:protection-key` attribute.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:protected" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:protection-key">
            <text/>
        </attribute>
    </optional>
</define>
```

Print

The `table:print` attribute specifies if a table is printed. It takes a Boolean value. If its value is `true`, the table is printed, if its value is `false`, the table is not printed. The default value is `true`. The `table:print` attribute will be overwritten by the `table:display` attribute described in section 15.8.14. That is, if the table is not displayed, it also will not be printed.

If the table is printed, the table range that actually is printed can be specified by `table:print-ranges` attribute (see section 8.1.1:Print Ranges). If this attribute is not existing, the used area of the table will be printed.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:print" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Print Ranges

The `table:print-ranges` attribute specifies the print ranges of the table, i.e., the cells that should be printed. It contains a list of cell addresses or cell range addresses as described in section 8.3.1.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:print-ranges">
            <ref name="cellRangeAddressList"/>
        </attribute>
    </optional>
</define>
```

Soft Page Breaks

The `<text:soft-page-break>` element represents a soft page break between two table rows. It may appear in front of `<table:table-row>` elements.

See section 2.3.1:Use Soft Page BreaksUse Soft Page Breaks for details regarding soft page breaks.

8.1.2 Table Row

The `<table:table-row>` element represents a row in a table. It content are elements that specify the cells of the table row.

The `<table:table-row>` element is similar to the [XSL] `<fo:table-row>` element.

```
<define name="table-table-row">
    <element name="table:table-row">
        <ref name="table-table-row-attlist"/>
        <oneOrMore>
            <choice>
                <ref name="table-table-cell"/>
                <ref name="table-covered-table-cell"/>
            </choice>
        </oneOrMore>
    </element>
</define>
```

Number of Rows Repeated

The `table:number-rows-repeated` attribute specifies the number of rows to which a row element applies. If two or more rows are adjoining, and have the same content and properties, and do not contain vertically merged cells, they may be described by a single `<table:table-row>` element that has a `table:number-rows-repeated` attribute with a value greater than 1.

```
<define name="table-table-row-attlist" combine="interleave">
    <optional>
        <attribute name="table:number-rows-repeated" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Row Style

A table row style stores the formatting properties of a table row, such as height and background color. A row style is defined by a `<style:style>` element with a family attribute value of `table-row`. The table row style can be either an automatic or a common style. It is referenced by the table row's `table:style-name` attribute.

```
<define name="table-table-row-attlist" combine="interleave">
    <optional>
        <attribute name="table:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Default Cell Style

The `table:default-cell-style-name` attribute specifies a default cell style. Cells contained in the row that don't have a `table:style-style` name attribute use this default cell style.

The attribute is applied to cells that are defined by a `<table:table-cell>` element. It is typically not applied to table cells that spreadsheet application may display in addition to those defined in the document.

```
<define name="table-table-row-attlist" combine="interleave">
    <optional>
        <attribute name="table:default-cell-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Visibility

The `table:visibility` attribute specifies whether the row is visible, filtered, or collapsed. Filtered and collapsed rows are not visible. Filtered rows are invisible, because a filter is applied to the table that does not select the table row. Collapsed rows have been made invisible by invisible in the UI directly.

```
<define name="table-table-row-attlist" combine="interleave">
    <optional>
        <attribute name="table:visibility" a:defaultValue="visible">
            <ref name="table-visibility-value"/>
        </attribute>
    </optional>
</define>

<define name="table-visibility-value">
    <choice>
        <value>visible</value>
        <value>collapse</value>
        <value>filter</value>
    </choice>
</define>
```

Example: Table with three rows and three columns

This example shows the OpenDocument code 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:table-properties style:width="12cm"
        fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
    <style:table-column-properties style:column-width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
    <style:table-column-properties style:column-width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
    <style:table-column-properties style:column-width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
    <style:table-row-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>
```

8.1.3 Table Cell

The `<table:table-cell>` and `<table:covered-table-cell>` elements specify the content of a table cells. They are contained in table row elements. A table cell can contain paragraphs and other text content as well as sub tables. Table cells may be empty.

The `<table:table-cell>` element is very similar to the table cell elements of [XSL] and [HTML4], and the rules regarding cells that span several columns or rows that exist in HTML and XSL apply to the OpenDocument specification as well. This means that there are no `<table:table-cell>` elements in the row/column grid for positions that are covered by a merged cell, that is, that are covered by a cell that spans several columns or rows. The `<table:covered-table-cell>` element exists to be able to specify cells for such positions . It has to appear wherever a position in the row/column grid is covered by a cell that spans several rows or columns. Its position in the grid is calculated by a assuming a column and row span of 1 for all cells regardless whether they are specified by a `<table:table-cell>` or a `<table:covered-table-cell>` element. The `<table:covered-table-cell>` is especially used by spreadsheet applications, where it is a common use case that a covered cell contains content.

```
<define name="table-table-cell">
    <element name="table:table-cell">
        <ref name="table-table-cell-attlist"/>
        <ref name="table-table-cell-attlist-extra"/>
        <ref name="table-table-cell-content"/>
    </element>
</define>

<define name="table-covered-table-cell">
    <element name="table:covered-table-cell">
        <ref name="table-table-cell-attlist"/>
        <ref name="table-table-cell-content"/>
    </element>
</define>

<define name="table-table-cell-content">
    <optional>
        <ref name="table-cell-range-source"/>
    </optional>
    <optional>
        <ref name="office-annotation"/>
    </optional>
    <optional>
        <ref name="table-detective"/>
    </optional>
    <zeroOrMore>
        <ref name="text-content"/>
    </zeroOrMore>
</define>
```

Number of Cells Repeated

The `table:number-columns-repeated` attribute specifies the number of successive columns in which a cell is repeated. It can be used to describe two or more adjoining cells with a single cell element, 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, a `table:number-columns-repeated` attribute must be used to specify the number of successive columns in which the cell is repeated. This attribute is specified with either the `<table:table-cell>` element or the `<table:covered-table-cell>` element.

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:number-columns-repeated" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Number of Rows and Columns Spanned

These attributes specify the number of rows and columns that a cell spans. These attributes can be used with the `<table:table-cell>` element only.

When a cell covers another cell because of a column or row span value greater than one, a `<table:covered-table-cell>` element must appear in the table to represent the covered cell.

```
<define name="table-table-cell-attlist-extra" combine="interleave">
    <optional>
        <attribute name="table:number-columns-spanned" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:number-rows-spanned" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Cell Style

A table cell style stores the formatting properties of a cell, such as the following:

- Background color
- Number format
- Vertical alignment
- Borders

The table cell style can be either an automatic or a common style. The style is specified with a `table:style-name` attribute. If a cell does not have a cell style assigned, the application checks if a the current row has a default cell style assigned. If the current row does not have a default cell assigned style as well, the application checks if the current column has a default cell style assigned.

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Cell Content Validation

The `table:content-validation-name` attribute specifies if a cell contains a validity check. The value of this attribute is the name of a `<table:content-validation>` element. If the attribute is not present, the cell may have arbitrary content.

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:content-validation-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

See section 8.5.3 for more information on cell content validation and the `<table:cell-content-validation>` element.

Formula

Formulas allow calculations to be performed within table cells. Every formula should begin with a namespace prefix specifying the syntax and semantics used within the formula. Typically, the formula itself begins with an equal (=) sign and can include the following components:

- Numbers.
- Text.
- Named ranges.
- Operators.
- Logical operators.
- Function calls.
- Addresses of cells that contain numbers. The addresses can be relative or absolute, see section 8.3.1. Addresses in formulas start with a "[" and end with a "]". See sections 8.3.1 and 8.3.1 for information about 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`". The function is "`sum`". 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 formula shown above:

```
=sum([.A1];[.A2];[.A3];[.A4];[.A5])
```

The result of this formula is the same. The components used in the formula depend on the application being used.

The `table:formula` attribute contains a formula for a table cell.

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:formula">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

In addition to this, the calculated value of the formula is available as well. One of the following attributes represents the current value of the cell:

- `office:value`
- `office:date-value`
- `office:time-value`
- `office:boolean-value`
- `office:string-value`

Matrix

When an application is performing spreadsheet calculations, a connected range of cells that contains values is called 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. To use a matrix in a formula, include the cell range address of the matrix in the formula. In a matrix formula, only 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.

```
<define name="table-table-cell-attlist-extra" combine="interleave">
    <optional>
        <attribute name="table:number-matrix-columns-spanned">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:number-matrix-rows-spanned">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Value Type

The `table:value-type` attribute specifies the type of value that can appear in a cell. It may contain one of the following values:

- `float`, `percentage` or `currency` (numeric types)

- `date`
- `time`
- `boolean`
- `string`

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <ref name="common-value-and-type-attlist"/>
    </optional>
</define>
```

Cell Current Numeric Value

The `office:value` attribute specifies the current numeric value of a cell. This attribute is only evaluated for cells that contain the following data types:

- `float`
- `percentage`
- `currency`

Cell Current Currency

The `tableoffice:currency` attribute specifies the current currency value of a cell. The value of this attribute is usually currency information such as DEM or EUR. This attribute is only evaluated for cells whose data type is `currency`.

Cell Current Date Value

The `office:date-value` attribute specifies the current date value of a cell. This attribute is only evaluated for cells whose data type is `date`.

Some application support date and time values in addition to dates.

Cell Current Time Value

The `office:time-value` attribute specifies the current time value of a cell. This attribute is only evaluated for cells whose data type is `time`.

Cell Current Boolean Value

The `office:boolean-value` attribute specifies the current Boolean value of a cell. This attribute is only evaluated for cells whose data type is `boolean`.

Cell Current String Value

The `office:string-value` attribute specifies the current string value of a cell. This attribute is only evaluated for cells whose data type is `string`.

Table Cell Protection

The `table:protected` attribute protects the table cells. Users can not edit the content of a cell that is marked as protected.

```
<define name="table-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:protect" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

This attribute is not related to the `table:protected` attribute for table elements (see section 8.1.1) and the `table:cell-protect` attribute for table cell styles (see section 15.11.14).

8.2 Advanced Table Model

8.2.1 Column Description

Every column in a table has a column description element `<table:table-column>`. It is similar to the [XSL] `<fo:table-column>` element, and its primary use is to reference a table column style that for instance specifies the table column's width.

```
<define name="table-table-column">
    <element name="table:table-column">
        <ref name="table-table-column-attlist"/>
        <empty/>
    </element>
</define>
```

Number of Columns Repeated

The `table:number-columns-repeated` attribute specifies the number of columns to which a column description applies. If two or more columns are adjoining, and have the same properties, this attribute allows to describe them with a single `<table:table-column>` element.

```
<define name="table-table-column-attlist" combine="interleave">
    <optional>
        <attribute name="table:number-columns-repeated" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Column Style

A table column style stores the formatting properties of a table column, such as width and background color. It is specified by a `<style:style>` element with a family attribute value of `table-column` and can be either an automatic or a common style. The style of a column is specified using a `table:style-name` attribute.

```
<define name="table-table-column-attlist" combine="interleave">
    <optional>
        <attribute name="table:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
```

```
</define>
```

Visibility

The `table:visibility` attribute specifies whether the column is visible, filtered, or collapsed. See section 8.1.2 for more details.

```
<define name="table-table-column-attlist" combine="interleave">
    <optional>
        <attribute name="table:visibility" a:defaultValue="visible">
            <ref name="table-visibility-value"/>
        </attribute>
    </optional>
</define>
```

Default Cell Style

The `table:default-cell-style-name` attribute specifies the default cell style. Cells that don't have a `table:style-style` name attribute use this style when there is no default cell style specified for the cell's row as well.

The attribute is applied to cells that are defined by a `<table:table-cell>` element. It is typically not applied to table cells that spreadsheet application may display in addition to those defined in the document.

```
<define name="table-table-column-attlist" combine="interleave">
    <optional>
        <attribute name="table:default-cell-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Example: Table with three columns

This example shows the OpenDocument code for a table with three columns.

```
<style:style style:name="Table 1" style:family="table">
    <style:table-properties style:width="12cm"
        fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
    <style:table-column-properties style:column-width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
    <style:table-column-properties style:column-width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
    <style:table-column-properties style:column-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>
```

8.2.2 Header Columns

For accessibility purposes, header information is needed. Therefore, any columns designated as headers by the author must be tagged as such by encapsulating them within a `<table:table-header-columns>` element. Using style information only to designate header columns is insufficient.

If a table does not fit on a single page, table columns that are included in a `<table:table-header-columns>` element are automatically repeated on every page. A table must not contain more than one `<table:table-header-columns>` element, and a `<table:table-columns>` must not follow another `<table:table-columns>` element, with the only exception of tables that contain grouped columns (see 8.2.3). Such tables may contain more than one `<table:table-header-columns>` element, provided that they are contained in different column groups and the columns contained in the elements are adjacent.

Applications that do not support header columns have to process header column descriptions the same way as non header column descriptions.

The `<table:table-header-columns>` and `<table:table-columns>` element are very similar to [HTML4]'s `<THEAD>` and `<TBODY>` elements for rows.

```
<define name="table-table-header-columns">
    <element name="table:table-header-columns">
        <oneOrMore>
            <ref name="table-table-column"/>
        </oneOrMore>
    </element>
</define>

<define name="table-table-columns">
    <element name="table:table-columns">
        <oneOrMore>
            <ref name="table-table-column"/>
        </oneOrMore>
    </element>
</define>
```

8.2.3 Column Groups

Adjacent table columns can be grouped with the `<table:table-column-group>` element. Every group can contain a new group, columns, and column headers. A column group can be visible or hidden. Column groups can for instance used by spreadsheet applications to group columns that are summarized, so that the individual columns that contribute to the sum can be made invisible easily, but the sum remains visible.

If a set of header columns and a column group overlap, the header column group breaks the column header set. That is, the `<table:table-column-group>` may contain `<table:table-header-columns>` elements, but not vice versa.

```
<define name="table-table-column-group">
    <element name="table:table-column-group">
        <ref name="table-table-column-group-attlist"/>
        <ref name="table-columns-and-groups"/>
    </element>
</define>
```

Display

The `table:display` attribute specifies whether or not the group is visible.

```
<define name="table-table-column-group-attlist" combine="interleave">
    <optional>
        <attribute name="table:display" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.2.4 Header Rows

For accessibility purposes, header information is needed. Therefore, any rows designated as headers by the author must be tagged as such by encapsulating them within a `<table:table-header-rows>` element. Using style information only to designate header rows is insufficient.

If a table does not fit on a single page, table rows that are included in a `<table:table-header-rows>` element are automatically repeated on every page. A table must not contain more than one `<table:table-header-rows>` element. The one exception to this is a table that contains grouped rows (see 8.2.5). Such a table may contain more than one `<table:table-header-rows>` element, provided that they are contained in different row groups and the rows contained in the elements are adjacent.

Applications that do not support header rows have to process header rows the same way as non header rows.

The `<table:table-header-rows>` and `<table:table-rows>` element are very similar to [HTML4]'s `<THEAD>` and `<TBODY>` elements.

```
<define name="table-table-header-rows">
    <element name="table:table-header-rows">
        <oneOrMore>
            <optional>
                <ref name="text-soft-page-break"/>
            </optional>
            <ref name="table-table-row"/>
        </oneOrMore>
    </element>
</define>

<define name="table-table-rows">
    <element name="table:table-rows">
        <oneOrMore>
            <optional>
                <ref name="text-soft-page-break"/>
            </optional>
            <ref name="table-table-row"/>
        </oneOrMore>
    </element>
</define>
```

8.2.5 Row Groups

Adjacent table rows can be grouped with the `<table:table-row-group>` element. Every group can contain a new group, rows, and row headers. A row group can be visible or hidden. Row groups can for instance used by spreadsheet applications to group rows that are summarized, so that the individual rows that contribute to the sum can be made invisible easily, but the sum remains visible.

If a set of header rows and a row group overlap, the header row group breaks the row header set. That is, the <table:table-row-group> may contain <table:table-header-rows> elements, but not vice versa.

```
<define name="table-table-row-group">
    <element name="table:table-row-group">
        <ref name="table-table-row-group-attlist"/>
        <ref name="table-rows-and-groups"/>
    </element>
</define>
```

Display

The `table:display` attribute specifies whether or not the group is visible.

```
<define name="table-table-row-group-attlist" combine="interleave">
    <optional>
        <attribute name="table:display" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.2.6 Subtables

If a table cell only contains a single table but no paragraphs or other content, this table can be specified as subtable. It then occupies the whole cell and no other content can appear in this cell.

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.

A nested table is turned into a subtable with the attribute `table:is-subtable` that is attached to the table element. A nested table that is not a specified to be a subtable 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.

```
<define name="table-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:is-sub-table" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Example of Representation of subtable

In the OpenDocument schema, this table can be represented in either of the ways detailed in Sample 1 and Sample 2.

A1	***B1***	***C1***
A2	.B2.A1	.B2.B1
	.B2.A2	

Sample 1

Using cells that span several rows, the table is specified as follows:

```
<style:style style:name="Table 1" style:family="table">
   <style:table-properties style:width="12cm"
```

```
                      fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
    <style:table-column-properties style:column-width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
    <style:table-column-properties style:column-width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
    <style:table-column-properties style:column-width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
    <style:table-row-properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
    <style:table-cell-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.A1
                </text:p>
            </table:table-cell>
            <table:table-cell>
                <text:p text:style="Table Body">
                    .B2.B1
                </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.A2
            </text:p>
            </table:table-cell>
            <table:covered-table-cell/>
        </table:table-row>
    </table:table-rows>
</table:table>
```

Sample 2

Using sub tables, the table is specified as follows:

```
<style:style style:name="Table 1" style:family="table">
    <style:table-properties fo:width="12cm" fo:background-color="light-
grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
    <style:table-column-properties style:column-width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
    <style:table-column-properties style:column-width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
    <style:table-column-properties style:column-width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
    <style:table-row-properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
    <style:table-cell-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:table is-subtable="true">
                    <table:table-columns>
                        <table:table-column table:style-name="Col2"/>
                        <table:table-column table:style-name="Col3"/>
                    </table:table-columns>
                    <table:rows>
                        <table:row>
                            <table:table-cell>
                                <text:p text:style="Table Body">
                                    .B2.A1
                                </text:p>
                            </table:table-cell>
                            <table:table-cell>
                                <text:p text:style="Table Body">
                                    .B2.B1
                                </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.A2
                                </text:p>
                            </table:table-cell>
                            <table:covered-table-cell/>
                        </table:table-row>
                    </table:table-rows>
                </table:table>
            </table:table-cell>
            <table:covered-table-cell/>
        </table:table-row>
    </table:table-rows>
</table:table>
```

8.3 Advanced Tables

8.3.1 Referencing Table Cells

To reference table cells so called cell addresses are used. 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.

This means that 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, in a table with the name `SampleTable` the cell in column 34 and row 16 is referenced by the cell address `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 blanks, dots (.) or apostrophes ('), the name must be quoted with apostrophes ('). Any apostrophes in the name must be escaped by doubling ('').

E.g. `'Tom''s Table'.A1` for the cell A1 in the table named Tom's Table.

Absolute and relative cell addressing

Cells can be referenced by using either absolute addresses or relative addresses. When an operation is performed on a table cell, for example when a formula is copied, absolute cell references do not change; In contrast to this, relative cell references are adapted to the address of target cell of the copy operation. The previous example uses relative addressing.

To create an absolute address, a dollar sign ($) has to be placed before each table name, column reference, and row reference. For example, the absolute address of the previous example is `$SampleTable.$AH$16`. Absolute and relative references can be mixed within a single cell address. For example, `SampleTable.AH$16` refers to a relative table and column, but to an absolute row. Absolute addresses must contain a table name. The differentiation between absolute and relative addressing is only necessary in some situations. Where a differentiation is not required, a cell reference without the dollar signs is used.

```
<define name="cellAddress">
    <data type="string">
        <param name="pattern">($?([^\. ']+|'([^']|'')+'))?\.$?[A-Z]+$?[0-
9]+</param>
    </data>
</define>
```

Cell Range Address

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.

A cell range address references a cell range. It is constructed as follow:

1. The address of the cell at the top left of the range.
2. A colon (:).
3. The address of the cell at the bottom right of the range.

For example, the address `.A1:.B2` references the cell range of cells from column 1 and row 1 to column 2 and row 2. The smallest range one can specify is a single cell. In this case, the range address is the same as the cell address.

```
<define name="cellRangeAddress">
    <data type="string">
        <param name="pattern">($?([^\. ']+|'([^']|'')+'))?\.$?[A-Z]+$?[0-
9]+(:($?([^\. ']+|'([^']|'')+'))?\.$?[A-Z]+$?[0-9]+)?</param>
    </data>
</define>
```

Cell Range Address List

A cell range address list is a list of cell ranges and cell addresses. Each item in the list is separated by a space. If table names used in the list contain a blank character, the table name has to be quoted within apostrophes (').

```
<define name="cellRangeAddressList">
    <!-- Value is a space separated list of "cellRangeAddress" patterns -->
    <data type="string"/>
</define>
```

8.3.2 Linked Tables

If a table is linked to an original table, the information about the source table is contained in a `<table:table-source>` element. The attributes that may be associated with the `<table:table-source>` element are:

- Mode
- Table name
- URL
- Filter name
- Filter options
- Refresh delay

```
<define name="table-table-source">
    <element name="table:table-source">
        <ref name="table-table-source-attlist"/>
        <ref name="table-linked-source-attlist"/>
        <empty/>
    </element>
</define>
```

Mode

The `table:mode` attribute specifies what data should be copied from the source table to the destination table. If the attribute's value is “copy-all” formulas and styles are copied. If the attribute's value is “copy-results-only”, only formula results and non calculated cell content will be copied.

```
<define name="table-table-source-attlist" combine="interleave">
    <optional>
        <attribute name="table:mode" a:defaultValue="copy-all">
            <choice>
                <value>copy-all</value>
                <value>copy-results-only</value>
            </choice>
        </attribute>
    </optional>
```

```
</define>
```

Table Name

The `table:table-name` attribute specifies the name of the table in the original document. If the table name is not specified, the first table in the document is used.

```
<define name="table-table-source-attlist" combine="interleave">
    <optional>
        <attribute name="table:table-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

URL

The original table is specified by a an XLink, where the `xlink:href` attribute specifies the URL of the document containing the original table.

```
<define name="table-linked-source-attlist" combine="interleave">
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <value>simple</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate" a:defaultValue="onRequest">
            <value>onRequest</value>
        </attribute>
    </optional>
    <attribute name="xlink:href">
        <ref name="anyURI"/>
    </attribute>
</define>
```

Filter Name

The `table:filter-name` attribute specifies the file type of the document containing the original table. The value of this attribute is application-specific.

```
<define name="table-linked-source-attlist" combine="interleave">
    <optional>
        <attribute name="table:filter-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Filter Options

The `table:filter-options` attribute specifies optional settings about the file type. The value of this attribute is application-specific.

```
<define name="table-linked-source-attlist" combine="interleave">
    <optional>
        <attribute name="table:filter-options">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Refresh Delay

The `table:refresh-delay` attribute specifies the time delay between refresh actions for the linked table.

```
<define name="table-linked-source-attlist" combine="interleave">
    <optional>
        <attribute name="table:refresh-delay">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

8.3.3 Scenario Tables

A scenario is an area of a table where data from other, so called scenario tables, is linked to temporarily. If several scenarios are defined for the same area, an user might choose between the scenarios. Whether a scenario table is visible itself is controlled by table's style. Only one scenario table can be active per table.

A table that contains a `<table:scenario>` 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 preceding the scenario table. If a scenario table is existing for a table, a scenario is displayed on that table automatically. These means the the existence of a scenario table implies the existence of a scenario.

The attributes that may be associated with this element are:

- Scenario Ranges
- Is Active
- Display Border
- Border Color
- Copy Back
- Copy Styles
- Copy Formulas
- Comment
- Protected

```
<define name="table-scenario">
    <element name="table:scenario">
        <ref name="table-scenario-attlist"/>
        <empty/>
    </element>
</define>
```

Scenario Ranges

The `table:scenario-ranges` attribute specifies the table range that is displayed as a scenario. The value of this attribute is a list of cell range addresses.

```
<define name="table-scenario-attlist" combine="interleave">
    <attribute name="table:scenario-ranges">
        <ref name="cellRangeAddressList"/>
    </attribute>
</define>
```

Is Active

The `table:is-active` attribute specifies whether or not the scenario that belongs to the scenario table is active.

```
<define name="table-scenario-attlist" combine="interleave">
    <attribute name="table:is-active">
        <ref name="boolean"/>
    </attribute>
</define>
```

Display Border

The `table:display-border` attribute specifies whether or not to display a border around the scenario that belongs to the scenario table.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:display-border" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Border Color

The `table:border-color` attribute specifies the color of the border that is displayed around the scenario that belongs to the scenario table.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:border-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

Copy Back

The `table:copy-back` attribute specifies whether or not data is copied back into the scenario table if another scenario is activated.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:copy-back" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Copy Styles

The `table:copy-styles` attribute specifies whether or not styles are copied from the scenario table to the destination table together with the data.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:copy-styles" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Copy Formulas

The `table:copy-formulas` attribute specifies whether or not formulas are copied from the scenario table to the destination table. 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.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:copy-formulas" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Comment

The `table:comment` attribute contains a comment about the scenario.

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:comment">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Protected

The `table:protected` attribute specifies whether or not the data that is displayed within the scenario is protected from being edited. The attribute is only evaluated if the table on which the scenario displayed is also protected (see section 8.1.1).

```
<define name="table-scenario-attlist" combine="interleave">
    <optional>
        <attribute name="table:protected">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.3.4 Shapes

The `<table:shapes>` element contains all graphic shapes with an anchor on the table this element is a child of. It is a container element and does not have any associated attributes.

```
<define name="table-shapes">
    <element name="table:shapes">
        <oneOrMore>
            <ref name="shape"/>
        </oneOrMore>
    </element>
</define>
```

8.4 Advanced Table Cells

8.4.1 Linked Table Cells

A cell range can be linked to a database range or named range of another file. In this case the information about the original database range or named range is contained in a `<table:cell-range-source>` element that is contained in the element of the first cell of the range. The attributes that may be associated with this element are:

- Name
- Last size
- URL
- Filter name
- Filter options
- Refresh delay

```
<define name="table-cell-range-source">
    <element name="table:cell-range-source">
        <ref name="table-table-cell-range-source-attlist"/>
        <ref name="table-linked-source-attlist"/>
        <empty/>
    </element>
</define>
```

Name

The `table:name` attribute specifies the name of the source database range or named range.

```
<define name="table-table-cell-range-source-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

Last Size

The `table:last-column-spanned` and `table:last-row-spanned` attributes specify the last known size of the range. If the size of the range is changed since the last operation, the values of these attributes are incorrect.

```
<define name="table-table-cell-range-source-attlist" combine="interleave">
    <attribute name="table:last-column-spanned">
        <ref name="positiveInteger"/>
    </attribute>
    <attribute name="table:last-row-spanned">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

URL, Filter Name, Filter Options and Refresh Delay

The attributes `xlink:href`, `xlink:type`, `xlink:actuate`, `table:filter-name` and `table:filter-options` are the same as for linked tables. See section 8.3.2 for details.

8.4.2 Cell Annotation

The OpenDocument format allows annotation to appear within table cells. See section 12.1 for details on annotations.

8.4.3 Detective

The `<table:detective>` element has two purposes. One the one hand, it contains information about relations that exist between table cells because of formulas and that should be highlighted in the UI. On the other hand, the element contains information about cells that are highlighted currently in the UI either because of the relations mentioned above or because of error conditions.

```
<define name="table-detective">
    <element name="table:detective">
        <zeroOrMore>
            <ref name="table-highlighted-range"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="table-operation"/>
        </zeroOrMore>
    </element>
</define>
```

The elements that can be contained in the `<table:detective>` element are:

- Detective Operation
- Highlighted range

8.4.4 Detective Operation

The `<table:operation>` element specifies that certain relations that exist between the cell the element is a child of and other cells should be made visible or invisible in the UI. One and the same detective operation can be applied multiple times to the same cell. In this case, the second operation is applied to the resulting cells of the first operation and so on. This means that an operation not necessarily is applied to the cell the operation is defined in, but also to other cells, and that it therefor can interact with operations defined in other cells. This especially applies to operations that make relations invisible. To get a determinate behavior, operations have an index and are applied in the order of that index. The attributes associated with the `<table:operation>` element are:

- Name
- Index

```
<define name="table-operation">
    <element name="table:operation">
        <ref name="table-operation-attlist"/>
        <empty/>
    </element>
```

```
</define>
```

Name

The `table:name` attribute specifies the name of the detective operation. Possible names are `trace-dependents` , `remove-dependents`, `trace-precedents`, `remove-precedents` and `trace-errors`. `trace-dependents` and `remove-dependents` displays or hides cells that use the value of the current cell in their formula. `Trace-precedents` and `remove-precedents` displays or hides cells whose value is used in the formula of the current cell. `Trace-errors` displays cells that cause an error while calculating the result of the current cell's formula.

```
<define name="table-operation-attlist" combine="interleave">
    <attribute name="table:name">
        <choice>
            <value>trace-dependents</value>
            <value>remove-dependents</value>
            <value>trace-precedents</value>
            <value>remove-precedents</value>
            <value>trace-errors</value>
        </choice>
    </attribute>
</define>
```

Index

The `table:index` attribute specifies the the order in which detective operations are applied.

```
<define name="table-operation-attlist" combine="interleave">
    <attribute name="table:index">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

8.4.5 Highlighted Range

The `<table:highlighted-range>` element specifies a cell range that is highlighted in the UI either because of detective operations described above or because it contains an error or invalid data.

The information contained in this element is not guaranteed to be up to date but reflects the state that at the time the detective operations or error conditions have been calculated.

The attributes that can be associated with the `<table:highlighted-range>` element are:

- Cell Range Address
- Direction
- Contains Error
- Marked Invalid

```
<define name="table-highlighted-range">
    <element name="table:highlighted-range">
        <choice>
            <group>
                <ref name="table-highlighted-range-attlist"/>
            </group>
            <group>
                <ref name="table-highlighted-range-attlist-invalid"/>
            </group>
        </choice>
        <empty/>
    </element>
</define>
```

Cell Range Address

The `table:cell-range-address` attribute contains the address of a range that is highlighted currently.

```
<define name="table-highlighted-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:cell-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

Direction

The `table:direction` attribute specifies the direction of the relation between this cell and the highlighted range. The direction for instance might be visualized by an arrow.

```
<define name="table-highlighted-range-attlist" combine="interleave">
    <attribute name="table:direction">
        <choice>
            <value>from-another-table</value>
            <value>to-another-table</value>
            <value>from-same-table</value>
        </choice>
    </attribute>
</define>
```

Contains Error

The `table:contains-error` attribute specifies whether or not the cell range contains an error.

```
<define name="table-highlighted-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:contains-error" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Marked Invalid

The `table:marked-invalid` attribute specifies whether or not the current cell is marked invalid. This attribute cannot be used together with any other attributes.

```
<define name="table-highlighted-range-attlist-invalid" combine="interleave">
    <attribute name="table:marked-invalid">
        <ref name="boolean"/>
    </attribute>
</define>
```

8.5 Spreadsheet Document Content

8.5.1 Document Protection

The structure of a spreadsheet document may be protected by using the `table:structure-protected` attribute, so that users can not insert, delete, move or rename the tables in the document. The optional `table:protection-key` attribute may be used to specify a password that prevents users from resetting the table protection flag to allow editing. To avoid saving the password directly into the XML file, only a hash value of the password is stored.

```
<define name="office-spreadsheet-attlist" combine="interleave">
    <optional>
        <attribute name="table:structure-protected" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:protection-key">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

8.5.2 Calculation Settings

Spreadsheet documents contain settings that affect 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.

```
<define name="table-calculation-settings">
    <element name="table:calculation-settings">
        <ref name="table-calculation-setting-attlist"/>
        <optional>
            <ref name="table-null-date"/>
        </optional>
        <optional>
            <ref name="table-iteration"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<table:calculation-settings>` element are:

- Case sensitive
- Precision as shown
- Search criteria must apply to whole cell
- Automatic find labels
- Use regular expression
- Null year
- Null date
- Iteration

Case Sensitive

The `table:case-sensitive` attribute specifies whether or not to distinguish between upper and lower case when comparison operators are applied to cell content.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:case-sensitive" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Precision as Shown

The `table:precision-as-shown` attribute specifies whether to perform a calculation using the rounded values displayed in the spreadsheet or using all of the digits in a number. If the value of this attribute is `true`, calculation are performed using the rounded values displayed in the spreadsheet. If the value of this attribute is `false`, calculations are performed using all of the digits in the number, but the result is still displayed as a rounded number.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:precision-as-shown" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Search Criteria Must Apply to Whole Cell

The `table:search-criteria-must-apply-to-whole-cell` attribute specifies whether or not the specified search criteria, according to the regular expression used, must apply to the entire cell contents.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:search-criteria-must-apply-to-whole-cell"
                    a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Automatic Find Labels

The `table:automatic-find-labels` attribute specifies whether or not to automatically find the labels of rows and columns.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:automatic-find-labels" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Use Regular Expressions

The `table:use-regular-expressions` attribute specifies whether regular expressions are enabled for character string comparisons and when searching.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:use-regular-expressions"
                     a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Null Year

The `table:null-year` attribute specifies the start year for year values that contain only two digits. All two digit year values are interpreted as a year that equals or follows the start year.

```
<define name="table-calculation-setting-attlist" combine="interleave">
    <optional>
        <attribute name="table:null-year" a:defaultValue="1930">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Null Date

The `<table:null-date>` element specifies the null date. The null date is the date that results in the value "0" if a date value is converted into a numeric value. The null date is specified in the element's `table:date-value` attribute. Commonly used values are `12/30/1899`, `01/01/1900`, and `01/01/1904`

```
<define name="table-null-date">
    <element name="table:null-date">
        <optional>
            <attribute name="table:value-type" a:defaultValue="date">
                <ref name="valueType"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="table:date-value"
                         a:defaultValue="1899-12-30">
                <ref name="date"/>
            </attribute>
        </optional>
        <empty/>
    </element>
</define>
```

Iteration

The `<table:iteration>` element enables formulas with iterative (or cyclic) references to be calculated after a specific number of iterations. Formulas with iterative references are repeated until the problem is solved. If this iterative calculations are not enabled, a formula with an iterative reference in a table causes an error message.

Iterative calculations are enabled and disabled with the `table:status` attribute. If iterative calculations are enabled, the `table:steps` attribute specifies the maximum number of iterations allowed. The `table:maximum-difference` attribute specifies the maximum difference allowed between two calculation results. The iteration is stopped if the result is less than the value of this attribute.

```
<define name="table-iteration">
    <element name="table:iteration">
        <optional>
            <attribute name="table:status" a:defaultValue="disable">
                <choice>
                    <value>enable</value>
                    <value>disable</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="table:steps" a:defaultValue="100">
                <ref name="positiveInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="table:maximum-difference"
                         a:defaultValue="0.001">
                <ref name="double"/>
            </attribute>
        </optional>
        <empty/>
    </element>
</define>
```

8.5.3 Table Cell Content Validations

Table cell content validations specify validation rules for the content of table cells. The `<table:content-validation>` element specifies such a validation rule. All validation rules that exist in a document are contained `<table:content-validations>` element. The validation rules themselves are named and referenced from the table cell by its name.

```
<define name="table-content-validations">
    <element name="table:content-validations">
        <oneOrMore>
            <ref name="table-content-validation"/>
        </oneOrMore>
    </element>
</define>

<define name="table-content-validation">
    <element name="table:content-validation">
        <ref name="table-validation-attlist"/>
        <optional>
            <ref name="table-help-message"/>
        </optional>
        <optional>
            <choice>
                <ref name="table-error-message"/>
                <group>
                    <ref name="table-error-macro"/>
                    <optional>
                        <ref name="office-event-listeners"/>
                    </optional>
                </group>
            </choice>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<table:content-validation>` element are:

- Name
- Condition
- Base cell address
- Allow empty cell
- Display list

Name

The `table:name` attribute specifies the name of the content validation. It is used to reference the validation rule from the cell the rule should applied to. The name is created automatically by the application.

```
<define name="table-validation-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

Condition

The `table:condition` attribute specifies the condition that must evaluate to "true" for all cells the validation rule is applied to. The value of this attribute should be a namespace prefix, followed by a Boolean expression.

A typical syntax of the expression may be 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() | cell-content-is-text()`
- `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) | cell-content-is-in-list( StringList )`
- `Operator ::= '<' | '>' | '<=' | '>=' | '=' | '!='`
- `Value ::= NumberValue | String | Formula`

- `StringList ::= StringList ';' String | String`
- A `Formula` is a formula without an equals (=) sign at the beginning. See section 8.1.3 for more information.
- A `String` comprises one or more characters surrounded by quotation marks.
- A `NumberValue` is a whole or decimal number. It must not contain comma separators for numbers of 1000 or greater.

```
<define name="table-validation-attlist" combine="interleave">
    <optional>
        <attribute name="table:condition">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Base Cell Address

The `table:base-cell-address` attribute specifies the address of the base cell for relative addresses in formulas that occur within a condition. This attribute is only necessary when the condition contains a formula. The value of this attribute must be an absolute cell address that contains a table name.

```
<define name="table-validation-attlist" combine="interleave">
    <optional>
        <attribute name="table:base-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

Allow Empty Cell

The `table:allow-empty-cell` attribute specifies whether or not a cell can be empty.

```
<define name="table-validation-attlist" combine="interleave">
    <optional>
        <attribute name="table:allow-empty-cell" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Display List

The `table:display-list` attribute specifies whether a list of values that occurs within a condition is displayed in the UI wile entering a cell value. The value of this attribute can be `none`, `unsorted` or `sort-ascending`.

- `none`: the list values are not displayed.
- `unsorted`: the list values are displayed in the order they occur in the condition.
- `sort-ascending`: the list values are displayed in ascending order.

```
<define name="table-validation-attlist" combine="interleave">
    <optional>
        <attribute name="table:display-list" a:defaultValue="unsorted">
```

```
            <choice>
                <value>none</value>
                <value>unsorted</value>
                <value>sort-ascending</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Help Message

The `<table:help-message>` element specifies a message to display if a user selects the cell. The element has an optional `table:title` attribute that specifies a title of the help message. It further has an optional `table:display` attribute that can be used to suppress the display of the message.

```
<define name="table-help-message">
    <element name="table:help-message">
        <optional>
            <attribute name="table:title">
                <ref name="string"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="table:display" a:defaultValue="false">
                <ref name="boolean"/>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
```

Error Message

The `<table:error-message>` element specifies a message to display if a user tries to enter invalid content into a cell i.e., content where the validation rule's condition evaluates to “false”. The element has an optional `table:title` attribute that specifies a title of the help message. It further has an optional `table:display` attribute that can be used to suppress the display of the message. The `table:message-type` attribute, that can take the values `stop`, `warning`, or `information`, specifies whether the message should be displayed as error (`stop`), warning (`warning`) or information only (`information`). In case the message is displayed as error, the operation that caused the validation check (for instance a cursor travel to leave the cell) is stopped.

```
<define name="table-error-message">
    <element name="table:error-message">
        <optional>
            <attribute name="table:title">
                <ref name="string"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="table:display" a:defaultValue="false">
                <ref name="boolean"/>
            </attribute>
        </optional>
        <optional>
```

```
            <attribute name="table:message-type" a:defaultValue="stop">
                <choice>
                    <value>stop</value>
                    <value>warning</value>
                    <value>information</value>
                </choice>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
```

Error Macro

As an alternative to displaying a message, a macro might be called if a cell contains invalid content. The macro in this case is specified by an `<office:event-listeners>` element as specified in section 12.4. The event name must be one that specifies an event that is called on invalid user input.

In addition to the `<office:event-listeners>` element, the `<table:error-macro>` element specifies whether the macro should be executed or not.

```
<define name="table-error-macro">
    <element name="table:error-macro">
        <optional>
            <attribute name="table:execute" a:defaultValue="true">
                <ref name="boolean"/>
            </attribute>
        </optional>
    </element>
</define>
```

8.5.4 Label Ranges

Label ranges can be used to assign names to rows and columns, or to parts of rows and columns, where the names themselves are specified as the content of table cells. More precisely, the label range element `<table:label-range>` specifies a label cell range which contain the labels, and data cell range which specifies the rows or columns whose content is referenced by the labels.

There are two types of label ranges.

- Label ranges for columns
- Label ranges for rows.

The data cell range should have the same height and vertical position like the label cell range if row labels are specified, or should have the same width and horizontal position like the label range if column labels are specified. For information on defining a cell range, see section 8.3.1.

Labels can be used within formula like any other name. All label ranges that exist in a document are contained within a single `<table:label-ranges>` element.

```
<define name="table-label-ranges">
    <element name="table:label-ranges">
        <zeroOrMore>
            <ref name="table-label-range"/>
        </zeroOrMore>
    </element>
```

```
</define>

<define name="table-label-range">
    <element name="table:label-range">
        <ref name="table-label-range-attlist"/>
        <empty/>
    </element>
</define>
```

Label Cell Range Address

The table:label-cell-range-address attribute specifies the cell range address of the labels.

```
<define name="table-label-range-attlist" combine="interleave">
    <attribute name="table:label-cell-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
</define>
```

Data Cell Range Address

The table:data-cell-range-address attribute specifies the cell range address of the data.

```
<define name="table-label-range-attlist" combine="interleave">
    <attribute name="table:data-cell-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
</define>
```

Orientation

The table:orientation attribute specifies the orientation of the label range. This attribute can have a value of column or row.

```
<define name="table-label-range-attlist" combine="interleave">
    <attribute name="table:orientation">
        <choice>
            <value>column</value>
            <value>row</value>
        </choice>
    </attribute>
</define>
```

8.5.5 Named Expressions

The named expressions element <table:named-expressions> contains a collection of assignments of names to expressions, so that the names can be use to refer to the expression.

The following expression can get names:

- cell ranges.
- Other expressions, for example, parts of a formula.

```
<define name="table-named-expressions">
    <element name="table:named-expressions">
        <zeroOrMore>
            <choice>
```

```
                <ref name="table-named-range"/>
                <ref name="table-named-expression"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

Named Range

The named range element `<table:named-range>` specifies a cell range that has a name assigned. For information on defining a cell range, see section 8.3.1.

The `table:name` attribute specifies the name of the range, and the `table:cell-range-address` attribute its address. The address can be either absolute or relative. If the cell range address is relative, the `table:base-cell-address` attribute must exist additionally. It specifies the base cell address for the cell range. This address must be absolute. Therefore a table name in the address is required, but the dollar signs that indicate an absolute address can be omitted.

An additional `table:range-usable-as` attribute specifies whether the name of the range can be used within the specification of a print range, a filter, a repeating row, or a repeat column. 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`.

```
<define name="table-named-range">
    <element name="table:named-range">
        <ref name="table-named-range-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-named-range-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
    <attribute name="table:cell-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
    <optional>
        <attribute name="table:base-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:range-usable-as" a:defaultValue="none">
            <choice>
                <value>none</value>
                <list>
                    <oneOrMore>
                        <choice>
                            <value>print-range</value>
                            <value>filter</value>
                            <value>repeat-row</value>
                            <value>repeat-column</value>
                        </choice>
                    </oneOrMore>
                </list>
            </choice>
        </attribute>
    </optional>
</define>
```

Named Expression

The named expression element `<table:named-expression>` contains an expression with a name, for example, a part of a formula.

The `table:name` attribute specifies the name of the expression, and the `table:expression` attribute the expression itself. The expressions do not support the equal (=) sign as the first character. If the expression contains a named range or another named expression, the named range or named expression must be specified first, before the containing expression. If the expression contains a relative cell range address, the `table:base-cell-address` attribute must exist additionally. It specifies the base cell address for the cell range. This address must be absolute. Therefore a table name in the address is required, but the dollar signs that indicate an absolute address can be omitted.

```
<define name="table-named-expression">
    <element name="table:named-expression">
        <ref name="table-named-expression-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-named-expression-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
    <attribute name="table:expression">
        <ref name="string"/>
    </attribute>
    <optional>
        <attribute name="table:base-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

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>
```

8.6 Database Ranges

A database range is a named area in a table where database operations, but also some other kind of operations like filtering and sorting, can be performed. The Database Ranges element `<table:database-ranges>` contains a collection of all database ranges defined in a document.

```
<define name="table-database-ranges">
```

```
    <element name="table:database-ranges">
        <zeroOrMore>
            <ref name="table-database-range"/>
        </zeroOrMore>
    </element>
</define>
```

8.6.1 Database Range

The `<table:database-range>` defines a single database range.

```
<define name="table-database-range">
    <element name="table:database-range">
        <ref name="table-database-range-attlist"/>
        <optional>
            <choice>
                <ref name="table-database-source-sql"/>
                <ref name="table-database-source-table"/>
                <ref name="table-database-source-query"/>
            </choice>
        </optional>
        <optional>
            <ref name="table-filter"/>
        </optional>
        <optional>
            <ref name="table-sort"/>
        </optional>
        <optional>
            <ref name="table-subtotal-rules"/>
        </optional>
    </element>
</define>
```

Database Range Name

The `table:name` attribute specifies the name of the database range on which to perform operations. Within a single document, only one database range is allowed to have no name. This database range is usually automatically created by the application and is used to filter or sort data in a cell ranges without the user explicitly creating a database range.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Is Selection

The `table:is-selection` attribute specifies whether the database range includes a complete database, or a selection of records from a database only.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:is-selection" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

On Update Keep Styles

The `table:on-update-keep-styles` attribute specifies the behavior if the database range is updated. If the attribute value is “true”, the cell styles that are assigned to the cells in the first non-label row of the database range are used for all rows with in the database range. If the attribute value is “false”, all cells in the database range get the default cell style of the document assigned.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:on-update-keep-styles" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

On Update Keep Size

The `table:on-update-keep-size` attribute specifies the behavior of the database range if the size of the data in the data source changes. If the attribute value is true, the range retains its size. If the attribute value is false, the range does not retain its size.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:on-update-keep-size" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Has Persistent Data

The `table:has-persistent-data` attribute specifies whether the current data in a database range is saved when the document itself is saved.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:has-persistent-data" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Orientation

The `table:orientation` attribute specifies the orientation of the database range. The values of this attribute are `row` and `column`. The orientation is for instance used when sorting database ranges (see 8.6.5). If the orientation is `row`, the sorting takes places for rows, otherwise for columns.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:orientation" a:defaultValue="row">
            <choice>
                <value>column</value>
                <value>row</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Contains Header

The `table:contains-header` attribute specifies whether or not the the content of the database range's first row or column should be used to specify labels. If the attribute's value is `true`, the content of the first cell within a row or column can be used to reference the whole row or column within many spreadsheet operations, for instance from within data pilots.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:contains-header" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Display Filter Buttons

The `table:display-filter-buttons` buttons attribute specifies whether or not to display filter buttons. Filter buttons are list box controls displayed in the label cells whose list entries are the values that exist in the labeled row or column. Selecting one of these entries equals applying a filter to the database range that selects all row or columns where the cells in the labeled row or column have the selected value.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:display-filter-buttons"
                   a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Target Range Address

The `table:target-range-address` attribute specifies the cell range address of the database range. A differentiation between absolute and relative addresses is not possible. Therefore, a table name must be specified in the address and dollar signs are ignored.

```
<define name="table-database-range-attlist" combine="interleave">
    <attribute name="table:target-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
</define>
```

Refresh Delay

The `table:refresh-delay` attribute specifies a time delay between automatic refresh actions.

```
<define name="table-database-range-attlist" combine="interleave">
    <optional>
        <attribute name="table:refresh-delay">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.6.2 Database Source SQL

The `<table:database-source-sql>` element describes an SQL database that contains the source data of the database range.

```
<define name="table-database-source-sql">
    <element name="table:database-source-sql">
        <ref name="table-database-source-sql-attlist"/>
        <empty/>
    </element>
</define>
```

Database Name

A `table:database-name` attribute specifies the name of the SQL database where the data is imported from.

```
<define name="table-database-source-sql-attlist" combine="interleave">
    <attribute name="table:database-name">
        <ref name="string"/>
    </attribute>
</define>
```

SQL Statement

An `table:sql-statement` attribute specifies the SQL statement to use when importing data from an SQL database.

```
<define name="table-database-source-sql-attlist" combine="interleave">
    <attribute name="table:sql-statement">
        <ref name="string"/>
    </attribute>
</define>
```

Parse SQL Statement

A `table:parse-sql-statement` attribute specifies whether or not the application will parse SQL statements.

```
<define name="table-database-source-sql-attlist" combine="interleave">
    <optional>
        <attribute name="table:parse-sql-statement" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.6.3 Database Source Table

The database source table element `<table:database-source-table>` specifies that the source data of the database range is stored in a database table.

```
<define name="table-database-source-query">
    <element name="table:database-source-table">
        <ref name="table-database-source-table-attlist"/>
        <empty/>
    </element>
</define>
```

Database Name

The `table:database-name` name attribute specifies the name of the database where the data is imported from.

```
<define name="table-database-source-table-attlist" combine="interleave">
    <attribute name="table:database-name">
        <ref name="string"/>
    </attribute>
</define>
```

Table Name

A `table:database-table-name` attribute specifies the database table that data is imported from.

```
<define name="table-database-source-table-attlist" combine="interleave">
    <attribute name="table:database-table-name">
        <ref name="string"/>
    </attribute>
</define>
```

8.6.4 Database Source Query

The database source query element `<table:database-source-query>` specifies that the source data of the database range is is the result of a database query.

```
<define name="table-database-source-table">
    <element name="table:database-source-query">
        <ref name="table-database-source-query-attlist"/>
        <empty/>
    </element>
</define>
```

Database Name

A `table:database-name` attribute specifies the name of the database that data is imported from.

```
<define name="table-database-source-query-attlist" combine="interleave">
    <attribute name="table:database-name">
        <ref name="string"/>
    </attribute>
</define>
```

Query Name

A `table:query-name` attribute specifies the query to perform on the database whose data is being imported.

```
<define name="table-database-source-query-attlist" combine="interleave">
    <attribute name="table:query-name">
        <ref name="string"/>
    </attribute>
</define>
```

8.6.5 Sort

The sort element `<table:sort>` describes the sort keys that should be applied to a database range.

```
<define name="table-sort">
    <element name="table:sort">
        <ref name="table-sort-attlist"/>
        <oneOrMore>
            <ref name="table-sort-by"/>
        </oneOrMore>
    </element>
</define>
```

Bind Styles to Content

The `table:bind-styles-to-content` attribute specifies whether or not cells retain their style attributes after a sort operation.

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:bind-styles-to-content" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Target Range Address

If the optional `table:target-range-address` attribute is present, the result of the sort is copied into the specified cell range rather than in the source cell range specified by the database range. A differentiation between absolute and relative addresses is not possible. Therefore, a table name has to exist in the address and dollar signs are ignored.

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:target-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

Case Sensitive

The `table:case-sensitive` attribute specifies whether or not the sort operation is case sensitive.

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:case-sensitive" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Language

The `table:language` attribute specifies the natural language in which the comparison will occur.

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
</define>
```

Country

The `table:country` attribute specifies the country specific rules to be used in string comparisons for a particular natural language.

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

Algorithm

The `table:algorithm` attribute specifies the algorithm used to compare sort keys. The attribute's value is a an application but also language and country specific sort algorithm name like “phonetic (alphanumeric first)”. To avoid name clashed between different applications, the name should start with a namespace prefix

```
<define name="table-sort-attlist" combine="interleave">
    <optional>
        <attribute name="table:algorithm">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

8.6.6 Sort By

The sort by element `<table:sort-by>` specifies a key or field to sort, the data type of this field, and how to sort it.

```
<define name="table-sort-by">
    <element name="table:sort-by">
        <ref name="table-sort-by-attlist"/>
        <empty/>
    </element>
</define>
```

Field Number

The `table:field-number` number attribute specifies the row or column number to sort by. It is the number of a row or column within the database range.

```
<define name="table-sort-by-attlist" combine="interleave">
    <attribute name="table:field-number">
        <ref name="nonNegativeInteger"/>
    </attribute>
```

```
</define>
```

Data Type

The `table:data-type` attribute specifies the data type of the field to be sorted. Its value can be `text`, `number`, `automatic` or the name of user defined sort order. If the attribute value is `automatic`, the application must determine what type of data is in the field. User defined sort orders are for instance lists of names of months. Specifying user defined sort orders is application specific.

```
<define name="table-sort-by-attlist" combine="interleave">
    <optional>
        <attribute name="table:data-type" a:defaultValue="automatic">
            <choice>
                <value>text</value>
                <value>number</value>
                <value>automatic</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Order

The `table:order` attribute specifies whether to sort the data in ascending or descending order.

```
<define name="table-sort-by-attlist" combine="interleave">
    <optional>
        <attribute name="table:order" a:defaultValue="ascending">
            <choice>
                <value>ascending</value>
                <value>descending</value>
            </choice>
        </attribute>
    </optional>
</define>
```

8.6.7 Subtotal Rules

The subtotal rules element <table:subtotal-rules> specifies that provisional results (called subtotals) should be calculated for a database range. It contains information about the row or column provisional results should be calculated for, and also how these results are calculated. To calculate provisional results, the cell values of a row or column a grouped by their value, that is, all cells with the same content in the same field form a group. A provisional result is calculated and displayed at the end of each group.

```
<define name="table-subtotal-rules">
    <element name="table:subtotal-rules">
        <ref name="table-subtotal-rules-attlist"/>
        <optional>
            <ref name="table-sort-groups"/>
        </optional>
        <zeroOrMore>
            <ref name="table-subtotal-rule"/>
        </zeroOrMore>
    </element>
</define>
```

Bind Styles To Content

The `table:bind-styles-to-content` attribute specifies whether or not cells retain their style after a subtotal calculation. This attribute is only evaluated if the `table:sort-groups` element is present.

```
<define name="table-subtotal-rules-attlist" combine="interleave">
    <optional>
        <attribute name="table:bind-styles-to-content" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Case Sensitive

The `table:case-sensitive` attribute specifies whether or not the case of characters is important when comparing entries, for example, when sorting groups.

```
<define name="table-subtotal-rules-attlist" combine="interleave">
    <optional>
        <attribute name="table:case-sensitive" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Page Breaks On Group Change

The `table:page-breaks-on-group-change` on group change attribute specifies whether or not to insert a page break after the subtotal for each group.

```
<define name="table-subtotal-rules-attlist" combine="interleave">
    <optional>
        <attribute name="table:page-breaks-on-group-change"
                   a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.6.8 Subtotal Sort Groups

The optional sort groups element `<table:sort-groups>` specifies that columns or rows are sorted before grouping them, and how to sort them. It belongs to the subtotal rules element, see section 8.6.7.

```
<define name="table-sort-groups">
    <element name="table:sort-groups">
        <ref name="table-sort-groups-attlist"/>
        <empty/>
    </element>
</define>
```

Data Type

The `table:data-type` attribute specifies the data type of the column or row group to sort. See section 8.6.6 for details.

```
<define name="table-sort-groups-attlist" combine="interleave">
    <optional>
        <attribute name="table:data-type" a:defaultValue="automatic">
            <choice>
                <value>text</value>
                <value>number</value>
                <value>automatic</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Order

The `table:order` attribute specifies whether to sort the group data in ascending or descending order. See section 8.6.6 for details.

```
<define name="table-sort-groups-attlist" combine="interleave">
    <optional>
        <attribute name="table:order" a:defaultValue="ascending">
            <choice>
                <value>ascending</value>
                <value>descending</value>
            </choice>
        </attribute>
    </optional>
</define>
```

8.6.9 Subtotal Rule

The subtotal rule element `<table:subtotal-rule>` describes how to calculate the subtotals for a certain row or 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 row a column where subtotals should be calculated as well as the function to use for the calculation.

```
<define name="table-subtotal-rule">
    <element name="table:subtotal-rule">
        <ref name="table-subtotal-rule-attlist"/>
        <zeroOrMore>
            <ref name="table-subtotal-field"/>
        </zeroOrMore>
    </element>
</define>
```

Group By Field Number

The `table:group-by-field-number` attribute specifies the field, for example, a column, that is to be grouped. It is the number of a row or column within the database range.

```
<define name="table-subtotal-rule-attlist" combine="interleave">
    <attribute name="table:group-by-field-number">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

8.6.10 Subtotal Field

The subtotal field element `<table:subtotal-field>` contains the field number and the function that is used to calculate a provisional result.

```
<define name="table-subtotal-field">
    <element name="table:subtotal-field">
        <ref name="table-subtotal-field-attlist"/>
        <empty/>
    </element>
</define>
```

Field Number

The `table:field-number` attribute specifies the row or column a subtotal should be calculated for. It is the number of a row or column within the database range.

```
<define name="table-subtotal-field-attlist" combine="interleave">
    <attribute name="table:field-number">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

Function

The `table: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`.

```
<define name="table-subtotal-field-attlist" combine="interleave">
    <attribute name="table:function">
        <choice>
            <value>auto</value>
            <value>average</value>
            <value>count</value>
            <value>countnums</value>
            <value>max</value>
            <value>min</value>
            <value>product</value>
            <value>stdev</value>
            <value>stdevp</value>
            <value>sum</value>
            <value>var</value>
            <value>varp</value>
            <ref name="string"/>
        </choice>
    </attribute>
</define>
```

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>
```

8.7 Filters

Filters specify that only rows that match certain conditions should be visible

8.7.1 Table Filter

The table filter element `<table:filter>` describes how the data contained in a database range or data pilot tables is filtered. The condition specified in the element are applied to all rows specified in the database range or the data pilot table. Rows where the condition does not evaluate to true are made invisible.

```
<define name="table-filter">
    <element name="table:filter">
        <ref name="table-filter-attlist"/>
        <choice>
            <ref name="table-filter-condition"/>
            <ref name="table-filter-and"/>
            <ref name="table-filter-or"/>
        </choice>
    </element>
</define>
```

Target Range Address

If the optional `table:target-range-address` attribute is present, the result of the filter is copied into the specified cell range but all table rows remain visible. If the attribute is not present, the rows that do not match the filter conditions are not displayed. A differentiation between absolute and relative addresses is not possible. Therefore, a table name has to exist in the address and dollar signs are ignored.

```
<define name="table-filter-attlist" combine="interleave">
    <optional>
        <attribute name="table:target-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

Condition Source

The `table:condition-source` attribute specifies whether the condition is contained in the filter or encoded in a table range. If the value is `self` the condition is specified by the <table:filter> element's child elements. If the value is `cell-range` the condition is encoded into the cell range specified by the `table:condition-source-range-address` attribute.

```
<define name="table-filter-attlist" combine="interleave">
    <optional>
        <attribute name="table:condition-source" a:defaultValue="self">
            <choice>
```

```
                <value>self</value>
                <value>cell-range</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Condition Source Range Address

The `table:condition-source-range-address` attribute specifies a cell range that contains encoded conditions. The first row of the cell range has to contain the labels of the columns whose content should be filtered. The following rows contain conditions that have to evaluate to true for the cells contained in the columns. The conditions in each row are connected by an "and" operation, while the rows are connected by an "or" operation. This means that a row is of the source table is displayed if there is at least one row in the condition range where all conditions evaluate to true if they are applied to the columns specified in the first row of the condition range.

Example: If the condition source range is E1:F3 (shown yellow) and the source range is A1:C3 (shown green), only rows 2 and 3 are displayed.

	A	B	C	D	E	F	G	G	I
1	1	3	4		A	B			
2	1	5	6		=1	=5			
3	2	8	9		>=2				

Row 2 is displayed because the cell in column A has the value 1 and the cell in column B the value 5, so all conditions of the 2nd row of the condition range evaluate to true. Row 3 is displayed because the cell in column A is larger or equal than 2, and therefor the only condition in the the 3rd row of the condition range evaluates to true.

```
<define name="table-filter-attlist" combine="interleave">
    <optional>
        <attribute name="table:condition-source-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

Display Duplicates

The `table:display-duplicates` attribute specifies whether or not to display duplicate matches in the result.

```
<define name="table-filter-attlist" combine="interleave">
    <optional>
        <attribute name="table:display-duplicates" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.7.2 Filter And

The `<table:filter-and>` element specifies that the logical operator AND is applied to the conditions specified by the element's child elements.

```
<define name="table-filter-and">
    <element name="table:filter-and">
        <oneOrMore>
            <choice>
                <ref name="table-filter-or"/>
                <ref name="table-filter-condition"/>
            </choice>
        </oneOrMore>
    </element>
</define>
```

8.7.3 Filter Or

The `<table:filter-or>` element specifies that the logical operator OR is applied to the conditions specified by the element's child elements.

```
<define name="table-filter-or">
    <element name="table:filter-or">
        <oneOrMore>
            <choice>
                <ref name="table-filter-and"/>
                <ref name="table-filter-condition"/>
            </choice>
        </oneOrMore>
    </element>
</define>
```

8.7.4 Filter Condition

The table `<table:filter-condition>` element describes a single condition to apply in a filter operation.

```
<define name="table-filter-condition">
    <element name="table:filter-condition">
        <ref name="table-filter-condition-attlist"/>
        <empty/>
    </element>
</define>
```

Field Number

The field number attribute `table:field-number` specifies which field to use for the condition. A field number is the number of a row or column in the source range of the filter.

```
<define name="table-filter-condition-attlist" combine="interleave">
    <attribute name="table:field-number">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

Value

The `table:value` attribute specifies a value for the filter condition.

```
<define name="table-filter-condition-attlist" combine="interleave">
    <attribute name="table:value">
        <ref name="string"/>
    </attribute>
</define>
```

Operator

The operator attribute `table:operator` specifies what operator to use in the filter condition. This means that each cell contained in the columns specified by the field number (i.e., the `table:field-number` attribute) is compared with the value (i.e., the `table:value` attribute) using the given operator. The result of this comparison is the result of the filter sub-conditions specified by the `<table:filter-condition>` element.

The operators may or may not make use of regular expressions. The operators that use regular expressions are the following:

- `match` (matches)
- `!match` (does not match)

In both case, the `table:value` attribute contains the regular expression that the table cells have to match or must not match.

The relational operators that do not use regular expressions are:

- = (Equal to)
- != (Not equal to)
- < (Less than)
- > (Greater than)
- <= (Less than or equal to)
- >= (Greater than or equal to)

In addition, operators “empty”, “!empty”, “bottom values”, “top values”, “bottom percent”, and “top percent” can be used. To filter for example the lowest and highest percentage values, the latter two operators can be used.

```
<define name="table-filter-condition-attlist" combine="interleave">
    <attribute name="table:operator">
        <ref name="string"/>
    </attribute>
</define>
```

Case Sensitive

The `table:case-sensitive` case sensitive attribute determines whether a filter condition is case sensitive.

```
<define name="table-filter-condition-attlist" combine="interleave">
    <optional>
        <attribute name="table:case-sensitive" a:defaultValue="false">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Data Type

The `table:data-type` attribute specifies whether comparison shall take place as text or as numeric values.

```
<define name="table-filter-condition-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="table:data-type" a:defaultValue="text">
            <choice>
                <value>text</value>
                <value>number</value>
            </choice>
        </attribute>
    </optional>
</define>
```

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>
```

8.8 Data Pilot Tables

Data pilot tables allow it to analyze and evaluate data contained in spreadsheet tables. The data pilot tables element `<table:data-pilot-tables>` contains the collection of all data pilot tables within a document.

```
<define name="table-data-pilot-tables">
    <element name="table:data-pilot-tables">
        <zeroOrMore>
            <ref name="table-data-pilot-table"/>
        </zeroOrMore>
    </element>
</define>
```

8.8.1 Data Pilot Table

The `<table:data-pilot-table>` specifies a single data pilot table. Within data pilot tables, all combinations of values that exist in selected columns are collected, and for each of these combinations a formula is applied to the cells of other columns.

Example: Given is the following source table

	A	B	C	D	
1	**Article**	**City**	**Country**	**Amount**	**Price**
2	Main Unit	Hamburg	Germany	1	12
3	Monitor	Hamburg	Germany	2	15
4	Printer	Paris	France	4	13

5	Monitor	Paris	France	2	14
6	Main Unit	Paris	France	1	12
7	Monitor	Hamburg	Germany	2	10
8	Printer	Paris	France	2	16

The following data pilot table groups the source table by the columns “County”, “City” and “Article” and calculates the sum of the “Amount” as well as of the “Price” columns for each combinations of values of these three columns. The values of the Country and City columns are shown in columns, while the ones of the Article columns are shown in rows.

			Article			
Country	City	Data	Main Unit	Monitor	Printer	**Total**
France	Paris	Sum - Amount	1	2	6	**9**
		Sum - Price	12	14	29	**55**
Germany	Hamburg	Sum - Amount	1	4		**5**
		Sum - Price	12	25		**37**
Total sum - Amount			**2**	**6**	**6**	**14**
Total sum - Price			**24**	**39**	**29**	**92**

The columns that are used for grouping (here “County”, “City” and “Article”) are called category columns. The columns for which a formula is calculated based on the value combinations of the category columns (here “Amount” and “Price”) are called data columns. The individual values that exists within a category column are called members.

In general, the behavior of a data pilot is specified by fields, where each field has a name and a so called orientation. The category columns are specified by fields with the orientation “row” or “column” and the data columns are specified by fields that have the orientation “data”. In the above example, “Article” is a field with the orientation column, while “Country” and “City” are fields with the orientation row. “Amount” and “Price” are fields with “data” orientation.

A third kind of fields are data layout fields. Data layout fields are not connected to a column in the source table, but have the only the purpose to change the layout of the data pilot table. In the example, “Data” is a data layout field.

The order in which fields are specified is of relevance. It specified the order in which the data of category columns is grouped and results are displayed. The data pilot table below displays how the data pilot table changes if for instance the data layout field is specified before the category column fields.

Example: A data pilot with a modified layout

			Article			
Data	Country	City	Main Unit	Monitor	Printer	**Total**
Sum - Amount	France	Paris	1	2	6	**9**
	Germany	Hamburg	1	4		**5**
Sum - Price	France	Paris	12	14	29	**55**
	Germany	Hamburg	12	25		**37**
Total sum - Amount			**2**	**6**	**6**	**14**
Total sum - Price			**24**	**39**	**29**	**92**

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
- Show Filter Button
- Drill Down On Double Click

```
<define name="table-data-pilot-table">
    <element name="table:data-pilot-table">
        <ref name="table-data-pilot-table-attlist"/>
        <optional>
            <choice>
                <ref name="table-database-source-sql"/>
                <ref name="table-database-source-table"/>
                <ref name="table-database-source-query"/>
                <ref name="table-source-service"/>
                <ref name="table-source-cell-range"/>
            </choice>
        </optional>
        <oneOrMore>
            <ref name="table-data-pilot-field"/>
        </oneOrMore>
    </element>
</define>
```

Data Pilot Table Source

The source of the data pilot table is either stored in a database, that is, a database table itself, a SQL query or a named query, or it is a cell range located within the same document. It can also be provided by an external component in an implementation dependent way.

The source of the data pilot table is specified by one of the following elements that are contained in the `<table:data-pilot-table>` element:

- `<table:database-source-sql>` (see section 8.6.2)

- `<table:database-source-table>` (see section 8.6.3)
- `<table:database-source-query>` (see section 8.6.4)
- `<table:source-cell-range>` (see section 8.8.2)
- `<table:source-service>` (see section 8.8.3)

Data Pilot Table Name

The `table:name` attribute specifies the name of the data pilot table.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

Application Data

The `table:application-data` attribute specifies extra information about the data pilot table, which can be used by the application, for instance within macros. This data does not influence the behavior of the data pilot.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:application-data">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Grand Total

The `table:grand-total` attribute specifies whether a grand total column, row, or both should be displayed in addition to values calculated for each combination of values in the category columns. In the above example, grand totals are enabled. They are displayed in the row and column labeled “Total”.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:grand-total" a:defaultValue="both">
            <choice>
                <value>none</value>
                <value>row</value>
                <value>column</value>
                <value>both</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Ignore Empty Rows

The `table:ignore-empty-rows` attribute specifies whether or not empty rows in the source range should be ignored.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="table:ignore-empty-rows" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Identify Categories

The `table:identify-categories` attribute specifies whether rows that do not contain a value in one of the category columns should use the value of the nearest ancestor row that has a value, or whether such rows should be moved into a group (or category) of its own. If the attribute's value is `false`, empty values form a category of its own.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:identify-categories" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Target Range Address

The `table:target-range-address` attribute specifies where the target range of the data pilot table output, that is, where the data pilot table is displayed. A differentiation between absolute and relative addresses is not possible, that is, the address is interpreted as an absolute address even if it contains dollar signs. The range address must contain a table name.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <attribute name="table:target-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
</define>
```

Buttons

Within a data pilot table, some cells might be displayed as buttons to allow interactive operations on the table like changing the order of columns. The `table:buttons` attribute specifies all cells which should be displayed this way. Its value is a list of cell-addresses. A differentiation between absolute and relative addresses is not possible, that is, the addresses are interpreted as absolute addresses even if they contain dollar signs. All addresses must contain a table name.

In the examples above, button cells are displayed with a gray background.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:buttons">
            <ref name="cellRangeAddressList"/>
        </attribute>
    </optional>
</define>
```

Show Filter Button

The `table:show-filter-button` attribute specifies whether a filter button is shown in the UI within the Data Pilot. A filter button displays a filter dialog if pushed.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="table:show-filter-button" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Drill Down On Double Click

The `table:drill-down-on-double-click` attribute specifies how the data pilot table reacts on a double click into the data pilot table. If the attribute's value is `false`, a double click on a member label or the empty area next to it starts the edit mode of the table cell, like for cells outside of the data pilot table. This can then be used to rename group fields or members. If the attribute's value is `true`, a double click on a member label or the empty area next to it shows or hides details for that member. A double click elsewhere in a data pilot table has no effect.

```
<define name="table-data-pilot-table-attlist" combine="interleave">
    <optional>
        <attribute name="table:drill-down-on-double-click"
                   a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.8.2 Source Cell Range

If the source of a data pilot table is a cell range, the `<table:source-cell-range>` element contains information about the cell range and how the data pilot table gets the data from the range. Before the source data is processed by the data pilot data, a filter may be applied to it. This filter has to be specified by a `<table:filter>` child element.

```
<define name="table-source-cell-range">
    <element name="table:source-cell-range">
        <ref name="table-source-cell-range-attlist"/>
        <optional>
            <ref name="table-filter"/>
        </optional>
    </element>
</define>
```

The only attribute that may be associated with the source cell range element is:

- Cell range address

Cell Range Address

The `table:cell-range-address` attribute specifies the cell range containing the source data. The source cell range's address must be absolute. Therefore, the cell range address must contain a table name and dollar signs are ignored.

```
<define name="table-source-cell-range-attlist" combine="interleave">
    <attribute name="table:cell-range-address">
        <ref name="cellRangeAddress"/>
    </attribute>
</define>
```

8.8.3 Source Service

The source of a data pilot table can be “service”, that is, it can be provided by an external component. The source service element `<table:source-service>` contains information about the service which is used to create the data pilot table.

```
<define name="table-source-service">
    <element name="table:source-service">
        <ref name="table-source-service-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- Service name
- Source name
- Object name
- Source user name
- Source password

Service Name

The `table:name` attribute specifies the name of the service. The value of this attribute is implementation specific.

```
<define name="table-source-service-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

Source Name

The `table:source-name` attribute specifies a source name that is passed to the service implementation. Its value is application and service specific.

```
<define name="table-source-service-attlist" combine="interleave">
    <attribute name="table:source-name">
        <ref name="string"/>
    </attribute>
</define>
```

Object Name

The `table:object-name` attribute specifies the name of the object in the source which contains the data and is passed to the service implementation. Its value is application and service specific.

```
<define name="table-source-service-attlist" combine="interleave">
    <attribute name="table:object-name">
        <ref name="string"/>
    </attribute>
</define>
```

Source User Name

The `table:user-name` attribute specifies the user name required to access the source. It is passed to the service implementation. Its value is application and service specific.

```
<define name="table-source-service-attlist" combine="interleave">
    <optional>
        <attribute name="table:user-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Source Password

The `table:password` attribute specifies the password required to access the source. It is passed to the service implementation. Its value is application and service specific.

```
<define name="table-source-service-attlist" combine="interleave">
    <optional>
        <attribute name="table:password">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

8.8.4 Data Pilot Field

A data pilot table's fields are specified by `<table:data-pilot-field>` elements.

```
<define name="table-data-pilot-field">
    <element name="table:data-pilot-field">
        <ref name="table-data-pilot-field-attlist"/>
        <optional>
            <ref name="table-data-pilot-level"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-field-reference"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-groups"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the data pilot field element are:

- Source field name
- Orientation
- Is data layout field
- Function
- Used hierarchy

Source Field Name

For fields that specify category or data columns, the `table:source-field-name` attribute specifies the name or label of the column the field is connected to. If the source of the data pilot table is for instance a cell range, then the attribute's value has to be the column's label.

There can be multiple `<table:data-pilot-field>` elements with the same value for this attribute.

```
<define name="table-data-pilot-field-attlist" combine="interleave">
    <attribute name="table:source-field-name">
        <ref name="string"/>
    </attribute>
</define>
```

Orientation

The `table:orientation` attribute specifies the orientation of the source field. If the value is `data`, then the field specifies a data column. If the value is `row` or `column`, then the field specifies a category column. The value `hidden` is used for fields that have a corresponding column in the data pilot's source, but are not visible within the data pilot table. The value `page` indicates that an automatic filter (i.e., one that allows to choose one of the values that are contained in the column) should be generated for the corresponding column. In this case, an additional field with row, column or data orientation has to exist for the column.

If the attribute value is `page`, the `table:selected-page` attribute can be used to specify which value is selected for the filter.

```
<define name="table-data-pilot-field-attlist" combine="interleave">
    <choice>
        <attribute name="table:orientation">
            <choice>
                <value>row</value>
                <value>column</value>
                <value>data</value>
                <value>hidden</value>
            </choice>
        </attribute>
        <group>
            <attribute name="table:orientation">
                <value>page</value>
            </attribute>
            <attribute name="table:selected-page">
                <ref name="string"/>
            </attribute>
        </group>
    </choice>
</define>
```

Is Data Layout Field

The `table:is-data-layout-field` attribute specifies whether a field is a data layout field (see section 8.8.1). Data layout fields usually don't have a name.

```
<define name="table-data-pilot-field-attlist" combine="interleave">
    <optional>
        <attribute name="table:is-data-layout-field" a:defaultValue="false">
            <ref name="string"/>
        </attribute>
    </optional>
```

```
</define>
```

Function

The `table:function` attribute specifies the function which is applied to the cell values of data columns. It is only evaluated if the value of the `table:orientation` attribute is `data`. Possible values for this attribute are: `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`. For category columns the attribute's value `auto` can be used that specifies that no function is applied at all.

```
<define name="table-data-pilot-field-attlist" combine="interleave">
    <optional>
        <attribute name="table:function">
            <choice>
                <value>auto</value>
                <value>average</value>
                <value>count</value>
                <value>countnums</value>
                <value>max</value>
                <value>min</value>
                <value>product</value>
                <value>stdev</value>
                <value>stdevp</value>
                <value>sum</value>
                <value>var</value>
                <value>varp</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Used Hierarchy

If the data pilot source is provided by an external component or service, the data contained within category columns may not only grouped by its value, but it may be further divided into sub-groups or hierarchies. A date value for instance might be grouped by

- “year”, “month” and “day of month”, or
- “year”, “week” and “day of week”

If an external components supports hierarchies, it has to assign unique numbers to it. These numbers can be used in the `table:used-hierarchy` attribute to select the hierarchy that should be applied to the source field. The value means that no hierarchy should be applied at all.

```
<define name="table-data-pilot-field-attlist" combine="interleave">
    <optional>
        <attribute name="table:used-hierarchy" a:defaultValue="-1">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

8.8.5 Data Pilot Level

The data pilot level element `<table:data-pilot-level>` contains additional information about a data pilot field.

```
<define name="table-data-pilot-level">
    <element name="table:data-pilot-level">
        <ref name="table-data-pilot-level-attlist"/>
        <optional>
            <ref name="table-data-pilot-subtotals"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-members"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-display-info"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-sort-info"/>
        </optional>
        <optional>
            <ref name="table-data-pilot-layout-info"/>
        </optional>
    </element>
</define>
```

The attribute that may be associated associate with the data pilot level element is:

- Show empty

Show Empty

The `table:show-empty` attribute specifies whether or not fields that don't have any members should be displayed. If this attribute is not present, the application might or might not display such fields.

```
<define name="table-data-pilot-level-attlist" combine="interleave">
    <optional>
        <attribute name="table:show-empty">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.8.6 Data Pilot Subtotals

The data pilot subtotals element `<table:data-pilot-subtotals>` contains information about the provisional results that are displayed for every member of a field and the function used to calculate the result. Several provisional results can be calculated simultaneously. If the element is not present, the application might or might not display provisional results.

```
<define name="table-data-pilot-subtotals">
    <element name="table:data-pilot-subtotals">
        <zeroOrMore>
            <ref name="table-data-pilot-subtotal"/>
        </zeroOrMore>
    </element>
</define>
```

8.8.7 Data Pilot Subtotal

The data pilot subtotal element `<table:data-pilot-subtotal>` contains information about a single provision result calculation.

```
<define name="table-data-pilot-subtotal">
```

```
    <element name="table:data-pilot-subtotal">
        <ref name="table-data-pilot-subtotal-attlist"/>
        <empty/>
    </element>
</define>
```

The attribute that may be associated associate with the data pilot subtotal element is:

- Function

Function

The `table:function` attribute specifies the function used for the subtotal. Possible functions are `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

```
<define name="table-data-pilot-subtotal-attlist" combine="interleave">
    <attribute name="table:function">
        <choice>
            <value>auto</value>
            <value>average</value>
            <value>count</value>
            <value>countnums</value>
            <value>max</value>
            <value>min</value>
            <value>product</value>
            <value>stdev</value>
            <value>stdevp</value>
            <value>sum</value>
            <value>var</value>
            <value>varp</value>
            <ref name="string"/>
        </choice>
    </attribute>
</define>
```

8.8.8 Data Pilot Members

For category columns, it can be controlled whether certain members themselves or the information displayed for a certain member actually is displayed or not. The `<table:data-pilot-members>` element contains such information.

```
<define name="table-data-pilot-members">
    <element name="table:data-pilot-members">
        <zeroOrMore>
            <ref name="table-data-pilot-member"/>
        </zeroOrMore>
    </element>
</define>
```

8.8.9 Data Pilot Member

The data pilot member element `<table:data-pilot-member>` specifies which information is displayed for a certain member.

```
<define name="table-data-pilot-member">
    <element name="table:data-pilot-member">
        <ref name="table-data-pilot-member-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the data pilot member element are:

- Member name
- Display
- Show details

Member Name

The `table:name` attribute specifies the value for which display information is specified.

```
<define name="table-data-pilot-member-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

Display

The `table:display` attribute specifies whether or not a data pilot member is visible at all. If this attribute is not present, the application might or might not display the member.

```
<define name="table-data-pilot-member-attlist" combine="interleave">
    <optional>
        <attribute name="table:display">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Show Details

The `table:show-details` attribute specifies whether additional fields are displayed for a member. This attribute changes the behavior of a data pilot only if there are several fields with the orientation row or column. If this is the case, and if the attribute's value is `false` for a field with row or column orientation that is not the last field with this orientation, then no members are displayed for all following fields with the same orientation. Instead of this, the data displayed for these fields will be summarized.

```
<define name="table-data-pilot-member-attlist" combine="interleave">
    <optional>
        <attribute name="table:show-details">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.8.10 Data Pilot Display Info

The `<table:data-pilot-display-info>` element restricts the number rows that are displayed for a category field to a specific number of values of a data field.

```
<define name="table-data-pilot-display-info">
    <element name="table:data-pilot-display-info">
        <ref name="table-data-pilot-display-info-attlist"/>
        <empty/>
    </element>
</define>
```

Enabled

The `table:enabled` attribute specifies whether the `<table:data-pilot-display-info>` element is evaluated or not.

```
<define name="table-data-pilot-display-info-attlist" combine="interleave">
    <attribute name="table:enabled">
        <ref name="boolean"/>
    </attribute>
</define>
```

Data Field

The `table:data-field` attribute specifies the data field whose values are taken into account.

```
<define name="table-data-pilot-display-info-attlist" combine="interleave">
    <attribute name="table:data-field">
        <ref name="string"/>
    </attribute>
</define>
```

Member Count

The `table:member-count` attribute specifies how many values from the top or from the bottom of data field's column are shown.

```
<define name="table-data-pilot-display-info-attlist" combine="interleave">
    <attribute name="table:member-count">
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

Display Member Mode

The `table:display-member-mode` attribute specifies whether the values specified by `table:member-count` should be taken from the top or from the bottom of a data field's column.

```
<define name="table-data-pilot-display-info-attlist" combine="interleave">
    <attribute name="table:display-member-mode">
        <choice>
            <value>from-top</value>
            <value>from-bottom</value>
        </choice>
    </attribute>
</define>
```

8.8.11 Data Pilot Sort Info

The `<table:data-pilot-sort-info>` element specifies how the members of a category field are sorted.

```
<define name="table-data-pilot-sort-info">
    <element name="table:data-pilot-sort-info">
        <ref name="table-data-pilot-sort-info-attlist"/>
        <empty/>
    </element>
</define>
```

Sort Mode

The `table:sort-mode` attribute describes how to sort the members of a single data pilot field. If the mode is `data`, then the members of the current category field a sorted according to their values in the data field specified by the `table:data-field` attribute. If the mode is `manual`, the user can sort the members in the field manually. If the mode is `name`, the members in the field are sorted by their name.

```
<define name="table-data-pilot-sort-info-attlist" combine="interleave">
    <choice>
        <group>
            <attribute name="table:sort-mode">
                <value>data</value>
            </attribute>
            <attribute name="table:data-field">
                <ref name="string"/>
            </attribute>
        </group>
        <attribute name="table:sort-mode">
            <choice>
                <value>none</value>
                <value>manual</value>
                <value>name</value>
            </choice>
        </attribute>
    </choice>
</define>
```

Sort Order

The `table:sort-order` attribute specifies whether to sort the members ascending or descending.

```
<define name="table-data-pilot-sort-info-attlist" combine="interleave">
    <attribute name="table:order">
        <choice>
            <value>ascending</value>
            <value>descending</value>
        </choice>
    </attribute>
</define>
```

8.8.12 Data Pilot Layout Info

The `<table:data-pilot-layout-info>` element describes how to layout the field.

```
<define name="table-data-pilot-layout-info">
    <element name="table:data-pilot-layout-info">
        <ref name="table-data-pilot-layout-info-attlist"/>
        <empty/>
    </element>
</define>
```

Layout Mode

The table:layout-mode attribute describes how to layout the field. It may have the following values:

- tabular-layout: Tabular layout mode is the layout, where each member's name is on the same row as the first member from the following field. Subtotals are always shown below a member's data in this mode.
- outline-subtotals-top: In outline layout mode, the members from the following field start in the row below a member's name, like in traditional database reports. Subtotals are shown at the top (in the same row as the member's name). When the subtotals take up more than one row (manually selected, or because there are several data fields), they are always shown below the member's data, regardless of the setting.
- outline-subtotals-bottom: Like outline-subtotals-top, except that subtotals are shown at the bottom (below the member's data, as in tabular layout mode).

```
<define name="table-data-pilot-layout-info-attlist" combine="interleave">
    <attribute name="table:layout-mode">
        <choice>
            <value>tabular-layout</value>
            <value>outline-subtotals-top</value>
            <value>outline-subtotals-bottom</value>
        </choice>
    </attribute>
</define>
```

Add empty lines

If the attribute table:add-empty-lines has the value true, an empty row is inserted in the data pilot table after the data (including the subtotals) for each member of the field.

```
<define name="table-data-pilot-layout-info-attlist" combine="interleave">
    <attribute name="table:add-empty-lines">
        <ref name="boolean"/>
    </attribute>
</define>
```

8.8.13 Data Pilot Field Reference

The <table:data-pilot-field-reference> element describes data which can be used to modify the displayed values of data fields.

```
<define name="table-data-pilot-field-reference">
    <element name="table:data-pilot-field-reference">
        <ref name="table-data-pilot-field-reference-attlist"/>
    </element>
</define>
```

Reference Field

The table:field-name attribute references a category field whose members influence the displayed values of the data field the <table:data-pilot-field-reference> is part of.

```
<define name="table-data-pilot-field-reference-attlist" combine="interleave">
    <attribute name="table:field-name">
        <ref name="string"/>
    </attribute>
```

```
</define>
```

Reference Member Type

The `table:member-type` attribute specifies the member of the referenced category field, whose value within the current data field has to be taken into account. If its value is `next` (`previous`) then the value of the data field for the next (previous) visible member of the referenced category field will be taken into account. If its value is `named`, then the `table:member-name` specifies the member whose value within the data field is taken into account.

For `previous` and `next`, empty members are skipped.

```
<define name="table-data-pilot-field-reference-attlist" combine="interleave">
    <choice>
        <group>
            <attribute name="table:member-type">
                <value>named</value>
            </attribute>
            <attribute name="table:member-name">
                <ref name="string"/>
            </attribute>
        </group>
        <attribute name="table:member-type">
            <choice>
                <value>previous</value>
                <value>next</value>
            </choice>
        </attribute>
    </choice>
</define>
```

Reference Type

The `table:type` attribute specifies the how the referenced category field influences the displayed values of the data field. It may have one of the following values:

- `none`: This value means that the results in the data fields are displayed unmodified.
- `member-difference`: From each result, the value calculated for the category field member specified by the `table:member-type` and `table:member-name` attributes is subtracted.
- `member-percentage`: Each result is divided by the value calculated for the category field member specified by the `table:member-type` and `table:member-name` attributes. Division by zero results in an error. Empty results are shown as “0”. If the `table:member-type` attribute has the value `previous`, “1” is displayed as first value. If the `table:member-type` attribute has the value `next`, “1” is displayed as last value.
- `member-percentage-difference`: From each result, the value calculated for the category field member specified by the `table:member-type` and `table:member-name` attributes is subtracted, and the result is divided by this value again. Division by zero results in an error. Otherwise, the rules for `member-difference` apply.
- `running-total`: Each result is added to the sum of the results for preceding members in the referenced category field, in the reference field's sort order, and the total sum is shown.
- `row-percentage`: Each result is divided by the total result for its row in the data pilot table. If there are several data fields, the total for the result's data field is used. If there are subtotals

with manually selected summary functions, the total is calculated with the data field's summary function. Division by zero results in an error.

- `column-percentage`: Same as `row-percentage`, but the total for the result's column is used.
- `total-percentage`: Same as `row-percentage`, but the grand total for the result's data field is used.
- `index`: The row and column totals and the grand total are calculated as described above, and then are used to calculate the following expression: (original result * grand total) / (row total * column total).Division by zero results in an error.

```
<define name="table-data-pilot-field-reference-attlist" combine="interleave">
    <attribute name="table:type">
        <choice>
            <value>none</value>
            <value>member-difference</value>
            <value>member-percentage</value>
            <value>member-percentage-difference</value>
            <value>running-total</value>
            <value>row-percentage</value>
            <value>column-percentage</value>
            <value>total-percentage</value>
            <value>index</value>
        </choice>
    </attribute>
</define>
```

8.8.14 Data Pilot Groups

The `<table:data-pilot-groups>` element specifies that a data pilot field is a group field. A group field allows grouping of other fields. For example, if a data pilot table contains a column field with the name “city” which has the members “Berlin”, “Munich”, “Frankfurt”, “Hamburg”, “London”, “Manchester”, “Hastings” and “Liverpool”, then one may want to group the cities by their countries. To do so, a group field with name “city2” could be added to the data pilot table, that contains two groups called “England” and “Germany”. Each group here contains a list of the names of its members. In this example, the group “England” would contain “London”, “Manchester”, “Hastings” and “Liverpool”. The group “Germany” would contain “Berlin”, “Munich”, “Frankfurt” and “Hamburg”.

Grouping may also take place for numeric or date values.

```
<define name="table-data-pilot-groups">
    <element name="table:data-pilot-groups">
        <ref name="table-data-pilot-groups-attlist"/>
        <oneOrMore>
            <ref name="table-data-pilot-group"/>
        </oneOrMore>
    </element>
</define>
```

Source Field Name

The `table:source-field-name` attribute references the field containing the data that is grouped, if this data differs from the data that is referenced by the field itself.

```
<define name="table-data-pilot-groups-attlist" combine="interleave">
    <attribute name="table:source-field-name">
        <ref name="string"/>
    </attribute>
</define>
```

Start

If numeric or date values are grouped, the `table:date-start` and `table:start` attributes specify the start value for the grouping. All values that are lower than the start value are contained in a single group, while values that are equal to or higher than the start value are grouped as specified by the `table:grouped-by` and `table:step` attributes.

If the attribute's value is `auto`, the lowest value of the field is taken as start value.

```
<define name="table-data-pilot-groups-attlist" combine="interleave">
    <choice>
        <attribute name="table:date-start">
            <choice>
                <ref name="dateOrDateTime"/>
                <value>auto</value>
            </choice>
        </attribute>
        <attribute name="table:start">
            <choice>
                <ref name="double"/>
                <value>auto</value>
            </choice>
        </attribute>
    </choice>
</define>
```

End

If numeric or date values are grouped, the `table:date-end` and `table:end` attributes specify the end value for the grouping. All values that are higher than the end value are contained in a single group, while values that are equal to or lower than the end value are grouped as specified by the `table:grouped-by` and `table:step` attributes.

If the attribute's value is `auto`, the highest value of the field is taken as end value.

```
<define name="table-data-pilot-groups-attlist" combine="interleave">
    <choice>
        <attribute name="table:date-end">
            <choice>
                <ref name="dateOrDateTime"/>
                <value>auto</value>
            </choice>
        </attribute>
        <attribute name="table:end">
            <choice>
                <ref name="double"/>
                <value>auto</value>
            </choice>
        </attribute>
    </choice>
</define>
```

Step

The `table:step` attribute specifies the grouping of numeric values, by specifying the distance between the groups. For example, if the table:start attribute for the grouping has the value 5, and the table:step attribute has the value 2, all values that are equal to or higher than 5, but also lower than 7 are in one group. All values that are equal to or higher than 7, but also lower than 9 are in next group, and so on, until the end value is reached.

```
<define name="table-data-pilot-groups-attlist" combine="interleave">
    <attribute name="table:step">
        <ref name="double"/>
    </attribute>
</define>
```

Grouped By

The `table:grouped-by` attribute specifies the grouping of the date values. Date values can be grouped by seconds, minutes, hours, days, months, quarters or years. It date values are for instance grouped by minutes, all dates or times that are within the same minute are within one group. That, is if the dates `2004-08-27T12:34:46`, `2004-08-27T12:34:56` and `2004-08-27T12:35:46` are given, the first two would be within one group, while the last date would be a group of its own.

```
<define name="table-data-pilot-groups-attlist" combine="interleave">
    <attribute name="table:grouped-by">
        <choice>
            <value>seconds</value>
            <value>minutes</value>
            <value>hours</value>
            <value>days</value>
            <value>months</value>
            <value>quarters</value>
            <value>years</value>
        </choice>
    </attribute>
</define>
```

8.8.15 Data Pilot Group

If grouping takes place by specifying the member names, then the `<table:data-pilot-group>` element specifies the member names of a single group.

```
<define name="table-data-pilot-group">
    <element name="table:data-pilot-group">
        <ref name="table-data-pilot-group-attlist"/>
        <oneOrMore>
            <ref name="table-data-pilot-group-member"/>
        </oneOrMore>
    </element>
</define>
```

Name

The `table:name` attribute specifies the name of the group.

```
<define name="table-data-pilot-group-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

8.8.16 Data Pilot Group Member

The `<table:data-pilot-group-member>` element specifies the name of a single group member.

```
<define name="table-data-pilot-group-member">
    <element name="table:data-pilot-group-member">
        <ref name="table-data-pilot-group-member-attlist"/>
    </element>
</define>
```

Name

The `table:name` attribute specifies the name of the member.

```
<define name="table-data-pilot-group-member-attlist" combine="interleave">
    <attribute name="table:name">
        <ref name="string"/>
    </attribute>
</define>
```

8.9 Consolidation

A consolidation combines data from several independent table ranges. A new table range is calculated by applying a mathematical function to all cells in the source table ranges that have the same relative address within these ranges. A consolidation is defined by the `<table:consolidation>` element.

```
<define name="table-consolidation">
    <element name="table:consolidation">
        <ref name="table-consolidation-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- Function
- Source cell range addresses
- Target cell address
- Use label
- Link to source data

Function

The `table:function` 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`.

```
<define name="table-consolidation-attlist" combine="interleave">
    <attribute name="table:function">
        <choice>
            <value>auto</value>
            <value>average</value>
            <value>count</value>
            <value>countnums</value>
```

```
                <value>max</value>
                <value>min</value>
                <value>product</value>
                <value>stdev</value>
                <value>stdevp</value>
                <value>sum</value>
                <value>var</value>
                <value>varp</value>
                <ref name="string"/>
            </choice>
        </attribute>
</define>
```

Source Cell Range Addresses

The `table:source-cell-range-addresses` attribute contains a list of cell range addresses that specify the source cell ranges.

```
<define name="table-consolidation-attlist" combine="interleave">
    <attribute name="table:source-cell-range-addresses">
        <ref name="cellRangeAddressList"/>
    </attribute>
</define>
```

Target Cell Address

The `table:target-cell-address` attribute contains the target cell address.

```
<define name="table-consolidation-attlist" combine="interleave">
    <attribute name="table:target-cell-address">
        <ref name="cellAddress"/>
    </attribute>
</define>
```

Use Label

The `table:use-label` attribute specifies whether or not labels should be used by the consolidation for rows, columns or both. Possible values are `none`, `column`, `row` and `both`. If labels are used for rows or columns, the mathematical functions is applied to cells with equally labeled rows or columns rather than to cells with the same relative cell address.

```
<define name="table-consolidation-attlist" combine="interleave">
    <optional>
        <attribute name="table:use-labels" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>row</value>
                <value>column</value>
                <value>both</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Link to Source Data

The `table:link-to-source-data` attribute specifies whether the data in the consolidation table range should be linked to the source data, so that it is automatically updated if any changes are made to the source data.

```
<define name="table-consolidation-attlist" combine="interleave">
    <optional>
        <attribute name="table:link-to-source-data" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.10 DDE Links

The `<table:dde-links>` container element stores all DDE links within a spreadsheet document. Every link contains the DDE Source and the data of the last connection. See section 12.6.3 for details.

See section 12.6 for the use of DDE connections.

```
<define name="table-dde-links">
    <element name="table:dde-links">
        <oneOrMore>
            <ref name="table-dde-link"/>
        </oneOrMore>
    </element>
</define>
```

8.11 Change Tracking in Spreadsheets

Within spreadsheet documents, changes to tables can be tracked. This section describes how this change tracking information is represented.

Change tracking of tables is not supported for text documents.

8.11.1 Tracked Changes

All changes that have been applied to a spreadsheet document are stored in a list. The list contains an element for each change made to the document. To track the changes to a spreadsheet document, the `<table:tracked-changes>` element must be present.

```
<define name="table-tracked-changes">
    <element name="table:tracked-changes">
        <ref name="table-tracked-changes-attlist"/>
        <zeroOrMore>
            <choice>
                <ref name="table-cell-content-change"/>
                <ref name="table-insertion"/>
                <ref name="table-deletion"/>
                <ref name="table-movement"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

Track Changes

The `table:track-changes` attribute specifies whether or not the change tracking is enabled.

```
<define name="table-tracked-changes-attlist" combine="interleave">
    <optional>
        <attribute name="table:track-changes" a:defaultValue="false">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

8.11.2 Insertion

The `<table:insertion>` element contains the information that is required to identify any insertion of content. This content can be one or more rows, one or more columns, or a table.

```
<define name="table-insertion">
    <element name="table:insertion">
        <ref name="table-insertion-attlist"/>
        <ref name="common-table-change-attlist"/>
        <ref name="office-change-info"/>
        <optional>
            <ref name="table-dependencies"/>
        </optional>
        <optional>
            <ref name="table-deletions"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- Acceptance State (see section 8.11.18)
- Rejecting Change ID (see section 8.11.18)
- Type
- Position
- Count
- Table

Type

The `table:type` attribute specifies the type of the insertion. It can be `row`, `column` or `table`.

```
<define name="table-insertion-attlist" combine="interleave">
    <attribute name="table:type">
        <choice>
            <value>row</value>
            <value>column</value>
            <value>table</value>
        </choice>
    </attribute>
</define>
```

Position

The `table:position` attribute specifies the position where the insertion was made in the table. Depending on the insertion type, It is either the number of a row, a column or a table.

```
<define name="table-insertion-attlist" combine="interleave">
    <attribute name="table:position">
```

```
        <ref name="integer"/>
    </attribute>
</define>
```

Count

The `table:count` attribute specifies the count of inserted rows, columns or tables.

```
<define name="table-insertion-attlist" combine="interleave">
    <optional>
        <attribute name="table:count" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Table

The `table:table` attribute specifies the number of the table where the insertion took place. This attribute only exists for column and row insertions.

```
<define name="table-insertion-attlist" combine="interleave">
    <optional>
        <attribute name="table:table">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Example: Insertion of text in a cell

```
<table:tracked-changes>
    <table:insertion table:id="c001" table:acceptance-state="pending"
                     table:type="column" table:position="5">
        <office:change-info>
            <dc:creator>Sascha Ballach</dc:creator>
            <dc:date>1999-55-18T12:56:04</dc:date>
        </office:change-info>
    </table:insertion>
</table:tracked-changes>
```

8.11.3 Dependencies

The `<table:dependencies>` element contains the information on which other tracked changes a tracked change depends. Every element of the tracked-changes can contain a `<table:dependencies>` element.

```
<define name="table-dependencies">
    <element name="table:dependencies">
        <oneOrMore>
            <ref name="table-dependency"/>
        </oneOrMore>
    </element>
</define>
```

8.11.4 Dependence

The `<table:dependency>` element contains the information about one change action on which the parent element depends. The change action on which the current depends is referenced by an id.

```
<define name="table-dependency">
    <element name="table:dependency">
        <attribute name="table:id">
            <ref name="string"/>
        </attribute>
        <empty/>
    </element>
</define>
```

8.11.5 Deletions

The `<table:deletions>` element contains all deletions which are performed while tracking a single change to a table.

```
<define name="table-deletions">
    <element name="table:deletions">
        <oneOrMore>
            <choice>
                <ref name="table-cell-content-deletion"/>
                <ref name="table-change-deletion"/>
            </choice>
        </oneOrMore>
    </element>
</define>
```

8.11.6 Cell Content Deletion

The `<table:cell-content-deletion>` element specifies that a cell content has been deleted. It contains the address of the effected cell and its former content. If a `text:id` attribute is present, it specifies the id of a previously tracked change for the cell that gets deleted by the current change.

```
<define name="table-cell-content-deletion">
    <element name="table:cell-content-deletion">
        <optional>
            <attribute name="table:id">
                <ref name="string"/>
            </attribute>
        </optional>
        <optional>
            <ref name="table-cell-address"/>
        </optional>
        <optional>
            <ref name="table-change-track-table-cell"/>
        </optional>
    </element>
</define>
```

8.11.7 Change Deletion

The `<table:change-deletion>` element specified the id of a previously tracked change that gets deleted by the current change.

```
<define name="table-change-deletion">
    <element name="table:change-deletion">
        <optional>
            <attribute name="table:id">
                <ref name="string"/>
            </attribute>
        </optional>
        <empty/>
    </element>
</define>
```

8.11.8 Deletion

A `<table:deletion>` element contains content that was deleted while change tracking was enabled. The content of a cell that was deleted is either contained in the `<table:dependencies>`, or in the `<table:deletions>` element.

```
<define name="table-deletion">
    <element name="table:deletion">
        <ref name="table-deletion-attlist"/>
        <ref name="common-table-change-attlist"/>
        <ref name="office-change-info"/>
        <optional>
            <ref name="table-dependencies"/>
        </optional>
        <optional>
            <ref name="table-deletions"/>
        </optional>
        <optional>
            <ref name="table-cut-offs"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- Acceptance State (see section 8.11.18)
- Rejecting Change ID (see section 8.11.18)
- Type
- Position
- Table
- Multi Deletion Spanned

Type

The `table:type` attribute specifies the type of the deletion. It can be `row`, `column` or `table`.

```
<define name="table-deletion-attlist" combine="interleave">
    <attribute name="table:type">
        <choice>
            <value>row</value>
            <value>column</value>
            <value>table</value>
        </choice>
    </attribute>
</define>
```

Position

The `table:position` attribute specifies the position where the deletion was made in the table. Depending on the deletion type, It is either the number of a row, a column or a table.

```
<define name="table-deletion-attlist" combine="interleave">
    <attribute name="table:position">
        <ref name="integer"/>
    </attribute>
</define>
```

Table

The `table:table` attribute specifies the number of the table where the deletion took place. This attribute only exists for column and row deletions.

```
<define name="table-deletion-attlist" combine="interleave">
    <optional>
        <attribute name="table:table">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Multi Deletion Spanned

If multiple columns or rows were deleted simultaneously, each deleted row or column gets its own `<table:deletion>` element. The element of the first deleted row or column in this case has to carry a `table:multi-deletion-spanned` attribute that specifies the total number of deleted rows or columns.

```
<define name="table-deletion-attlist" combine="interleave">
    <optional>
        <attribute name="table:multi-deletion-spanned">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

8.11.9 Cut Offs

A `<table:cut-offs>` element contains information about previously tracked insertions or movements where parts of the new content created by this operation now gets deleted. An example for this might be a cell range that has previously been moved and that now overlaps with a row that gets deleted.

```
<define name="table-cut-offs">
    <element name="table:cut-offs">
        <choice>
            <oneOrMore>
                <ref name="table-movement-cut-off"/>
            </oneOrMore>
            <group>
                <ref name="table-insertion-cut-off"/>
                <zeroOrMore>
                    <ref name="table-movement-cut-off"/>
                </zeroOrMore>
            </group>
        </choice>
    </element>
</define>
```

8.11.10 Insertion Cut Off

The `<table:insertion-cut-off>` element contains the information where a insertion was deleted and which.

```
<define name="table-insertion-cut-off">
    <element name="table:insertion-cut-off">
        <ref name="table-insertion-cut-off-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- position

Id

The `table:id` attribute contains the id of the insertion where parts of now get deleted.

```
<define name="table-insertion-cut-off-attlist" combine="interleave">
    <attribute name="table:id">
        <ref name="string"/>
    </attribute>
</define>
```

Position

The `table:position` attribute specifies the number of the row or column within the insertion that gets deleted.

```
<define name="table-insertion-cut-off-attlist" combine="interleave">
    <attribute name="table:position">
        <ref name="integer"/>
    </attribute>
</define>
```

8.11.11 Movement Cut Off

The `<table:movement-cut-off>` element contains the information where a movement was deleted and which.

```
<define name="table-movement-cut-off">
    <element name="table:movement-cut-off">
        <ref name="table-movement-cut-off-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- start position, end position, position

Start Position, End Position, Position

The `table:start-position`, `table:end-position` and `table:position` attributes specify the position within the movement that gets deleted. If a single row or column gets deleted, the `table:position` attribute contains its number. If multiple rows or columns get deleted, the `table:start-position` and `table:end-position` attributes contain the number of the first (inclusive) and last (exclusive) deleted rows or columns.

```
<define name="table-movement-cut-off-attlist" combine="interleave">
    <choice>
        <attribute name="table:position">
            <ref name="integer"/>
        </attribute>
        <group>
            <attribute name="table:start-position">
                <ref name="integer"/>
            </attribute>
            <attribute name="table:end-position">
                <ref name="integer"/>
            </attribute>
        </group>
    </choice>
</define>
```

Example: Deletion of a column which do not contain content

```
<table:tracked-changes>
    <table:deletion table:id="c002" table:acceptance-state="pending"
                     table:type="column" table:position="9">
        <office:change-info>
            <dc:creator>Sascha Ballach</dc:creator>
            <dc:date>1999-05-18T12:56:04</dc:creator>
        </office:change-info>
    </table:deletion>
</table:tracked-changes>
```

8.11.12 Movement

A `<table:movement>` 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.

```
<define name="table-movement">
    <element name="table:movement">
        <ref name="common-table-change-attlist"/>
        <ref name="table-source-range-address"/>
        <ref name="table-target-range-address"/>
        <ref name="office-change-info"/>
        <optional>
            <ref name="table-dependencies"/>
        </optional>
        <optional>
            <ref name="table-deletions"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- Acceptance State (see section 8.11.18)

- Rejecting Change ID (see section 8.11.18)

8.11.13 Target Range Address, Source Range Address

The `<table:source-range-address>` and `<table:target-range-address>` specify the source and target cell address or cell range address of a movement.

```
<define name="table-source-range-address">
    <element name="table:source-range-address">
        <ref name="common-table-range-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-target-range-address">
    <element name="table:target-range-address">
        <ref name="common-table-range-attlist"/>
        <empty/>
    </element>
</define>


<define name="common-table-range-attlist" combine="interleave">
    <choice>
        <group>
            <ref name="common-table-cell-address-attlist"/>
        </group>
        <group>
            <ref name="common-table-cell-range-address-attlist"/>
        </group>
    </choice>
</define>
```

The attributes that may be associated with these elements are either

- Column, Row, and Table, or
- Start column, End column, Start row, End row, Start table, and End table

Column, Row, and Table

If the range address is a cell address then the three attributes `table:column`, `table:row` and `table:table` specify the column, row and table number of the cell.

```
<define name="common-table-cell-address-attlist" combine="interleave">
    <attribute name="table:column">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:row">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:table">
        <ref name="integer"/>
    </attribute>
</define>
```

Start Column, End Column, Start Row, End Row, Start Table, and End Table

If the range address is a cell range address instead of a cell address, the attributes `table:start-column`, `table:end-column`, `table:start-row`, `table:end-row`, table:start-table and table:end-table specify the start and end columns, rows and tables of the range. Start and end numbers both are inclusive.

```
<define name="common-table-cell-range-address-attlist" combine="interleave">
    <attribute name="table:start-column">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:start-row">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:start-table">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:end-column">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:end-row">
        <ref name="integer"/>
    </attribute>
    <attribute name="table:end-table">
        <ref name="integer"/>
    </attribute>
</define>
```

Example: Moving a cell

```
<table:tracked-changes>
    <table:movement table:id="ct1">
        <table:source-range-address table:column="0" table:row="0"
                                     table:table="0"/>
        <table:target-range-address table:column="1" table:row="1"
                                    table:table="0"/>
        <office:change-info>
            <dc:creator>Michael Brauer</dc:creator>
            <dc:date>2003-12-29T11:46:13,21"</dc:date>
        </office:change-info>
    </table:movement>
</table:tracked-changes>
```

8.11.14 Change Track Cell

The <table:change-track-table-cell> element contains all information of a table cell which are needed inside the change tracking elements. The element is very similar to a <table:table-cell> element, but contains some additional information.

```
<define name="table-change-track-table-cell" combine="interleave">
    <element name="table:change-track-table-cell">
        <ref name="table-change-track-table-cell-attlist"/>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
```

Cell Address

If the cell is a formula cell, the table:cell-address attribute is required and specifies the original address of the cell used in calculations.

```
<define name="table-change-track-table-cell-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="table:cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

Matrix Covered

If the cell is a matrix cell and not the base of the matrix the, `table:matrix-covered` attribute is necessary and its value has to be `true` to indicate that the cell is contained in a matrix.

```
<define name="table-change-track-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:matrix-covered" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Formulas and Values

The change track table cells additionally supports the attributes `table:formula`, `table:number-matrix-rows-spanned`, `table:number-matrix-columns-spanned`, `office:value-type`, `office:value`, `office:date-value`, `office:time-value` and `office:string-value` as described in section 8.1.3.

```
<define name="table-change-track-table-cell-attlist" combine="interleave">
    <optional>
        <attribute name="table:formula">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:number-matrix-columns-spanned">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:number-matrix-rows-spanned">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
    <optional>
        <ref name="common-value-and-type-attlist"/>
    </optional>
</define>
```

8.11.15 Cell Content Change

A `<table:cell-content-change>` element contains the information that is required to identify changes of the cell content.

```
<define name="table-cell-content-change">
    <element name="table:cell-content-change">
        <ref name="common-table-change-attlist"/>
        <ref name="table-cell-address"/>
        <ref name="office-change-info"/>
        <optional>
            <ref name="table-dependencies"/>
        </optional>
        <optional>
            <ref name="table-deletions"/>
        </optional>
        <ref name="table-previous"/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- ID (see section 8.11.18)
- Acceptance State (see section 8.11.18)
- Rejecting Change ID (see section 8.11.18)

8.11.16 Cell Address

The `<table:cell-address>` element contains the address of cell that is changed. Unlike other cell addresses, the address consists of the row, column and table number of the cell. This allows specifying addresses that are outside the valid cell address range, for instance have a negative column number.

```
<define name="table-cell-address">
    <element name="table:cell-address">
        <ref name="common-table-cell-address-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with this element are:

- Column, Row, and Table number (see section 8.11.13)

8.11.17 Previous

The `table:previous` element contains the previous cell content which is overwritten by the current change. If a `text:id` attribute is present, it specifies the id of a previously tracked change for the cell that gets changed again by the current change.

```
<define name="table-previous">
    <element name="table:previous">
        <optional>
            <attribute name="table:id">
                <ref name="string"/>
            </attribute>
        </optional>
        <ref name="table-change-track-table-cell"/>
    </element>
</define>
```

8.11.18 Common Change Tracking Attributes

Id

The `table:id` attribute specifies the id of the tracked change.

```
<define name="common-table-change-attlist" combine="interleave">
    <attribute name="table:id">
```

```
        <ref name="string"/>
    </attribute>
</define>
```

Acceptance state

The `table:acceptance-state` attribute specifies whether the tracked change has been accepted or rejected already, or whether an acceptance or rejection is still pending.

```
<define name="common-table-change-attlist" combine="interleave">
    <optional>
        <attribute name="table:acceptance-state" a:defaultValue="pending">
            <choice>
                <value>accepted</value>
                <value>rejected</value>
                <value>pending</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Rejecting Change Id

If the `table:rejecting-change-id` attribute is present, then the current change has been made to the table to implement the rejection of another previously tracked change. The attribute's value is the id of this previously tracked change that has been rejected.

```
<define name="common-table-change-attlist" combine="interleave">
    <optional>
        <attribute name="table:rejecting-change-id">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9 Graphic Content

This chapter provides the specification for the core elements of graphic applications like drawing or presentation applications, and for graphical objects contained in non-graphical applications, like word processor or spreadsheet applications.

9.1 Enhanced Page Features for Graphical Applications

9.1.1 Handout Master

For applications that support printing handout pages, this element is a template for automatically generating the handout pages. The element `<style:handout-master>` can contain any types of shapes. The most useful shape is the `<draw:page-thumbnail>`, which is replaced by actual pages from the document. The `<style:handout-master>` element is contained in the `<office:master-styles>` element. The `<office:master-styles>` must not contain more than one `<style:handout-master>` element.

```
<define name="style-handout-master">
    <element name="style:handout-master">
        <ref name="common-presentation-header-footer-attlist"/>
        <ref name="style-handout-master-attlist"/>
        <zeroOrMore>
            <ref name="shape"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<style:handout-master>` element are:

- Presentation Page Layout (placeholder objects)
- Page Layout (page size, margins etc.)
- Page Style
- Header Declaration
- Footer Declaration
- Date and Time Declaration

Presentation Page Layout

The attribute `presentation:presentation-page-layout-name` links to a `<style:presentation-page-layout>` element. See section 14.15 for information on the presentation page layout element. This attribute is optional.

```
<define name="style-handout-master-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:presentation-page-layout-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Page Layout

The `style:page-layout-name` attribute specifies a page layout which contains the sizes, border and orientation of the handout master page. See section 14.3 for details on page layouts.

```
<define name="style-handout-master-attlist" combine="interleave">
    <attribute name="style:page-layout-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

Page Style

The attribute `draw:style-name` assigns an additional formatting attributes to a handout master page by assigning a drawing page style. This attribute is optional. The fixed family for page styles is `drawing-page`.

```
<define name="style-handout-master-attlist" combine="interleave">
    <optional>
        <attribute name="draw:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Header Declaration

The `presentation:use-header-name` attribute specifies the name of the header field declaration (see section 9.11.2) that is used for all header fields (see section 9.10.1) that are displayed on the handout master page. See also section 9.1.4.

Footer Declaration

The `presentation:use-footer-name` attribute specifies the name of the footer field declaration (see section 9.11.3) that is used for all footer fields (see section 9.10.2) that are displayed on the handout master page. See also section 9.1.4.

Date and Time Declaration

The `presentation:use-date-time-name` attribute specifies the name of the date-time field declaration (see section 9.11.4) that is used for all date-time fields (see section 9.10.3) that are displayed on the handout master page. See also section 9.1.4.

9.1.2 Layer Sets

The element `<draw:layer-set>` may be contained in the master styles of graphical applications. It defines a set of layers. Layers group drawing objects. Drawing objects may be assigned to these layers with the help of their `draw:layer-name` attribute.

```
<define name="draw-layer-set">
    <element name="draw:layer-set">
        <zeroOrMore>
            <ref name="draw-layer"/>
        </zeroOrMore>
    </element>
</define>
```

9.1.3 Layer

The <draw:layer> element defines a single layer.

```
<define name="draw-layer">
    <element name="draw:layer">
        <ref name="draw-layer-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
    </element>
</define>
```

The <draw:layer element> may contain the following elements:

- Title (short accessible name). Use the <svg:title> child element as described in section 9.2.20.
- Long description (in support of accessibility). Use the <svg:desc> child element as described in section 9.2.20.

Name

Each element <draw:layer> is defined and referenced by its name that is contained in the draw:name attribute . Each drawing object inside a drawing or presentation document can be assigned to a layer. Layers virtually group the object. Each object that is assigned to a layer inherits the settings of the layer.

```
<define name="draw-layer-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="string"/>
    </attribute>
</define>
```

Protection

The draw:protected attribute specifies whether the drawing objects contain in the layer are protected from being modified.

```
<define name="draw-layer-attlist" combine="interleave">
    <optional>
        <attribute name="draw:protected" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Display

The draw:display attribute specifies whether the drawing objects contain in the layer are visible on the screen and/or printed.

```
<define name="draw-layer-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display" a:defaultValue="always">
            <choice>
                <value>always</value>
                <value>screen</value>
                <value>printer</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
```

9.1.4 Drawing Pages

The element `<draw:page>` is a container for content in a drawing or presentation document. Drawing pages are used for the following:

- Forms (see section 11.1)
- Drawings (see section 9.2)
- Frames (see section 9.3)
- Presentation Animations (see section 9.7)
- Presentation Notes (see section 9.1.5)

A master page must be assigned to each drawing page.

```
<define name="draw-page">
    <element name="draw:page">
        <ref name="common-presentation-header-footer-attlist"/>
        <ref name="draw-page-attlist"/>
        <optional>
            <ref name="office-forms"/>
        </optional>
        <zeroOrMore>
            <ref name="shape"/>
        </zeroOrMore>
        <optional>
            <choice>
                <ref name="presentation-animations"/>
                <ref name="animation-element"/>
            </choice>
        </optional>
        <optional>
            <ref name="presentation-notes"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<draw:page>` element are:

- Page name
- Page style
- Master page
- Presentation page layout
- Header declaration
- Footer declaration
- Date and time declaration

- ID

The elements that my be included in the `<draw:page>` element are:

- Forms
- Shapes
- Animations
- Presentation notes

Page Name

The `draw:name` attribute specifies the **name** of a drawing page. This attribute is optional; if it is used, the name must be unique. If it is not used, the application may generate a unique name.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="draw:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Page Style

The attribute `draw:style-name` assigns an additional formatting attributes to a drawing page by assigning a drawing page style. This attribute is optional. The fixed family for page styles is `drawing-page`.

For pages inside a presentation document, attributes from Presentation Page Attributes can also be used.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="draw:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Master Page

Each drawing page must have one master page assigned to it. The *master page*:

- Defines properties such as the size and `borders` of the *drawing page*
- Serves as a container for shapes that are used as a common background

The `draw:master-page-name` attribute specifies the name of the master page assigned to the drawing page. This attribute is required.

```
<define name="draw-page-attlist" combine="interleave">
    <attribute name="draw:master-page-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

Presentation Page Layout

If the drawing page was created using a presentation page layout, the attribute `presentation:presentation-page-layout-name` links to the corresponding `<style:presentation-page-layout>` element. See section 14.15 for information on the presentation page layout element. This attribute is optional.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:presentation-page-layout-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Header Declaration

The `presentation:use-header-name` attribute specifies the name of the header field declaration (see section 9.11.2) that is used for all header fields (see section 9.10.1) that are displayed on the page.

```
<define name="common-presentation-header-footer-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:use-header-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Footer Declaration

The `presentation:use-footer-name` attribute specifies the name of the footer field declaration (see section 9.11.3) that is used for all footer fields (see section 9.10.2) that are displayed on the page.

```
<define name="common-presentation-header-footer-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:use-footer-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Date and Time Declaration

The `presentation:use-date-time-name` attribute specifies the name of the date-time field declaration (see section 9.11.4) that is used for all date-time fields (see section 9.10.3) that are displayed on the page.

```
<define name="common-presentation-header-footer-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:use-date-time-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

ID

The `draw:id` attribute assigns a unique ID to a drawing page.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="draw:id">
            <ref name="ID"/>
        </attribute>
    </optional>
</define>
```

Navigation Order

The `draw:nav-order` attribute defines a logical navigation sequence for the graphical elements included in the page. Its value is a sequence of unique IDREFs. If this optional attribute is present, it must include all graphic elements not contained within a `<draw:g>` element. This attribute should reflect the intentional ordering of graphics as set by the document author.

```
<define name="draw-page-attlist" combine="interleave">
    <optional>
        <attribute name="draw:nav-order">
            <ref name="IDREFS"/>
        </attribute>
    </optional>
</define>
```

9.1.5 Presentation Notes

Each drawing page element in a presentation can have an additional presentation notes page, which contains a preview of the corresponding drawing page and additional graphic shapes. A notes page is described by the `<presentation:notes>` element, that may be contained in the `<draw:page>` element. See section 14.4.2 for more information about this element.

Example: Drawing page

```
<office:automatic-styles>
    <style:style style:name="gg3434" style:family="drawing-page">
        <style:drawing-page-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="titledia">
        <draw:rect .../>
        presentation:notes>
            <draw:text ...>this is a note</draw:text>
        </presentation:notes>
    </draw:page>
</office:body>
```

9.2 Drawing Shapes

This section describes drawing shapes that might occur within all kind of applications.

```
<define name="shape">
    <choice>
        <ref name="draw-rect"/>
        <ref name="draw-line"/>
        <ref name="draw-polyline"/>
        <ref name="draw-polygon"/>
        <ref name="draw-regular-polygon"/>
        <ref name="draw-path"/>
        <ref name="draw-circle"/>
        <ref name="draw-ellipse"/>
        <ref name="draw-g"/>
        <ref name="draw-page-thumbnail"/>
        <ref name="draw-frame"/>
        <ref name="draw-measure"/>
        <ref name="draw-caption"/>
        <ref name="draw-connector"/>
        <ref name="draw-control"/>
        <ref name="dr3d-scene"/>
        <ref name="draw-custom-shape"/>
    </choice>
</define>
```

9.2.1 Rectangle

The `<draw:rect>` element represents a rectangular drawing shape.

```
<define name="draw-rect">
    <element name="draw:rect">
        <ref name="draw-rect-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:rect>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position – see section 9.2.16.
- Round corners

The elements that may be contained in the `<draw:rect>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Round Corners

The attribute `draw:corner-radius` specifies the radius of the circle used to round off the corners of the rectangle.

```
<define name="draw-rect-attlist" combine="interleave">
    <optional>
        <attribute name="draw:corner-radius">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

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">
```

9.2.2 Line

The `<draw:line>` element represents a line.

```
<define name="draw-line">
    <element name="draw:line">
        <ref name="draw-line-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:line>` element are:

- Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position– see section 9.2.16.
- Start point
- End point

The elements that may be contained in the `<draw:line>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Start Point

The start point attributes `svg:x1` and `svg:y1` specify the start coordinates of the line.

```
<define name="draw-line-attlist" combine="interleave">
    <attribute name="svg:x1">
        <ref name="coordinate"/>
    </attribute>
    <attribute name="svg:y1">
        <ref name="coordinate"/>
    </attribute>
</define>
```

End Point

The end point attributes `svg:x2` and `svg:y2` specify the end coordinates of the line.

```
<define name="draw-line-attlist" combine="interleave">
    <attribute name="svg:x2">
        <ref name="coordinate"/>
    </attribute>
    <attribute name="svg:y2">
        <ref name="coordinate"/>
    </attribute>
</define>
```

9.2.3 Polyline

The `<draw:polyline>` element represents a polyline drawing shape.

Some implementations may ignore the size attribute, and instead determine the size of a shape exclusively from the shape data (i.e., polygon vertices).

```
<define name="draw-polyline">
    <element name="draw:polyline">
        <ref name="common-draw-points-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:polyline>` element are:

- Position, Size, View box, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Points

The elements that may be contained in the `<draw:polyline>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Points

The `draw:points` attribute stores a sequence of points, which are connected by straight lines. Each point consists of two coordinates. The coordinates are separated by a comma and the points are separated by white spaces.

```
<define name="common-draw-points-attlist">
    <attribute name="draw:points">
        <ref name="points"/>
    </attribute>
</define>
```

9.2.4 Polygon

The `<draw:polygon>` element represents a polygon. A polygon is a closed set of straight lines.

Some implementations may ignore the size attribute, and instead determine the size of a shape exclusively from the shape data (i.e., polygon vertices).

```
<define name="draw-polygon">
    <element name="draw:polygon">
        <ref name="common-draw-points-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:polygon>` element are:

- Position, Size, View box, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Points – see section 9.2.3

The elements that may be contained in the `<draw:polygon>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

9.2.5 Regular Polygon

The `<draw:regular-polygon>` element represents a regular polygon. A regular polygon is a polygon that is specified by its number of edges (that is equal to the number of its corners), rather than by arbitrary points.

```
<define name="draw-regular-polygon">
    <element name="draw:regular-polygon">
        <ref name="draw-regular-polygon-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:polygon>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Concave
- Corners
- Sharpness

The elements that may be contained in the `<draw:regular-polygon>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Concave

The `draw:concave` attribute specifies whether the polygon is convex or concave. For a convex polygon, the polygon corners are located on a single ellipse which has its center in the center of the polygon. In a concave polygon, two such ellipses are required, and corners that are located next to each other are located on different ellipses. An example for a convex polygon is a hexagon. An example for a concave polygon is a star. For concave polygons, an additional `draw:sharpness` attribute is required.

```
<define name="draw-regular-polygon-attlist" combine="interleave">
    <choice>
        <attribute name="draw:concave">
            <value>false</value>
        </attribute>
        <group>
            <attribute name="draw:concave">
                <value>true</value>
            </attribute>
            <ref name="draw-regular-polygon-sharpness-attlist"/>
        </group>
    </choice>
</define>
```

Corners

The `draw:corners` attribute specifies the number of polygon corners.

```
<define name="draw-regular-polygon-attlist" combine="interleave">
    <attribute name="draw:corners">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

Sharpness

For concave attributes, the `draw:sharpness` attribute specifies the radius of the ellipse on which the inner polygon corners are located. The value is a percentage, where 0% means that all corners are located on a single ellipse, while 100% means that the inner corners are located at the center point of the polygon. In general, if *r* is the radius of the polygon, and *s* is the sharpness, the inner corners a located on a ellipse that's radius is *r(100-s)/100*.

```
<define name="draw-regular-polygon-sharpness-attlist">
    <attribute name="draw:sharpness">
        <ref name="percent"/>
    </attribute>
</define>
```

9.2.6 Path

The `<draw:path>` element represents a 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 Bézier
- *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 holes in objects.

Some implementations may ignore the size attribute, and instead determine the size of a shape exclusively from the shape data (i.e., polygon vertices).

```
<define name="draw-path">
    <element name="draw:path">
        <ref name="common-draw-path-data-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:path>` element are:

- Position, Size, View box, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Path data

The elements that may be contained in the `<draw:path>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Path Data

The syntax for the attribute `svg:d` is described in §8 of the *Scalable Vector Graphics (SVG) 1.1 Specification* [SVG].

Some implementations may only supports a subset of the SVG path specification, for instance no mixtures of open and closed curves for one shape, or no elliptical arc command.

```
<define name="common-draw-path-data-attlist">
    <attribute name="svg:d">
        <ref name="pathData"/>
    </attribute>
</define>
```

9.2.7 Circle

The `<draw:circle>` element represents a circular drawing shape.

```
<define name="draw-circle">
    <element name="draw:circle">
        <ref name="draw-circle-attlist"/>
        <ref name="common-draw-circle-ellipse-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:circle>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Center point

- Radius
- Kind
- Start angle
- End angle

The elements that may be contained in the `<draw:circle>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Center Point

The center point attributes `svg:cx` and `svg:cy` specify the coordinates of the center point of the circle. If these optional attributes are not set, the position and size attributes are used to create them.

```
<define name="common-draw-circle-ellipse-attlist" combine="interleave">
    <optional>
        <attribute name="svg:cx">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:cy">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

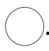
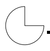
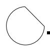
Radius

The `svg:r` attribute specifies the radius of the circle. If this optional attribute are not set, the position and size attributes are used to create circle.

```
<define name="draw-circle-attlist" combine="interleave">
    <optional>
        <attribute name="svg:r">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Kind

The `draw:kind` attribute specifies the appearance of the circle.

- `full` specifies a full circle or ellipse, like .
- `section` specifies a section of a circle or ellipse, like .
- `cut` specifies a circle or ellipse with a cut, like .

- `arc` specifies a circle or ellipse arc, like .

```
<define name="common-draw-circle-ellipse-attlist" combine="interleave">
    <optional>
        <attribute name="draw:kind" a:defaultValue="full">
            <choice>
                <value>full</value>
                <value>section</value>
                <value>cut</value>
                <value>arc</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Start Angle

For circles where the `draw:kind` attribute value is `section`, `cut` or `arc`, the `svg:start-angle` attribute specifies the start angle of the section, cut, or arc.

```
<define name="common-draw-circle-ellipse-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-angle">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

End Angle

For circles where the `draw:kind` attribute value is `section`, `cut` or `arc`, the `svg:end-angle` attribute specifies the end angle of the section, cut, or arc.

```
<define name="common-draw-circle-ellipse-attlist" combine="interleave">
    <optional>
        <attribute name="draw:end-angle">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

9.2.8 Ellipse

The `<draw:ellipse>` element represents an ellipse.

```
<define name="draw-ellipse">
    <element name="draw:ellipse">
        <ref name="common-draw-circle-ellipse-attlist"/>
        <ref name="draw-ellipse-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
```

```
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:ellipse>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Center point, Kind, Start angle, End angle – see section 9.2.7
- Radius

The elements that may be contained in the `<draw:ellipse>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Radius

The `svg:rx` and `svg:rx` attribute specify the horizontal and vertical radius of the ellipse. If these optional attributes are not set, the position and size attributes are used to create the ellipse.

```
<define name="draw-ellipse-attlist" combine="interleave">
    <optional>
        <attribute name="svg:rx">
            <ref name="length"/>
        </attribute>
        <attribute name="svg:ry">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

9.2.9 Connector

The `<draw:connector>` element represents a series of lines that are connected to the glue points of two other shapes.

```
<define name="draw-connector">
    <element name="draw:connector">
        <ref name="draw-connector-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:connector>` element are:

- Style, Layer, Z-Index, ID and Caption ID – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Type
- Start position
- Start shape
- Start glue point
- End position
- End shape
- End glue point
- Line skew

The elements that may be contained in the `<draw:connector>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Type

The `draw:type` attribute specifies how the connection between two points is rendered. The value of this attribute can be `standard`, `lines`, `line`, or `curve`.

- `standard`: a standard connector escapes the two connecting objects with straight lines and connects them with a straight perpendicular line.
- `lines:` a lines connector escapes the two connecting objects with straight lines and connects them with a straight (not necessarily perpendicular) line.
- `line:` a line connector draws one straight line between the two escape points of the connected objects.
- `curve:` a curve connector draws a single curved line between the two escape points of the connected objects.

```
<define name="draw-connector-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="draw:type" a:defaultValue="standard">
            <choice>
                <value>standard</value>
                <value>lines</value>
                <value>line</value>
                <value>curve</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Start Position

The start position attributes `svg:x1` and `svg:y1` specify the start position of a connector.

If the start position is connected to a shape, these attributes are optional because the start position defaults to the corresponding glue point on the target shape.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="svg:x1">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:y1">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

Start Shape

The `draw:start-shape` attribute identifies the drawing shape to which the start of this connector is connected by its name.

If a shape is connected to the start of a connector, the start position defaults to the corresponding glue point on the target shape.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-shape">
            <ref name="IDREF"/>
        </attribute>
    </optional>
</define>
```

Start Glue Point

The `draw:start-glue-point` attribute identifies the glue point in the start shape of the connector by its number. See section 9.2.19 for details on glue points.

If this attribute is not set and the start of the connector is connected to a shape, the application may choose the glue point. If the start of the connector is not connected to a shape, this attribute is ignored.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-glue-point">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

End Position

The end position attributes `svg:x2` and `svg:y2` specify the end position of a connector.

If the end position is connected to a shape, these attributes are optional because the end position defaults to the corresponding glue point on the target shape.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="svg:x2">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:y2">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

End Shape

The `draw:end-shape` attribute identifies the drawing shape to which the end of the connector is connected by its name.

If a shape is connected to the end of a connector, the end position defaults to the corresponding glue point on the target shape.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="draw:end-shape">
            <ref name="IDREF"/>
        </attribute>
    </optional>
</define>
```

End Glue Point

The `draw:end-glue-point` attribute identifies the glue point in the end shape of the connector by its number. See section 9.2.19 for details on glue points.

If this attribute is not set and the end of the connector is connected to a shape, the application may choose the glue point. If the end of the connector is not connected to a shape, this attribute is ignored.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="draw:end-glue-point">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Line Skew

The `draw:line-skew` attribute controls the generation of the lines that connect the start and end points. Depending on the type of connector, this can vary from one to three distances that move the connector lines relative to their normal position.

```
<define name="draw-connector-attlist" combine="interleave">
    <optional>
        <attribute name="draw:line-skew">
            <list>
                <ref name="length"/>
                <optional>
                    <ref name="length"/>
                    <optional>
                        <ref name="length"/>
                    </optional>
                </optional>
            </list>
        </attribute>
    </optional>
</define>
```

9.2.10 Caption

The `<draw:caption>` element represents a rectangular drawing shape with an additional set of lines. It can be used as a description for a fixed point inside a drawing.

```
<define name="draw-caption">
    <element name="draw:caption">
        <ref name="draw-caption-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:caption>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Caption point
- Round corners

The elements that may be contained in the `<draw:caption>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.

- Text – see section 9.2.17.

Caption Point

The caption point attributes `draw:caption-point-x and draw:caption-point-y` specify the position of the point that is captioned. A set of lines are rendered from the caption area.

```
<define name="draw-caption-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-point-x">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="draw:caption-point-y">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

Round Corners

The `draw:corner-radius` attribute specifies the radius of the circle used to round off the corners of the caption.

```
<define name="draw-caption-attlist" combine="interleave">
    <optional>
        <attribute name="draw:corner-radius">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

9.2.11 Measure

The `<draw:measure>` element represents a shape that is used to measure distances in drawings.

```
<define name="draw-measure">
    <element name="draw:measure">
        <ref name="draw-measure-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:measure>` element are:

- Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15

- Text anchor, table background, draw end position – see section 9.2.16
- Start position
- End position

The elements that may be contained in the `<draw:measure>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.

Start Position

The attributes `svg:x1 and svg:y1` specify the start point of the measured distance.

```
<define name="draw-measure-attlist" combine="interleave">
    <attribute name="svg:x1">
        <ref name="coordinate"/>
    </attribute>
    <attribute name="svg:y1">
        <ref name="coordinate"/>
    </attribute>
</define>
```

Draw End Position

The attributes `svg:x2 and svg:y2` specify the end point of the measured distance.

```
<define name="draw-measure-attlist" combine="interleave">
    <attribute name="svg:x2">
        <ref name="coordinate"/>
    </attribute>
    <attribute name="svg:y2">
        <ref name="coordinate"/>
    </attribute>
</define>
```

9.2.12 Control

The `<draw:control>` element represents a shape that is linked to a control inside an `<office:forms>` element (see section 11.1).

```
<define name="draw-control">
    <element name="draw:control">
        <ref name="draw-control-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:control>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position – see section 9.2.16
- Control

The elements that may be contained in the `<draw:control>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Glue points – see section 9.2.19.

Control

The attributes `draw:control` attribute specifies the control within a form (see section 11.5.2) that is linked to the control shape.

```
<define name="draw-control-attlist" combine="interleave">
    <attribute name="draw:control">
        <ref name="IDREF"/>
    </attribute>
</define>
```

9.2.13 Page Thumbnail

The `<draw:page-thumbnail>` element represents a rectangular area that displays the thumbnail of a drawing page.

```
<define name="draw-page-thumbnail">
    <element name="draw:page-thumbnail">
        <ref name="draw-page-thumbnail-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="presentation-shape-attlist"/>
        <ref name="common-draw-shape-with-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<draw:page-thumbnail>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position – see section 9.2.16

- Presentation class – see section 9.6.1
- Page number

The elements that may be contained in the `<draw:page-thumbnail>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.

Page Number

The `draw:page-number` attribute specifies the number of the page that is displayed as a thumbnail. For thumbnails on notes pages, the value of this attribute is fixed to the drawing page of the notes page. For thumbnails on handout master pages, the value of this attribute is the order in which the pages are previewed on the handout. For example, on a handout page with 4 thumbnails, the thumbnail with the lowest page number displays the first page when printing the first handout page and the fifth page when printing the second handout page and so on.

```
<define name="draw-page-thumbnail-attlist">
    <optional>
        <attribute name="draw:page-number">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

9.2.14 Grouping

The `<draw:g>` element represents a group of drawing shapes.

```
<define name="draw-g">
    <element name="draw:g">
        <ref name="draw-g-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-name-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-text-spreadsheet-shape-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="shape"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:g>` element are:

- Style, Z-Index, ID and Caption ID – see section 9.2.15.

- Text anchor, table background, draw end position – see section 9.2.16
- Position

The elements that may be contained in the `<draw:g>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Drawing shapes.

Position

For group shapes that are contained in text documents and anchored as character, the `svg:y` attribute specifies the vertical position of the shape.

```
<define name="draw-g-attlist" combine="interleave">
    <optional>
        <attribute name="svg:y">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

9.2.15 Common Drawing Shape Attributes

The attributes described in this section are common to all drawing shapes.

Name

The attribute `draw:name` assigns a name to the drawing shape.

```
<define name="common-draw-name-attlist" combine="interleave">
    <optional>
        <attribute name="draw:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Caption-ID

The `draw:caption-id` attribute establishes a relationship between a drawing objects and its caption. It takes a value of type IDREF. The value for `draw:caption-id` attribute is the target ID assigned to the `<draw:text-box>` (see section 9.3.1) used to represent the corresponding caption.

When a caption is assigned by a user agent, an id must be assigned to the element containing the text used to caption a drawing element. The drawing element being captioned must then be assigned the `draw:caption-id` attribute with an IDREF equivalent to the id `<draw:text-box>` containing the captioning text, thus establishing a relationship between the captioned text and the object captioned as needed for accessibility. Removing the caption should result in removing the `draw:caption-id` attribute of the object that was being captioned.

If the user agent supports a platform which provides a `draw:caption-id` relationship in its accessibility API, this relationship for captions should be used to fulfill the relationship.

See appendix E for guidelines how to use this attribute.

```
<define name="common-draw-caption-id-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-id">
            <ref name="IDREF"/>
        </attribute>
    </optional>
</define>
```

Position

The position attributes `svg:x` and `svg:y` specify the x and y coordinates of the start position of the drawing shape.

```
<define name="common-draw-position-attlist">
    <optional>
        <attribute name="svg:x">
            <ref name="coordinate"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:y">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

Size

The attributes `svg:width` and `svg:height` specify the width and height of the drawing shape.

```
<define name="common-draw-size-attlist">
    <optional>
        <attribute name="svg:width">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Transformation

The `draw:transform` attribute specifies a list of transformations that can be applied to a drawing shape.

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.

```
<define name="common-draw-transform-attlist">
    <optional>
        <attribute name="draw:transform">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

View Box

The `svg:viewBox` 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 `<draw:path>` element.

The syntax for using this attribute is the same as the [SVG] syntax. The value of the attribute are four numbers separated by white spaces, which define the left, top, right, and bottom dimensions of the user coordinate system.

Some implementations may ignore the view box attribute. The implied coordinate system then has its origin at the left, top corner of the shape, without any scaling relative to the shape.

```
<define name="common-draw-viewbox-attlist">
    <attribute name="svg:viewBox">
        <list>
            <ref name="integer"/>
            <ref name="integer"/>
            <ref name="integer"/>
            <ref name="integer"/>
        </list>
    </attribute>
</define>
```

Style

The `draw:style-name` and `presentation:style-name` attributes specify a style for the drawing shape. If `draw:style-name` is used, the shape is a regular graphic shape. If `presentation:style-name` is used, the shape is a presentation shape as described in section 9.6.

The value of both attributes is the name of a `<style:style>` element. If the `draw:style-name` attribute is used, the style must have a family value of `graphic`. If the `presentation:style-name` is used, the style must have a family value of `presentation`. The formatting properties of the specified style and its optional parent styles are used to format the shape. See also section 14.13.1.

The `draw:class-names` and `presentation:class-names` attributes take a whitespace separated list of either graphic or presentation style names. The referenced styles are applied in the order they are contained in the list. If both, `draw:style-name` and `draw:class-names`, or both `presentation:style-name` and `presentation:class-names` are present, the style referenced by the `style-name` attribute is treated as the first style in the list in the `class-names` attribute. Conforming application should support the `class-names` attribute and also should preserve it while editing.

```
<define name="common-draw-style-name-attlist">
    <choice>
        <group>
            <optional>
                <attribute name="draw:style-name">
                    <ref name="styleNameRef"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="draw:class-names">
                    <ref name="styleNameRefs"/>
                </attribute>
            </optional>
        </group>
        <group>
            <optional>
                <attribute name="presentation:style-name">
                    <ref name="styleNameRef"/>
                </attribute>
            </optional>
            <optional>
                <attribute name="presentation:class-names">
                    <ref name="styleNameRefs"/>
                </attribute>
            </optional>
        </group>
    </choice>
</define>
```

Text Style

The `draw:text-style-name` attribute specifies a style for the drawing shape that is used to format the text that can be added to this shape.

The value of this attribute is the name of a `<style:style>` element with a family value of `paragraph`.

```
<define name="common-draw-text-style-name-attlist">
    <optional>
        <attribute name="draw:text-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Layer

The attribute `draw:layer` can assign each shape to a layer. The value of this attribute must be the name of a layer inside the layer-set of the document.

```
<define name="common-draw-layer-name-attlist">
    <optional>
        <attribute name="draw:layer">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

ID

The `draw:id` attribute assigns an unique ID to a drawing shape that can be used to reference the shape.

```
<define name="common-draw-id-attlist">
    <optional>
        <attribute name="draw:id">
            <ref name="ID"/>
        </attribute>
    </optional>
</define>
```

Z-Index

Drawing shapes are rendered in a specific order. In general, the shapes are rendered in the order in which they appear in the XML document. To change the order, use the `svg:z-index` attribute.

This attribute is optional.

```
<define name="common-draw-z-index-attlist">
    <optional>
        <attribute name="draw:z-index">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

9.2.16 Common Shape Attributes for Text and Spreadsheet Documents

The attributes described in this section are common to all drawing shapes contained in text and spreadsheet documents.

End Position

If a drawing shape is included in a spreadsheet document and if the anchor of the shape is in a cell, then the attributes `table:end-cell-address`, `table:end-x` and `table:end-y` specify the end position of the shape and the size attributes are ignored. The end position is specified using the cell address of the cell in which the end position is located, and the `x` and `y` coordinates of the end position relative to the top left edge of the cell.

```
<define name="common-text-spreadsheet-shape-attlist" combine="interleave">
    <optional>
        <attribute name="table:end-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:end-x">
            <ref name="coordinate"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="table:end-y">
```

```
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

Table Background

If a drawing shape is included in a spreadsheet document, then the `table:table-background` attribute specifies whether or not the shape is in the table background. If the attribute is not existing, the shape is included in the foreground of the table.

```
<define name="common-text-spreadsheet-shape-attlist" combine="interleave">
    <optional>
        <attribute name="table:table-background">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Text Anchor

Within text documents, the anchor type attribute `text:anchor-type` specifies how a frame is bound to the text document. 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 drawing shape 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 drawing shape element. If no `text:anchor-page-number` attribute is given, the anchor position is the page at which the character behind the drawing object element appears.	Either • At the start of the document body, outside any paragraph or frame, provided a `text:anchor-page-number` attribute is given. Or • Inside any paragraph element that is not contained in a header, footer, footnote, or text box, if a `text:anchor-page-number` attribute is not given.	The physical page number is the number assigned to the page if all pages in the document are counted starting with page 1.

*If the value of the **`text:anchor-type`** attribute is ...*	*The anchor position is...*	*The drawing shape element appears ...*	*Notes*
`frame`	The parent text box that the current drawing shape element is contained in.	In the element representing the text box to which the drawing object is bound. For example, if an image is bound to a text box, the image element is located in the text box element.	
`paragraph`	The paragraph that the current drawing shape element is contained in.	At the start of the paragraph element.	
`char`	The character after the drawing shape element.	Just before the character.	
`as-char`	There is no anchor position. The drawing shape behaves like a character.	At the position where the character appears in the document.	

```
<define name="common-text-spreadsheet-shape-attlist" combine="interleave">
    <ref name="common-text-anchor-attlist"/>
</define>

<define name="common-text-anchor-attlist" combine="interleave">
    <optional>
        <attribute name="text:anchor-type">
            <choice>
                <value>page</value>
                <value>frame</value>
                <value>paragraph</value>
                <value>char</value>
                <value>as-char</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Anchor Page Number

Within text documents, the `text:anchor-page-number` attribute specifies the physical page number of an anchor if the drawing object is bound to a page.

```
<define name="common-text-anchor-attlist" combine="interleave">
    <optional>
        <attribute name="text:anchor-page-number">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

9.2.17 Common Drawing Shape Content

Most drawing shapes may contain text content. The text content may contain paragraphs (see section 4.1.2) as well as lists (see section 4.3).

```
<define name="draw-text">
    <zeroOrMore>
        <choice>
            <ref name="text-p"/>
            <ref name="text-list"/>
        </choice>
    </zeroOrMore>
</define>
```

9.2.18 Common Shape Attribute Groups

The following defined attributes are common for all shapes that supports styles and no text.

```
<define name="common-draw-shape-with-styles-attlist">
    <ref name="common-draw-z-index-attlist"/>
    <ref name="common-draw-id-attlist"/>
    <ref name="common-draw-layer-name-attlist"/>
    <ref name="common-draw-style-name-attlist"/>
    <ref name="common-draw-transform-attlist"/>
    <ref name="common-draw-name-attlist"/>
    <ref name="common-text-spreadsheet-shape-attlist"/>
</define>
```

The following defined attributes are common for all shapes that supports styles and text.

```
<define name="common-draw-shape-with-text-and-styles-attlist">
    <ref name="common-draw-shape-with-styles-attlist"/>
    <ref name="common-draw-text-style-name-attlist"/>
</define>
```

9.2.19 Glue Points

Glue points are designated points on the area of a drawing object to which a connector shape can connect. Most drawing objects have four standard glue points at the four edges of the object. Additional glue points may be added to a drawing object by inserting one or more `<draw:glue-point>` elements into a drawing object element. A `<draw:glue-point>` element creates a single user-defined glue point if placed inside a drawing object element, for example, a `<draw:rectangle>` element.

```
<define name="draw-glue-point">
    <element name="draw:glue-point">
        <ref name="draw-glue-point-attlist"/>
        <empty/>
    </element>
</define>
```

ID

The `draw:id` attribute contains the id of the glue point. The id a number and is used inside the `draw:start-glue-point` and `draw:end-glue-point` attributes of a `<draw:connector>` element. The Ids 0 to 3 are reserved for the 4 standard glue points that most drawing objects have. The glue points are numbered clockwise, starting at the top left corner of the shape.

```
<define name="draw-glue-point-attlist" combine="interleave">
    <attribute name="draw:id">
```

```
        <ref name="nonNegativeInteger"/>
    </attribute>
</define>
```

Position

The svg:x and svg:y attributes specifies the position of the glue point. The coordinates are either percentage values relative to the drawing objects center or, if the draw:align attribute is also specified, absolute distance values relative to the edge specified with the draw:align attribute.

```
<define name="draw-glue-point-attlist" combine="interleave">
    <attribute name="svg:x">
        <choice>
            <ref name="distance"/>
            <ref name="percent"/>
        </choice>
    </attribute>
    <attribute name="svg:y">
        <choice>
            <ref name="distance"/>
            <ref name="percent"/>
        </choice>
    </attribute>
</define>
```

Align

The attribute draw:align specifies the alignment behavior of the glue point if the drawing object is resized and the shape edge to which the glue point's position relates. A missing vertical or horizontal position in the attribute's value means that the glue point is horizontally or vertically centered.

```
<define name="draw-glue-point-attlist" combine="interleave">
    <optional>
        <attribute name="draw:align">
            <choice>
                <value>top-left</value>
                <value>top</value>
                <value>top-right</value>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>bottom-left</value>
                <value>bottom-right</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Escape Direction

The attribute draw:escape-direction specifies the direction in which the connection line escapes from the drawing object if a connector connects to the glue point. The value horizontal means the the connection line may escape to the left or to the right, the value vertical means that the connection line may escape up or down. The value auto means that the connection line may escape in all four directions.

```
<define name="draw-glue-points-attlist" combine="interleave">
    <attribute name="draw:escape-direction">
```

```
        <choice>
            <value>auto</value>
            <value>left</value>
            <value>right</value>
            <value>up</value>
            <value>down</value>
            <value>horizontal</value>
            <value>vertical</value>
        </choice>
    </attribute>
</define>
```

9.2.20 Title and Description

The `<svg:title>` and `<svg:desc>` elements specify text-only description strings for graphical objects as specified in §5.4 of [SVG].

The `<svg:title>` element is used as a short accessible name.

```
<define name="svg-title">
    <element name="svg:title">
        <text/>
    </element>
</define>
```

The `<svg:desc>` element is used for the long description in support of accessibility.

```
<define name="svg-desc">
    <element name="svg:desc">
        <text/>
    </element>
</define>
```

See appendix E for guidelines how to use these elements.

The `<svg:title>` and `<svg:desc>` elements can be used with the following drawing shape elements:

- `<draw:rect>`
- `<draw:line>`
- `<draw:polyline>`
- `<draw:polygon>`
- `<draw:regular-polygon>`
- `<draw:path>`
- `<draw:circle>`
- `<draw:ellipse>`
- `<draw:g>`
- `<draw:page-thumbnail>`
- `<draw:frame>`
- `<draw:measure>`
- `<draw:caption>`

- `<draw:connector>`
- `<draw:control>`
- `<dr3d:scene>`
- `<draw:custom-shape>`

It is further supported by layers (see section 9.1.3) and client side image maps (see section 9.3.10).

9.2.21 Event Listeners

Drawing shapes may have event listeners attached. The event listeners that are attached to, for example, a text box or an image, are represented by an event listener element as described in section 12.4. This element is contained within the drawing object element, for example, the `<draw:text-box>` element or the `<draw:image>` element.

9.3 Frames

A **frame** is a rectangular container where that contains enhanced content like text boxes, images or objects. Frames are very similar to regular drawing shapes, but support some features that are not available for regular drawing shapes, like contours, image maps and hyperlinks. In particular, a frame allows to have multiple renditions of an object. That is, a frame may for instance contain an object as well as an image. In this case, the application may choose the content that it supports best. If the application supports the object type contained in the frame, it probably will render the object. If it does not support the object, it will render the image.

In general, an application must not render more than one of the content elements contained in a frame. The order of content elements dictates the document author's preference for rendering, with the first child being the most preferred. This means that applications should render the first child element that it supports. A frame must contain at least one content element. The inclusion of multiple content elements is optional. Application may preserve the content elements they don't render, but don't have to.

Within text documents, frames are also used to position content outside the default text flow of a document.

Frames can contain:

- Text boxes
- Objects represented either in the OpenDocument format or in a object specific binary format
- Images
- Applets
- Plug-ins
- Floating frames

Like the formatting properties of drawing shapes, frame formatting properties are stored in styles belonging to the `graphic` family. The way a frame is contained in a document also is the same as for drawing shapes.

```
<define name="draw-frame">
    <element name="draw:frame">
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-rel-size-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <ref name="presentation-shape-attlist"/>
        <ref name="draw-frame-attlist"/>
        <zeroOrMore>
            <choice>
                <ref name="draw-text-box"/>
                <ref name="draw-image"/>
                <ref name="draw-object"/>
                <ref name="draw-object-ole"/>
                <ref name="draw-applet"/>
                <ref name="draw-floating-frame"/>
                <ref name="draw-plugin"/>
            </choice>
        </zeroOrMore>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <optional>
            <ref name="draw-image-map"/>
        </optional>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <choice>
                <ref name="draw-contour-polygon"/>
                <ref name="draw-contour-path"/>
            </choice>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<draw:frame>` element are:

- Position, Size (relative sizes, see below), Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position – see section 9.2.16
- Presentation class – see section 9.6.1
- Copy frames

The following elements may be contained in the image element:

- Event Listeners – see section 12.4.
- Glue Points – see section 9.2.19.
- Image Map – see section 9.3.10.
- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Contour – see section 9.3.8.

Relative Sizes

For frames, the width and height of the drawing object may be specified as a relative value using the `style:rel-width` and `style:rel-height` attributes. The relative value either is a percentage value, the special value `scale`, or the special value `scale-min`.

The interpretation of relative values depends on the anchor of the drawing object. If the anchor for the drawing object is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the drawing object is in a text box, the percentage value relates to the surrounding text box. In other cases, the percentage values relate to the width of the page or window.

The value `scale` for the width means that the width should be calculated depending on the height, so that the ratio of with and height of the original image or object size is preserved.

The value `scale` for the height means that the height should be calculated depending on the width, so that the ratio of with and height of the original image or object size is preserved.

The value `scale-min` equals the value scale, except that the calculated width or height is a minimum height rather than an absolute one.

To support application that don't support relative with and heights, applications that save the attributes `style:rel-width` or `style:rel-height` should also provide the real width and heights in the `svg:width` and `svg:height`/`fo:min-height` attributes.

```
<define name="common-draw-rel-size-attlist">
    <ref name="common-draw-size-attlist"/>
    <optional>
        <attribute name="style:rel-width">
            <choice>
                <ref name="percent"/>
                <value>scale</value>
                <value>scale-min</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:rel-height">
            <choice>
                <ref name="percent"/>
                <value>scale</value>
                <value>scale-min</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Copy Frames

Multiple frames can be set to display the exact same underlying data: for instance for a company logo, that must appear somewhere on every page, without being part of a header or footer.

A frame can be set to display the contents of another frame, referenced by the `draw:copy-of` attribute. This does not effect style and position information. This is, the frame that has the `draw:copy-of` attribute has its own style and position information and does not use the one of the referenced frame.

```
<define name="draw-frame-attlist" combine="interleave">
    <optional>
        <attribute name="draw:copy-of">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.3.1 Text Box

The `<draw:text-box>`element represents a text box. A text box may be used to place text in a container that is outside of the normal flow of the document.

```
<define name="draw-text-box">
    <element name="draw:text-box">
        <ref name="draw-text-box-attlist"/>
        <zeroOrMore>
            <ref name="text-content"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:text-box>` element are:

- Chain
- Round Corners
- Minimum Height and Width
- Maximum Height and Width

Text boxes don't support contours as described in section 9.3.8 and alternative texts as described in section 9.2.20.

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. To chain text boxes, the attribute `draw:chain-next-name` is used, The value of this attribute is the name of the next text box in the chain. Chained text boxes usually are supported by text documents only.

```
<define name="draw-text-box-attlist" combine="interleave">
    <optional>
        <attribute name="draw:chain-next-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Round Corners

The attribute `draw:corner-radius` specifies the radius of the circle used to round off the corners of the text-box.

```
<define name="draw-text-box-attlist" combine="interleave">
    <optional>
        <attribute name="draw:corner-radius">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

Minimum Height and Width

The `fo:min-height` and `fo:min-width` attributes specify a minimum height or width for a text box. If they are existing, they overwrite the height or width of a text box specified by the `svg:height` and `svg:width` attributes of the surrounding `<draw:frame>` element. Their value can be either a length or a percentage. If the anchor for the text box is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the text box is in a text box, the percentage value relates to the surrounding text box. In other cases, the percentage values relate to the height of the page or window.

```
<define name="draw-text-box-attlist" combine="interleave">
    <optional>
        <attribute name="fo:min-height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:min-width">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Maximum Height and Width

If the width or height of a text box is specified as a minimum width or height (using the `fo:min-width` or `fo:min-height` attributes), then the `fo:max-width` and `fo:max-height` attributes specify a maximum width and height for the text box. When these maximum values are reached, the text box stops increasing in size. The attributes' value can be either a length or a percentage. If the anchor for the text box is in a table cell, the percentage value relates to the size of the surrounding table cell. If the anchor for the text box is in a text box, the percentage value relates to the size of the surrounding text box. In other cases, the percentage values relate to the width or height of the page or window.

```
<define name="draw-text-box-attlist" combine="interleave">
    <optional>
        <attribute name="fo:max-height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:max-width">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

ID

A text box may have an ID. This ID can be used to reference the text box from other elements.

```
<define name="draw-text-box-attlist" combine="interleave">
    <optional>
        <ref name="text-id"/>
    </optional>
</define>
```

9.3.2 Image

The `<draw:image>` element represents an image. An image can be either:

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

or

- Embedded in a document

This element can be an [XLink], in which case the element contains some attributes with fixed values that describe the link semantics.

While the image data may have an arbitrary format, it is recommended that vector graphics are stored in the [SVG] format and bitmap graphics in the [PNG] format.

```
<define name="draw-image">
    <element name="draw:image">
        <ref name="draw-image-attlist"/>
        <choice>
            <ref name="common-draw-data-attlist"/>
            <ref name="office-binary-data"/>
        </choice>
        <ref name="draw-text"/>
    </element>
</define>
```

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

- Image data
- Filter name

Like most other drawing shapes, image drawing shapes may have text content. It is displayed in addition to the image data.

Image Data

The image data can be stored in one of the following ways:

- The image data is contained in an external file. Use the `xlink:href` and associated attributes described below to link to the external file.
- The image data is contained in the `<draw:image>` element. The `<draw:image>` then element contains an `<office:binary-data>` element that contains the image data in BASE64 encoding (as defined in [RFC2045]). In this situation the `xlink:href` attribute is not required.

```
<define name="common-draw-data-attlist" combine="interleave">
    <group>
        <attribute name="xlink:href">
```

```
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <choice>
                    <value>simple</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show" a:defaultValue="embed">
                <choice>
                    <value>embed</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onLoad">
                <choice>
                    <value>onLoad</value>
                </choice>
            </attribute>
        </optional>
    </group>
</define>

<define name="office-binary-data">
    <element name="office:binary-data">
        <ref name="base64Binary"/>
    </element>
</define>
```

Filter Name

If required, the `draw:filter-name` attribute can represent the filter name of the image. This attribute contains the internal filter name that the office application software used to load the graphic.

```
<define name="draw-image-attlist" combine="interleave">
    <optional>
        <attribute name="draw:filter-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.3.3 Objects

A document in OpenDocument format can contain two types of objects, as follows:

- Objects that have an OpenDocument or other XML representation. Objects that have an OpenDocument representation are:
  - Formulas (represented as [MathML])
  - Charts
  - Spreadsheets
  - Text documents

  - Drawings
  - Presentations
- Objects that do not have an XML representation. These objects only have a binary representation, An example for this kind of objects OLE objects (see [OLE]).

The `<draw:object>` element represents objects that have a XML representation. The `<draw:object-ole>` element represents objects that only have a binary representation.

```
<define name="draw-object">
    <element name="draw:object">
        <ref name="draw-object-attlist"/>
        <choice>
            <ref name="common-draw-data-attlist"/>
            <ref name="office-document"/>
            <ref name="math-math"/>
        </choice>
    </element>
</define>

<define name="draw-object-ole">
    <element name="draw:object-ole">
        <ref name="draw-object-ole-attlist"/>
        <choice>
            <ref name="common-draw-data-attlist"/>
            <ref name="office-binary-data"/>
        </choice>
    </element>
</define>
```

The attributes that may be associated with the `<draw:object>` and `<draw:object-ole>` elements are:

- Object data
- Table Change Notifications
- Class Id

Objects do not support transformations as described in section 9.2.15.

Object Data

The object data can be called in one of the following ways:

- The `xlink:href` attribute links to the object representation, as follows:
  - For objects that have an XML representation, the link references the sub package of the object. The object is contained within this sub page exactly as it would as it is a document of its own.
  - For objects that do not have an XML representation, the link references a sub stream of the package that contains the binary representation of the object.

  Application that support objects should support linking to objects that are contained within the same package. They may also support linking to object located outside the package.
- The object data is contained in the `<draw:object>` or `<draw:object-ole>` element, as follows:

- The `<draw:object>` element contains the XML representation of the object, for example, an `<office:document>` or a `<math:math>` element.
- The `<draw:object-ole>` element contains an `<office:binary-data>` element, which contains the binary data for the object in BASE64 encoding.

In these situations, the `xlink:href` attributes are not required.

The `xlink:href` attribute is described in section 9.3.2.

It is recommended to include an image representation of the object into the frame in addition to the object itself.

Notification on Table Change

Some objects, especially charts, may require a notification when a table in the document changes. To enable this notification, use the `draw:notify-on-change-of-table` attribute, which contains the name of the table. This attribute can be associated with the `<draw:object>` element.

```
<define name="draw-object-attlist" combine="interleave">
    <optional>
        <attribute name="draw:notify-on-update-of-ranges">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Class Id

If the embedded object is an OLE object, the `draw:class-id` attribute optionally contains the OLE class id of the object (see also [OLE]).

```
<define name="draw-object-ole-attlist" combine="interleave">
    <optional>
        <attribute name="draw:class-id"/>
    </optional>
</define>
```

9.3.4 Applet

An applet is a small Java-based program that is embedded in a document. The `<draw:applet>` element is based on the `<applet>` tag in [HTML4]. This element must contain either the `draw:code` or `draw:object` attribute.

```
<define name="draw-applet">
    <element name="draw:applet">
        <ref name="draw-applet-attlist"/>
        <optional>
            <ref name="common-draw-data-attlist"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-param"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:applet>` element are:

- Codebase

- Code
- Object
- Archive
- Mayscript

The only element that may be contained in the `<draw:applet>` element is:

- Parameter (see section 9.3.6)

Applets do not support transformations as described in section 9.2.15.

Codebase

The codebase specifies the base IRI for the applet. If this attribute is not specified, then it defaults the same base IRI as for the current document. The codebase is represented be the [XLink] attributes `xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate`. The `xlink:href` attribute is described in section 9.3.2.

Code

The `draw:code` attribute specifies one of the following:

- The name of the class file that contains the compiled applet subclass.
- The path to the class, including the class file itself.

Either this attribute or the `draw:object` attribute is required. The value of this attribute is interpreted in relation to the codebase for the applet.

```
<define name="draw-applet-attlist" combine="interleave">
    <optional>
        <attribute name="draw:code"/>
    </optional>
</define>
```

Object

The `draw:object` attribute specifies a resource that contains a serialized representation of the state of the applet. The serialized data contains the class name of the applet but not the implementation. The value of this attribute is interpreted in relation to the codebase for the applet.

```
<define name="draw-applet-attlist" combine="interleave">
    <optional>
        <attribute name="draw:object"/>
    </optional>
</define>
```

Archive

The `draw:archive` attribute specifies a comma-separated list of URLs for archives that contain classes and other resources that are preloaded.

```
<define name="draw-applet-attlist" combine="interleave">
    <optional>
        <attribute name="draw:archive"/>
    </optional>
```

```
</define>
```

Mayscript

The `draw:mayscript` attribute specifies whether or not the applet can be scripted.

```
<define name="draw-applet-attlist" combine="interleave">
    <optional>
        <attribute name="draw:may-script" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.3.5 Plugins

A plugin is a binary object that is plugged into a document to represent a media-type that usually is not handled natively by office application software. Plugins are represented by the `<draw:plugin>` element

```
<define name="draw-plugin">
    <element name="draw:plugin">
        <ref name="draw-plugin-attlist"/>
        <ref name="common-draw-data-attlist"/>
        <zeroOrMore>
            <ref name="draw-param"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:plugin>` element are:

- Mime type
- Source

The only element that may be contained in the `<draw:plugin>` element is:

- Parameter (see section 9.3.6)

Plugins do not support transformations as described in section 9.2.15.

Mime type

The `draw:mimetype` attribute specifies the MIME type to which this plugin should be registered.

```
<define name="draw-plugin-attlist" combine="interleave">
    <optional>
        <attribute name="draw:mime-type"/>
    </optional>
</define>
```

Source

The [XLink] attributes `xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate` specify the source of the plugin. The `xlink:href` attribute is described in section 9.3.2.

9.3.6 Parameters

The `<draw:param>` element contains parameters that are passed to an applet or plugin when they are initialized.

```
<define name="draw-param">
    <element name="draw:param">
        <ref name="draw-param-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:param>` element are:

- Name
- Value

Name

The `draw:name` attribute specifies the name of a runtime parameter.

```
<define name="draw-param-attlist" combine="interleave">
    <optional>
        <attribute name="draw:name"/>
    </optional>
</define>
```

Value

The `draw:value` attribute specifies the value of the runtime parameter specified by the name.

```
<define name="draw-param-attlist" combine="interleave">
    <optional>
        <attribute name="draw:value"/>
    </optional>
</define>
```

9.3.7 Floating Frame

A floating frame is a frame embedded in a document, which may contain, for example, a text document or spreadsheet. A floating frame is represented by the `<draw:floating-frame>` element.

```
<define name="draw-floating-frame">
    <element name="draw:floating-frame">
        <ref name="draw-floating-frame-attlist"/>
        <ref name="common-draw-data-attlist"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:floating-frame>` element are:

- Source
- Frame Name

Floating frames do not support transformations as described in section 9.2.15.

Source

The [XLink] attributes `xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate` specify the source of the floating frame. The `xlink:href` attribute is described in section 9.3.2.

Frame Name

The `draw:frame-name` specifies the name of the frame. This name can be used as target from within hyperlinks.

```
<define name="draw-floating-frame-attlist" combine="interleave">
    <optional>
        <attribute name="draw:frame-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.3.8 Contour

The `<draw:contour-polygon>` and `<draw:contour-path>` elements may be contained in the following elements:

- `<draw:image>`
- `<draw:object>`
- `<draw:object-ole>`
- `<draw:applet>`
- `<draw:plugin>`
- `<draw:floating-frame>`

These elements describe the contour of an image or object.

```
<define name="draw-contour-polygon">
    <element name="draw:contour-polygon">
        <ref name="common-contour-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-points-attlist"/>
        <empty/>
    </element>
</define>

<define name="draw-contour-path">
    <element name="draw:contour-path">
        <ref name="common-contour-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-path-data-attlist"/>
        <empty/>
    </element>
</define>
```

The elements are similar to the `<draw:polygon>` (see section 9.2.4) and `<draw:path>` (see section 9.2.6) elements, except that they specify a contour rather than a drawing shape. The attributes they support are the ones for the size, the viewbox, the points (contour polygon only) and the path (contour path only).

For the `svg:width` and `svg:height` attributes of the `<draw:contour-polygon>` and `<draw:contour-path>` elements, applications should support pixel lengths (i.e., 20px) in addition to traditional lengths like 2cm.

Recreate on Edit

The `draw:recreate-on-edit` attribute specifies if the contour of the image or object should be recreated automatically when the image or object is edited.

```
<define name="common-contour-attlist" combine="interleave">
    <attribute name="draw:recreate-on-edit">
        <ref name="boolean"/>
    </attribute>
</define>
```

9.3.9 Hyperlinks

Frames may behave like hyperlinks. Such hyperlinks are represented by the `<draw:a>` element, where. the element's content is the frame that should be the source of the link.

This element is an [XLink] and has some attributes with fixed values and describe the semantics of the link.

```
<define name="draw-a">
    <element name="draw:a">
        <ref name="draw-a-attlist"/>
        <ref name="draw-frame"/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:a>` element are:

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

Link Location

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

```
<define name="draw-a-attlist" combine="interleave">
    <attribute name="xlink:href">
        <ref name="anyURI"/>
    </attribute>
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <value>simple</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate" a:defaultValue="onRequest">
            <choice>
                <value>onRequest</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Link Target Frame

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

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 `<draw:a>` element. 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`.

```
<define name="draw-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:target-frame-name">
            <ref name="targetFrameName"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:show">
            <choice>
                <value>new</value>
                <value>replace</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Name

A hyperlink 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. The name does not have to be unique.

This attribute is specified for compatibility with [HTML4] only, where an `<a>` element may serve as a link source and target simultaneously. We strongly recommend that this attribute not be used for any purpose other than to represent links that originally came from a HTML document.

```
<define name="draw-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Title

The `office:title` attribute specifies a short accessible description for hint text.

See appendix E for guidelines how to use this attribute.

```
<define name="draw-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:title">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Server Side Image Map

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

```
<define name="draw-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:server-map" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.3.10 Client Side Image Maps

An client side image map is a collection of hyperlinks that are associated with graphic elements. The image map is a sequence of image map elements. Each image map element associates a hyperlink with an area. The area can be one of the following shapes:

- Rectangular
- Circular
- Polygonal

The `<draw:image-map>` element represents an image map.

```
<define name="draw-image-map">
    <element name="draw:image-map">
        <zeroOrMore>
            <choice>
                <ref name="draw-area-rectangle"/>
                <ref name="draw-area-circle"/>
                <ref name="draw-area-polygon"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

The `<draw:image-map>` element can contain three types of image map elements, which represent the three types of image map areas as follows:

- Rectangular image map elements
- Circular image map elements

• Polygonal image map elements

Image map elements are described in terms of absolute positions. When loading the XML file, the office application must map the image map onto its associated graphical element, for example an image, in its original size. The application then must scale the image map to match the current size of the image, but in the file format the image is always saved in its unscaled version, matching the dimensions of the unscaled image.

Rectangular Image Map Areas

The `<draw:area-rectangle>` element describes a rectangular image map area by an x, y position (`svg:x` and `svg:y` attributes) as well as a width and the height (`svg:width` and `svg:height` attributes). These attributes are required. In addition to this, the attributes described in section 9.3.10:Common Image Map Attributes and Elements are optionally supported.

```
<define name="draw-area-rectangle">
    <element name="draw:area-rectangle">
        <ref name="common-draw-area-attlist"/>
        <attribute name="svg:x">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:y">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:width">
            <ref name="length"/>
        </attribute>
        <attribute name="svg:height">
            <ref name="length"/>
        </attribute>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
    </element>
</define>
```

Circular Image Map Areas

The `<draw:area-circle>` element describes a circular image map area. The additional attributes for circular image maps are described below in the common attributes section.

The required attributes `svg:cx` and `svg:cy` specify the center point of the circle. The required `svg:r` attribute specifies the radius of the circle.

The attributes described in section 9.3.10:Common Image Map Attributes and Elements are optional.

```
<define name="draw-area-circle">
    <element name="draw:area-circle">
        <ref name="common-draw-area-attlist"/>
        <attribute name="svg:cx">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:cy">
```

```
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:r">
            <ref name="length"/>
        </attribute>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
    </element>
</define>
```

Polygonal Image Map Areas

The `<draw:area-polygon>` element describes a polygonal image map area. A polygonal image map area is comprised of the following components:

- A bounding box.
  The bounding box, which is represented in the same way as a rectangular image map area using the `svg:x`, `svg:y`, `svg:width`, and `svg:height` attributes, establishes the reference frame for the view box and the polygon point sequence. The reference frame enables the coordinates to be translated into absolute coordinates.
- A view box.
  The view box attribute `svg:viewBox` establishes a coordinate system for the point sequence. The view box obviates the need to record every point of the point sequence as absolute coordinates with length and unit of measurement.
- A sequence of points in view box coordinates in the `svg:points` attribute.

For more information about how to represent polygons, see section 9.2.4.

The attributes above are required. The attributes described in section 9.3.10:Common Image Map Attributes and Elements are optional.

```
<define name="draw-area-polygon">
    <element name="draw:area-polygon">
        <ref name="common-draw-area-attlist"/>
        <attribute name="svg:x">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:y">
            <ref name="coordinate"/>
        </attribute>
        <attribute name="svg:width">
            <ref name="length"/>
        </attribute>
        <attribute name="svg:height">
            <ref name="length"/>
        </attribute>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-points-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
```

```
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
    </element>
</define>
```

Example: Polygonal image map area

The element shown in the following example defines a triangle that is located in the middle of a 2cm by 2cm image. The bounding box covers an area of 2cm by 1.5cm. One view box unit corresponds to 0.01mm.

```
<draw:area-polygon ...
   svg:x="0" svg:y="0" svg:width="2.0cm" svg:height="2.0cm"
   svg:viewBox="0 0 2000 2000"
   svg:points="400,1500 1600,1500 1000,400"/>
```

Common Image Map Attributes and Elements

In addition to the shape attributes, each image map element can contain the following information:

- Link, including a IRI and link target frame.
- Name.
- Inactive flag.
- Title (short accessible name). Use the `<svg:title>` child element as described in section 9.2.20.
- Long description (in support of accessibility). Use the `<svg:desc>` child element as described in section 9.2.20.
- Events associated with the area. Use the `<office:event-listeners>` child element as described in section 12.4.

Other attributes of the image maps are taken from the HTML image map representation.

Each image map element identifies a hyperlink and uses the [XLink] `href`, `type`, and `show` attributes, and the `office:target-frame-name` attribute to describe the link.

```
<define name="common-draw-area-attlist" combine="interleave">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <choice>
                <value>simple</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="office:target-frame-name">
            <ref name="targetFrameName"/>
        </attribute>
    </optional>
    <optional>
```

```
        <attribute name="xlink:show">
            <choice>
                <value>new</value>
                <value>replace</value>
            </choice>
            </attribute>
    </optional>
</define>
```

The `office:name` attribute assigns a name to each image map element.

```
<define name="common-draw-area-attlist" combine="interleave">
    <optional>
        <attribute name="office:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

The `draw:nohref` attribute declares that the image map element and the associated area is inactive. The IRI that is contained in the image map element is not used.

```
<define name="common-draw-area-attlist" combine="interleave">
    <optional>
        <attribute name="draw:nohref">
            <choice>
                <value>nohref</value>
            </choice>
        </attribute>
    </optional>
</define>
```

9.4 3D Shapes

9.4.1 Scene

The `<dr3d:scene>` element is the only element that can contain three-dimensional shapes. A scene is like a group, but it also defines the projection, lighting, and other render details for the shapes inside the scene.

```
<define name="dr3d-scene">
    <element name="dr3d:scene">
        <ref name="dr3d-scene-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-layer-name-attlist"/>
        <ref name="common-text-spreadsheet-shape-attlist"/>
        <ref name="common-dr3d-transform-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <zeroOrMore>
            <ref name="dr3d-light"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="shapes3d"/>
        </zeroOrMore>
    </element>
</define>

<define name="shapes3d">
    <choice>
        <ref name="dr3d-scene"/>
        <ref name="dr3d-extrude"/>
        <ref name="dr3d-sphere"/>
        <ref name="dr3d-rotate"/>
        <ref name="dr3d-cube"/>
    </choice>
</define>
```

The attributes that may be associated with the `<dr3d:scene>` element are:

- Position, Size, Style, Layer, Z-Index, ID and Caption ID – see section 9.2.15
- Text anchor, table background, draw end position – see section 9.2.16
- Camera vectors
- Projection
- Distance
- Focal length
- Shadow slant
- Shade mode
- Ambient color
- Lighting mode

The elements that may be contained in the `<dr3d:scene>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Light – see section 9.4.2.
- Scene – see section 9.4.1.
- Extrude – see section 9.4.5.
- Sphere – see section 9.4.4.
- Rotate – see section 9.4.6.
- Cube – see section 9.4.3.

Camera Vectors

The camera vectors define a viewing volume. The `dr3d:vrp` attribute specifies the origin, the `dr3d:vpn` attribute points towards the projected objects, and the `dr3d:vup` attribute defines the up vector.

```
<define name="dr3d-scene-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="dr3d:vrp">
            <ref name="vector3D"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:vpn">
            <ref name="vector3D"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:vup">
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

Projection

The `dr3d:projection` attribute specifies the projection. The projection can be perspective or parallel. In perspective mode, objects become smaller in the distance.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:projection">
            <choice>
                <value>parallel</value>
                <value>perspective</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Distance

The `dr3d:distance` attribute specifies the distance between the camera and the object.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:distance">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Focal Length

The `dr3d:focal-length` attribute specifies the length of the focus for the virtual camera of this scene.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:focal-length">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Shadow Slant

The `dr3d:shadow-slant` attribute defines the angle from the three-dimensional scene to a virtual paper on which the shadow is cast.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:shadow-slant">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Shade Mode

The shade mode defines how the lighting is calculated for rendered surfaces

- `flat:` lighting is calculated by one surface normal.
- `phong:` lighting is calculated by interpolating the surface normals over the surface.
- `gouraud:` lighting is calculated by interpolating the color calculated with the surface normals at each edge.
- `draft:` surfaces are not lit and drawn as wireframe only.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:shade-mode">
            <choice>
                <value>flat</value>
                <value>phong</value>
                <value>gouraud</value>
                <value>draft</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Ambient Color

The `dr3d:ambient-color` attribute specifies the color for ambient light. Ambient light is that light that seems to come from all directions.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:ambient-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

Lighting Mode

The attribute `dr3d:lighting-mode` enables or disables the use of lighting in the three-dimensional scene.

```
<define name="dr3d-scene-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:lighting-mode">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

3D Transformation

The value of the `dr3d:transform` attribute is a list of transform definitions, which are applied in the order provided. The individual transform definitions are separated by whitespace. The available types of transform definitions include:

- `matrix (<a> <b> <c> <d> <e> <f> <g> <h> <i> <j> <k> <l>)`, which specifies a transformation in the form of a transformation matrix of six values. matrix(a,b,c,d,e,f,g,h,i,j,k,l) is equivalent to applying the transformation matrix [a b c d e f g h i j k l].
- `translate (<tx> <ty> <tz>)`, which specifies a translation by tx, ty and tz.
- `scale (<sx> <sy> <sz>)`, which specifies a scale operation by sx, sy and sz.
- `rotatex (<rotate-angle> )`, which specifies a rotation by <rotate-angle> degrees along the x-axis.
- `rotatey (<rotate-angle> )`, which specifies a rotation by <rotate-angle> degrees along the y-axis.
- `rotatez (<rotate-angle> )`, which specifies a rotation by <rotate-angle> degrees along the y-axis.

```
<define name="common-dr3d-transform-attlist">
    <optional>
        <attribute name="dr3d:transform"/>
    </optional>
</define>
```

9.4.2 Light

The `<dr3d:light>` element represents a light inside a scene.

This element must be the first element contained in a `<dr3d:scene>` element. There may be several lights, but applications may only support a limited number per scene. A typical limitation are 8 lights per scene.

```
<define name="dr3d-light">
    <element name="dr3d:light">
        <ref name="dr3d-light-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<dr3d:light>` element are:

- Diffuse color
- Direction
- Enabled
- Specular

Diffuse Color

The `dr3d:diffuse-color` attribute specifies the base color that the light is emitting.

```
<define name="dr3d-light-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:diffuse-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

Direction

The `dr3d:direction` attribute specifies the direction in which the light is emitted.

```
<define name="dr3d-light-attlist" combine="interleave">
    <attribute name="dr3d:direction">
        <ref name="vector3D"/>
    </attribute>
</define>
```

Enabled

The `dr3d:enabled` attribute specifies whether or not the light is enabled. If a light is not enabled, it does not emit any light.

```
<define name="dr3d-light-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:enabled">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Specular

The `dr3d:specular` attribute specifies whether or not the light causes a specular reflection on the objects. Applications may evaluate this attribute only for the first light in a scene.

```
<define name="dr3d-light-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:specular">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.4.3 Cube

The `<dr3d:cube>` element represents a three-dimensional cube shape.

```
<define name="dr3d-cube">
    <element name="dr3d:cube">
        <ref name="dr3d-cube-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-layer-name-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-dr3d-transform-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<dr3d:cube>` element are:

- Style, Layer, Z-Index and ID – see section 9.2.15
- Minimum and Maximum Edge

Minimum and Maximum Edge

The attributes `dr3d:min-edge` and `dr3d:max-edge` specify the minimum and maximum edge of the cube in a 3D space.

```
<define name="dr3d-cube-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:min-edge">
            <ref name="vector3D"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:max-edge">
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

9.4.4 Sphere

The `<dr3d:sphere>` element represents a three-dimensional sphere shape.

```
<define name="dr3d-sphere">
    <element name="dr3d:sphere">
        <ref name="dr3d-sphere-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-layer-name-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-dr3d-transform-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<dr3d:sphere>` element are:

- Style, Layer, Z-Index, and ID – see section 9.2.15
- Center
- Size

Center

The `dr3d:center` attribute defines the center of the sphere in a three-dimensional space.

```
<define name="dr3d-sphere-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:center">
```

```
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

Size

The `dr3d:size` attribute defines the size of the sphere in a three-dimensional space.

```
<define name="dr3d-sphere-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:size">
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

9.4.5 Extrude

The `<dr3d:extrude>` element represents a three-dimensional extrude based on a polygon.

```
<define name="dr3d-extrude">
    <element name="dr3d:extrude">
        <ref name="common-draw-path-data-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-layer-name-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-dr3d-transform-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<dr3d:extrude>` element are:

- Viewbox, Style, Layer, Z-Index, and ID – see section 9.2.15
- Path Data – see section 9.2.6

9.4.6 Rotate

The `<dr3d:rotate>` element represents a three-dimensional rotation shape based on a polygon.

```
<define name="dr3d-rotate">
    <element name="dr3d:rotate">
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-path-data-attlist"/>
        <ref name="common-draw-z-index-attlist"/>
        <ref name="common-draw-id-attlist"/>
        <ref name="common-draw-layer-name-attlist"/>
        <ref name="common-draw-style-name-attlist"/>
        <ref name="common-dr3d-transform-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<dr3d:rotate>` element are:

- Viewbox, Style, Layer, Z-Index, and ID – see section 9.2.15
- Path Data – see section 9.2.6

9.5 Custom Shape

A `<draw:custom-shape>` represents a shape that is capable of rendering complex figures. It is offering font work and extrusion functionality. A custom shape may have a geometry that influences its shape. This geometry may be visualized in office application user interfaces, for instance by displaying interaction handles, that provide a simple way to modify the the geometry.

```
<define name="draw-custom-shape">
    <element name="draw:custom-shape">
        <ref name="draw-custom-shape-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <ref name="common-draw-caption-id-attlist"/>
        <optional>
            <ref name="svg-title"/>
        </optional>
        <optional>
            <ref name="svg-desc"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <ref name="draw-glue-point"/>
        </zeroOrMore>
        <ref name="draw-text"/>
        <optional>
            <ref name="draw-enhanced-geometry"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<draw:custom shape>` element are:

- Position, Size, Style, Layer, Z-Index, ID, Caption ID and Transformation – see section 9.2.15.
- Text anchor, table background, draw end position – see section 9.2.16.
- Draw engine
- Draw data

The elements that may be contained in the `<draw:custom-shape>` element are:

- Title (short accessible name) – see section 9.2.20.
- Long description (in support of accessibility) – see section 9.2.20.
- Event listeners – see section 9.2.21.
- Glue points – see section 9.2.19.
- Text – see section 9.2.17.
- Enhanced geometry – see section 9.5.1,

Draw Engine

The optional `draw:engine` attribute specifies the name of a rendering engine that can be used to render the custom shape. The attribute's value is a namespaced token, meaning an identifier prefixed by an XML namespace prefix, just like any attribute or element name in this specification. The drawing engine may get its data either from the `draw:data` attribute, or it may evaluate the `<draw:enhanced-geometry>` child element.

If the `draw:engine` attribute is omitted, the office application's default enhanced custom shape rendering engine will be used. This engine gets its geometry data from the `<draw:enhanced-geometry>` element only.

```
<define name="draw-custom-shape-attlist" combine="interleave">
    <optional>
        <attribute name="draw:engine">
            <ref name="namespacedToken"/>
        </attribute>
    </optional>
</define>
```

Draw Data

The `draw:data` attribute contains rendering engine specific data that describes the geometry of the custom shape. This attribute is only evaluated if a non default rendering engine is specified by the `draw:engine` attribute.

```
<define name="draw-custom-shape-attlist" combine="interleave">
    <optional>
        <attribute name="draw:data">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.5.1 Enhanced Geometry

The `<draw:enhanced-geometry>` element contains the geometry for a `<draw:custom-shape>` element if its `draw:engine` attribute has been omitted.

```
<define name="draw-enhanced-geometry">
    <element name="draw:enhanced-geometry">
        <ref name="draw-enhanced-geometry-attlist"/>
        <zeroOrMore>
            <ref name="draw-equation"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="draw-handle"/>
        </zeroOrMore>
    </element>
</define>
```

The attributes that may be associated with the `<draw:enhanced-geometry>` element are

- Type
- View Box
- Mirror
- Text Rotate Angle

- Extrusion Allowed
- Text Path Allowed
- Concentric Gradient Fill Allowed
- Enhanced Geometry - Extrusion Attributes (see section 9.5.2)
- Enhanced Geometry - Path Attributes (see section 9.5.3)
- Enhanced Geometry - Text Path Attributes (see section 9.5.4)
- Enhanced Geometry - Equation (see section 9.5.5)
- Enhanced Geometry - Handle Attributes (see section 9.5.6)

Type

The `draw:type` attribute contains the name of a shape type. This name can be used to offer specialized user interfaces for certain classes of shapes, like for arrows, smileys, etc.

The shape type is rendering engine dependent and does not influence the geometry of the shape. If the value of the `draw:type` attribute is `non-primitive`, then no shape type is available.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:type" a:defaultValue="non-primitive">
            <ref name="custom-shape-type"/>
        </attribute>
    </optional>
</define>

<define name="custom-shape-type">
    <choice>
        <value>non-primitive</value>
        <ref name="string"/>
    </choice>
</define>
```

View Box

The `svg:viewBox` 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 `<draw:enhanced-path>` element.

The syntax for using this attribute is the same as the [SVG] syntax. The value of the attribute are four numbers separated by white spaces, which define the left, top, right, and bottom dimensions of the user coordinate system.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="svg:viewBox">
            <list>
                <ref name="integer"/>
                <ref name="integer"/>
                <ref name="integer"/>
                <ref name="integer"/>
            </list>
        </attribute>
    </optional>
</define>
```

Mirror

The draw:mirror-vertical and draw:mirror-horizontal attributes specify if the geometry of the shape is to be mirrored.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:mirror-vertical" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:mirror-horizontal" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Text Rotate Angle

The draw:text-rotate-angle attribute specifies the angle by which the text within the custom shape is rotated in addition to the rotation included in the shape's draw:transform attribute.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-rotate-angle" a:defaultValue="0">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Extrusion Allowed

The draw:extrusion-allowed attribute specifies whether the shape is capable to be rendered as extrusion object.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-allowed" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Text Path Allowed

The draw:text-path-allowed attribute specifies if the shape is capable of being rendered as Fontwork object. The text of a Fontwork object is distinguished from normal text objects by being able to render text along or between lines that are specified by the draw:enhanced-path attribute. Fontwork objects are capable to support standard graphic attributes such as fill, shadow and or line styles.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-path-allowed" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Concentric Gradient Fill Allowed

The `draw:concentric-gradient-fill-allowed` attribute specifies if the shape is capable being rendered with a concentric gradient that uses the custom shape path.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:concentric-gradient-fill-allowed"
                    a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.5.2 Enhanced Geometry - Extrusion Attributes

Extrusion

The `draw:extrusion` attribute determines if an extrusion is displayed.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Extrusion Brightness

The `draw:extrusion-brightness` attribute specifies the brightness of a scene.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-brightness" a:defaultValue="33%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Extrusion Depth

The `draw:extrusion-depth` attribute specifies the depth of the extrusion. It takes two space separated values. The first value specifies the depth of the extrusion, the second value specifies the fraction of the extrusion that lies before the shape. It must be in the range [0,1]. A value of 0 is default.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-depth" a:defaultValue="36pt 0">
            <list>
                <ref name="length"/>
                <ref name="double"/>
            </list>
        </attribute>
    </optional>
</define>
```

Extrusion Diffusion

The amount of diffusion reflected by the shape is specified by the `draw:extrusion-diffusion` attribute.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-diffusion" a:defaultValue="0%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Extrusion Number Of Line Segments

The `draw:extrusion-number-of-line-segments` attribute specifies the number of line segments that should be used to display curved surfaces. The higher the number the more line segments are used.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-number-of-line-segments"
                   a:defaultValue="30">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Extrusion Light Face

The `draw:extrusion-light-face` attribute specifies if the front face of the extrusion responds to lightning changes.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-light-face" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Extrusion First Light Harsh

The `draw:extrusion-first-light-harsh` attribute specifies if the primary light is harsh.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-first-light-harsh"
                   a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Extrusion Second Light Harsh

The `draw:extrusion-second-light-harsh` attribute specifies if the secondary light is harsh.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-second-light-harsh"
                   a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Extrusion First Light Level

The `draw:extrusion-first-light-level` attribute specifies the intensity for the first light.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-first-light-level"
                   a:defaultValue="66%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Extrusion Second Light Level

The `draw:extrusion-second-light-level` attribute specifies the intensity for the second light.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-second-light-level"
                   a:defaultValue="66%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Extrusion First Light Direction

The `draw:extrusion-first-light-direction` attribute specifies the direction of the first light.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-first-light-direction"
                   a:defaultValue="(5 0 1)">
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

Extrusion Second Light Direction

The `draw:extrusion-second-light-direction` attribute specifies the direction of the second light.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-second-light-direction"
                   a:defaultValue="(-5 0 1)">
```

```
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

Extrusion Metal

The `draw:extrusion-metal` attribute specifies if the surface of the extrusion object looks like metal.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-metal" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Extrusion Shade Mode

The `dr3d:shade-mode` attribute defines how the lighting is calculated for rendered surfaces

- `flat:` lighting is calculated by one surface normal.
- `phong:` lighting is calculated by interpolating the surface normals over the surface.
- `gouraud:` lighting is calculated by interpolating the color calculated with the surface normals at each edge.
- `draft:` surfaces are not lit and drawn as wireframe only.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:shade-mode" a:defaultValue="flat">
            <choice>
                <value>flat</value>
                <value>phong</value>
                <value>gouraud</value>
                <value>draft</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Extrusion Rotation Angle

The first value of the `draw:extrusion-rotation-angle` specifies the rotation about the x-axis. The second value of the `draw:extrusion-rotation-angle` specifies the rotation about the y-axis. The rotation about the z-axis is specified by the rotate angle of the `draw:transform` attribute.

The order of the rotation is: z-axis, y-axis and then x-axis.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-rotation-angle" a:defaultValue="0 0">
            <list>
                <ref name="double"/>
                <ref name="double"/>
            </list>
        </attribute>
    </optional>
</define>
```

Extrusion Rotation Center

The `draw:extrusion-rotation-center` attribute specifies the position of the rotation center in terms of shape size fractions, if it is omitted then the geometrical center of the shape is used.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-rotation-center">
            <ref name="vector3D"/>
        </attribute>
    </optional>
</define>
```

Extrusion Shininess

The `draw:extrusion-shininess` attribute specifies the shininess of a mirror.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-shininess" a:defaultValue="50%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Extrusion Skew

The `draw:extrusion-skew` attribute specifies the skew amount and skew angle of an extrusion. Skew settings are only applied if the attribute `dr3d:projection` has the value `parallel`.

The first parameter represents the skew amount in percent, the second parameter specifies the skew angle.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-skew" a:defaultValue="50 45">
            <list>
                <ref name="double"/>
                <ref name="double"/>
            </list>
        </attribute>
    </optional>
</define>
```

Extrusion Specularity

The `draw:extrusion-specularity` attribute specifies the specularity of an extrusion object.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-specularity" a:defaultValue="0%">
            <ref name="percent"/>
        </attribute>
    </optional>
```

```
</define>
```

Extrusion Projection Mode

The `dr3d:projection` attribute specifies if the projection mode is perspective or parallel.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:projection" a:defaultValue="parallel">
            <choice>
                <value>parallel</value>
                <value>perspective</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Extrusion Viewpoint

The `draw:extrusion-viewpoint` attribute specifies the viewpoint of the observer as an 3D point. The attribute's value syntax is similar to vector3D, solely a unit is following each parameter. An example for a 3D point is: “(1cm 1cm 0m)”.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-viewpoint"
                   a:defaultValue="3.5cm -3.5cm 25cm">
            <ref name="point3D"/>
        </attribute>
    </optional>
</define>

<define name="point3D">
    <data type="string"/>
</define>
```

Extrusion Origin

The `draw:extrusion-origin` attributes specifies the origin within the bounding box of the shape in terms of the shape size fractions.

The first parameter represents the horizontal origin, a value of -0.5 represents the left side of the shape, a value of 0 represents the center of the shape, a value of 0.5 represents the right side of the shape.

The second parameter represents the vertical origin, a value of -0.5 represents the top side of the shape, a value of 0 represents the center of the shape, a value of 0.5 represents the bottom side of the shape.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-origin" a:defaultValue="0.5 -0.5">
            <list>
                <ref name="double"/>
                <ref name="double"/>
            </list>
        </attribute>
    </optional>
</define>
```

Extrusion Color

The `draw:extrusion-color` attribute specifies if an extrusion color is used. The extrusion color is then defined by the `draw:secondary-fill-color` attribute specified in the custom shape's graphic style.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:extrusion-color" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.5.3 Enhanced Geometry - Path Attributes

Enhanced Path

The `draw:enhanced-path` attribute specifies a path similar to the `svg:d` attribute of the `<svg:path>` element. Instructions such as `moveto`, `lineto`, `arcto` and other instructions together with its parameter are describing the geometry of a shape which can be filled and or stroked. Relative commands are not supported.

The syntax of `draw:enhanced-path` attribute is as follows:

- Instructions are expressed as one character (e.g., a moveto is expressed as an `M`).
- A prefix notation is being used, that means that each command is followed by its parameter.
- Superfluous white space and separators such as commas can be eliminated. (e.g., “`M 10 10 L 20 20 L 30 20`” can also be written: “`M10 10L20 20L30 20`”
- If the command is repeated multiple times, only the first command is required. (e.g., “`M 10 10 L 20 20 L 30 20`” can also be expressed as followed “`M 10 10 L 20 20 30 20`”
- Floats can be used, therefore the only allowable decimal point is a dot (“.”)

The above mentioned rules are the same as specified for the `<svg:path>` element.

A parameter can also have one of the following enhancements:

- A “?” is used to mark the beginning of a formula name. The result of the element's `draw:formula` attribute is used as parameter value in this case.
- If “$” is preceding a integer value, the value is indexing a `draw:modifiers` attribute. The corresponding modifier value is used as parameter value then.

Following notation is used in the table below:

- (): grouping of parameters
- +: 1 or more of the given parameter(s) is required

Example for a custom-shape that uses the draw:enhanced-path to describe a pie-chart whose top right quarter segment is taken out:

```
<draw:custom-shape
   svg:width="10cm" svg:height="10cm" svg:x="0cm" svg:y="0cm">
   <draw:enhanced-geometry svg:viewBox="0 0 10 10"
       draw:enhanced-path="V 0 0 10 10 10 5 5 0 L 5 5 Z N">
```

```
    </draw:enhanced-geometry>
</draw:custom-shape>
```

The following commands are supported:

Command	***Name***	***Parameters***	***Description***
M	moveto	(x y) +	Start a new sub-path at the given (x,y) coordinate. If a moveto is followed by multiple pairs of coordinates, they are treated as lineto.
L	lineto	(x y) +	Draws a line from the current point to (x, y). If multiple coordinate pairs are following, they are all interpreted as lineto.
C	curveto	(x1 y1 x2 y2 x y) +	Draws a cubic Bézier curve from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve.
Z	closepath	(none)	Close the current sub-path by drawing a straight line from the current point to current sub-path's initial point.
N	endpath	(none)	Ends the current set of sub-paths. The sub-paths will be filled by using the “even-odd” filling rule. Other following subpaths will be filled independently.
F	nofill	(none)	Specifies that the current set of sub-paths won't be filled.
S	nostroke	(none)	Specifies that the current set of sub-paths won't be stroked.
T	angle-ellipseto	(x y w h t0 t1) +	Draws a segment of an ellipse. The ellipse is specified by the center(x, y), the size(w, h) and the start-angle t0 and end-angle t1.
U	angle-ellipse	(x y w h t0 t1) +	The same as the “T” command, except that a implied moveto to the starting point is done.
A	arcto	(x1 y1 x2 y2 x3 y3 x y) +	(x1, y1) and (x2, y2) is defining the bounding box of a ellipse. A line is then drawn from the current point to the start angle of the arc that is specified by the radial vector of point (x3, y3) and then counter clockwise to the end-angle that is specified by point (x4, y4).
B	arc	(x1 y1 x2 y2 x3 y3 x y) +	The same as the “A” command, except that a implied moveto to the starting point is done.
W	clockwisearcto	(x1 y1 x2 y2 x3 y3 x y) +	The same as the “A” command except, that the arc is drawn clockwise.
V	clockwisearc	(x1 y1 x2 y2 x3 y3 x y)+	The same as the “A” command, except that a implied moveto to the starting point is done and the arc is drawn clockwise.

Command	Name	Parameters	Description
X	elliptical-quatrantx	(x y) +	Draws a quarter ellipse, whose initial segment is tangential to the x-axis, is drawn from the current point to (x, y).
Y	elliptical-quadranty	(x y) +	Draws a quarter ellipse, whose initial segment is tangential to the y-axis, is drawn from the current point to (x, y).
Q	quadratic-curveto	(x1 y1 x y)+	Draws a quadratic Bézier curve from the current point to (x, y) using (x1, y1) as the control point. (x, y) becomes the new current point at the end of the command.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:enhanced-path">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Path Stretchpoint

The `draw:path-stretchpoint-x` and `draw:path-stretchpoint-y` attributes specifies the stretchpoint of a shape.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:path-stretchpoint-x" a:defaultValue="0">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:path-stretchpoint-y" a:defaultValue="0">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Text Areas

The `draw:text-areas` attribute specifies a list of text areas. The text area is used to position and align the text. If no text area is omitted, the area of the shape itself is used. If a second text area is available it is used for vertical text.

An area consists of four parameters:

The first parameter specifies the left side of the text area.

The second parameter specifies the top side of the text area.

The third parameter specifies the right side of the text area.

The fourth parameter specifies the bottom side of the text area.

A parameter can also have one of the following enhancements:

- A “?” is used to mark the beginning of a formula name. The result of the element's `draw:formula` attribute is used as parameter value in this case.
- If “$” is preceding a integer value, the value is a indexing a `draw:modifiers` attribute. The corresponding modifier value is used as parameter value then.

A example of the draw:text-areas attribute that defines two text areas, including modifier and equation usage, would be: `draw:text-areas=”0 0 100 100 ?Formula1 $1 200 200”`

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-areas">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Glue Points

The `draw:glue-points` attribute specifies a list of object defined glue points. In contradiction to the user defined glue points which are defined by the `<draw:glue-point>` sub-element, the object defined glue point can make use of equations and modifiers.

The first parameter specifies the horizontal position of the glue point.

The second parameter specifies the vertical position of the glue point.

Each parameter can be a float, or it can also have one of the following enhancements:

- A “?” is used to mark the beginning of a formula name. The result of the element's `draw:formula` attribute is used as parameter value in this case.
- If “$” is preceding a integer value, the value is a indexing a `draw:modifiers` attribute. The corresponding modifier value is used as parameter value then.

A example of the draw:glue-points attribute that defines two glue points, including modifier and equation usage, would be: draw:glue-points=”0 ?Formula1 100 $1”

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:glue-points">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Glue Point Type

The `draw:glue-point-type` attribute specifies the glue-point type. If the `draw:glue-points` attribute is also available this attribute is ignored.

- `none`: there are no special object glue points.
- `segments`: a connector will connect with each point of the `draw:enhanced-path attribute`
- `rectangle:` the middle of each side of the shape bound rectangle specifies a object specific glue point

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="draw:glue-point-type" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>segments</value>
                <value>rectangle</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Glue Point Leaving Directions

The `draw:glue-point-leaving-directions` attribute is containing a comma separated list of angles in grad. The angle can be a float value. The position in the list is the same as the to be referenced glue-point of the `draw:glue-points` attribute.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:glue-point-leaving-directions"/>
    </optional>
</define>
```

9.5.4 Enhanced Geometry - Text Path Attributes

Text Path

The `draw:text-path` attribute specifies if text is displayed on a text path.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-path" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Text Path Mode

The `draw:text-path-mode` attribute specifies how the text is drawn.

- `normal`: the text is drawn along the path without scaling.
- `path`: the text is fit to the path.
- `shape`: the text is fit to the bounding box of the shape.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-path-mode" a:defaultValue="normal">
            <choice>
                <value>normal</value>
                <value>path</value>
                <value>shape</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Text Path Scale

The draw:text-path-scale attribute specifies the scaling of the text path.

- path: The text scaling is determined by the length of the path from the draw:enhanced-path attribute.
- shape: The text scaling is determined by the width of a shape.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-path-scale" a:defaultValue="path">
            <choice>
                <value>path</value>
                <value>shape</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Text Path Same Letter Heights

The draw:text-path-same-letter-heights attribute specifies if all letters in the custom shape will have the same height.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:text-path-same-letter-heights"
                   a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Modifiers

The draw:modifiers attribute contains list of modifier values. The modifier can be a float value. In the majority of cases, the draw:modifiers attribute is being used by the draw:handle-position attribute to store the handle position.

```
<define name="draw-enhanced-geometry-attlist" combine="interleave">
    <optional>
        <attribute name="draw:modifiers">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.5.5 Enhanced Geometry – Equation

Equation

The <draw:equation> element can be referenced by handles, text areas, glue points and enhanced paths to calculate values which are dependent to modifier values. Due to the fact that modifier values may changed by interaction it is a convenient way to integrate dynamic values into the shape geometry.

```
<define name="draw-equation">
```

```
    <element name="draw:equation">
        <ref name="draw-equation-attlist"/>
        <empty/>
    </element>
</define>
```

Name

The `draw:name` attribute specifies the name of the equation. The name is not allowed to include spaces.

```
<define name="draw-equation-attlist" combine="interleave">
    <optional>
        <attribute name="draw:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Formula

The `draw:formula` attribute specifies an equation that should be used to evaluate a value. A formula can make use of other formulas or modifier values by function and or modifier reference.

```
number_digit = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'

number = number number_digit | number_digit

identifier = 'pi'|'left'|'top'|'right'|'bottom'|'xstretch'|'ystretch'|
             'hasstroke'|'hasfill'|'width'|'height'|'logwidth'|'logheight'

unary_function = 'abs'|'sqrt'|'sin'|'cos'|'tan'|'atan'|'atan2'
binary_function = 'min'|'max'
ternary_function = 'if'

function_reference = '?' 'a-z,A-Z,0-9' ' '
modifier_reference = '$' '0-9' ' '

basic_expression =
    number |
    identifier |
    function_reference |
    modifier_reference |
    unary_function '(' additive_expression ')' |
    binary_function '(' additive_expression ',' additive_expression ')' |
    ternary_function '(' additive_expression ',' additive_expression ',
                       ' additive_expression ')' | '(' additive_expression ')'

unary_expression = '-' basic_expression

multiplicative_expression =
                          basic_expression |
                          multiplicative_expression '*' basic_expression |
                          multiplicative_expression '/' basic_expression

additive_expression =
                    multiplicative_expression |
                    additive_expression '+' multiplicative_expression |
                    additive_expression '-' multiplicative_expression
```

identifier	***Description***
left	The left position of the `svg:viewBox` attribute has to be used.
top	The top position the `svg:viewBox` attribute has to be used.
right	The right position the `svg:viewBox` attribute has to be used.
bottom	The bottom position the `svg:viewBox` attribute has to be used.
xstretch	The value of `draw:path-stretchpoint-x` is used.
ystretch	The value of `draw:path-stretchpoint-y` is used.
hasstroke	If the shape has a line style, a value of 1 is used.
hasfill	If the shape has a fill style, a value of 1 is used.
width	The width of the svg:viewBox is used.
height	The height of the svg:viewBox is used.
logwidth	The width of the svg:viewBox in 1/100th mm is used.
logheight	The height of the svg:viewBox in 1/100th mm is used.

A example for the `draw:formula` attribute would be: `draw:formula`=”width+10-$0” If the value of the first modifier value is “100” and the width of the `svg:viewbox` is “10000”, then the result of the above formula would be 10000 + 10 – 100 = 9910

```
<define name="draw-equation-attlist" combine="interleave">
    <optional>
        <attribute name="draw:formula">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.5.6 Enhanced Geometry - Handle Attributes

Handle

The `<draw:handle>` element specifies a single interaction handle.

```
<define name="draw-handle">
    <element name="draw:handle">
        <ref name="draw-handle-attlist"/>
        <empty/>
    </element>
</define>
```

Handle Mirror Vertical

The `draw:handle-mirror-vertical` attribute specifies if the x position of the handle is mirrored.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-mirror-vertical" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Handle Mirror Horizontal

The `draw:handle-mirror-horizontal` attribute specifies if the y position of the handle is mirrored.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-mirror-horizontal" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Handle Switched

The `draw:handle-switched` attribute specifies if the handle directions are swapped if the shape height is higher than the shape width.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-switched" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Handle Position

The `draw:handle-position attribute` specifies the position of the handle and consists of two parameters.

Each parameter can be a float or it can have one of the following enhancements:

- A “?” is used to mark the beginning of a formula name. The result of the element's `draw:formula` attribute is used as parameter value in this case.
- If “$” is preceding a integer value, the value is a indexing a `draw:modifiers` attribute. The corresponding modifier value is used as parameter value then.
- Instead of a number a parameter can also be one of the following constants:

Constant	***Description***
left	The value of the draw:coordinate-origin-x attribute has to be used.
top	The value of the draw:coordinate-origin-y attribute has to be used.
right	The value of the draw:coordinate-origin-x attribute + the value of the draw:coordinate-width has to be used.

Constan t	***Description***
bottom	The value of the draw:coordinate-origin-y attribute + the value of the draw:coordinate-height has to be used.
xstretch	The value of draw:path-stretchpoint-x is used.
ystretch	The value of draw:path-stretchpoint-y is used.
hasstroke	If the shape has a line style, a value of 1 is used.
hasfill	If the shape has a fill style, a value of 1 is used.
width	The width of the svg:viewBox is used.
height	The height of the svg:viewBox is used.
logwidth	The width of the svg:viewBox in 1/100th mm is used.
logheight	The height of the svg:viewBox in 1/100th mm is used.

The `draw:handle-position` attribute specifies the position of the handle. If the `draw:handle-polar` attribute is not set, the first parameter of the `draw:handle-position` attribute specifies the horizontal handle position, the vertical handle position is described by the second parameter. If the `draw:handle-polar` attribute is set, then the handle is a polar handle and the first parameter of the draw:handle-position attribute specifies the angle in grad, the handle radius is specified by the second parameter. A example for the `draw:handle-position` attribute is: `draw:handle-position` = "left $5"

```
<define name="draw-handle-attlist" combine="interleave">
    <attribute name="draw:handle-position">
        <ref name="string"/>
    </attribute>
</define>
```

Handle Range X Minimum

The `draw:handle-range-x-minimum` attribute specifies the horizontal minimum value of the range the handle can be moved within. The syntax for the attribute is the same as for the attribute `draw:handle-position`, except that only the first parameter is used. Example for this attribute declaring a minimum value that results from the first formula equation: `draw:handle-range-x-minimum` = ”?Formula1”

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-range-x-minimum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Range X Maximum

The `draw:handle-range-x-maximum` attribute specifies the horizontal maximum value of the range the handle can be moved within. The syntax for the attribute is the same as for the attribute `draw:handle-range-x-minimum`.

```
<define name="draw-handle-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="draw:handle-range-x-maximum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Range Y Minimum

The `draw:handle-range-y-minimum` attribute specifies the vertical minimum value of the range the handle can be moved within. The syntax for the attribute is the same as for the attribute `draw:handle-range-x-minimum`.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-range-y-minimum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Range Y Maximum

The `draw:handle-range-y-maximum` attribute specifies the vertical maximum value of the range the handle can be moved within. The syntax for the attribute is the same as for the attribute `draw:handle-range-x-minimum`.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-range-y-maximum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Polar

The `draw:handle-polar` attribute specifies that the handle is a polar handle. The syntax for this attribute is the same as for the attribute `draw:handle-position`. The first parameter specifies the horizontal center position, the vertical center position is specified by the second parameter. If this attribute is set, the attributes `draw:handle-range-x` and `draw:handle-range-y` are ignored, instead the attributes `draw:handle-radius-range-minimum` and `draw:handle-radius-range-maximum` can be used.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-polar">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Radius Range Minimum

If the attribute `draw:handle-radius-range-minimum` is set, it specifies the minimum radius range that can be used for a polar handle. The syntax is the same as for the attribute `draw:handle-range-x-minimum`.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-radius-range-minimum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Handle Radius Range Maximum

If the attribute `draw:handle-radius-range-maximum` is set, it specifies the maximum radius range that can be used for a polar handle. The syntax is the same as for the attribute `draw:handle-range-x-minimum`.

```
<define name="draw-handle-attlist" combine="interleave">
    <optional>
        <attribute name="draw:handle-radius-range-maximum">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.6 Presentation Shapes

Presentation shapes are special text box, image, object or thumbnail drawing 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. If a draw page's presentation layout (see section 14.15) is changed, all presentation shapes are adapted automatically.

Standard drawing shapes can also be used in presentations. The `presentation:class` attribute distinguishes presentation shapes from drawing shapes. Unlike presentation shapes, standard drawing shapes are not adapted if the presentation page layout is changed.

9.6.1 Common Presentation Shape Attributes

The attributes described in this section are common to all presentation shapes.

Style

Presentation shapes can have styles from the style family `presentation` assigned to them. A presentation shape can be distinguished from a drawing shape by checking whether it has a `presentation:style-name` attribute. A drawing shape uses a `draw:style-name` attribute with a style from the `graphic` 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 formatting properties in this style and its optional parent styles are used to format this shape. See also section 9.2.15.

Class

The `presentation:class` attribute classifies presentation shapes by their usage within a draw page (for instance as title or outline). The following classes exist:

- `title`: Titles are standard text shapes.
- `outline`: Outlines are standard text shapes.

- `subtitle`: Subtitles are standard text shapes.
- `text`: Presentation texts are standard text shapes.
- `graphic`: Presentation graphics are standard graphic shapes
- `object`: Presentation objects are standard object shapes.
- `chart`: Presentation charts are standard object shapes.
- `table`: Presentation tables are standard object shapes.
- `orgchart`: Presentation organization charts are standard object shapes.
- `page`: Presentation pages are used on notes pages.
- `notes`: Presentation notes are used on notes pages.
- `handout`: Presentation handouts are placeholder for the drawing page in a handout page.

The next four classes can be used only for drawing shapes that are contained in master pages. Depending on the settings of the page (see section 15.36), they are displayed automatically on drawing pages that use the master page.

- `header`: The drawing shape is used as a header. Header shapes are standard text shapes.
- `footer`: The drawing shape is used as a footer. Footer shapes are standard text shapes.
- `date-time`: The drawing shape is used as a date and/or time shape. Date and Time shapes are standard text shapes.
- `page-number`: The drawing shape is used as a page number shape. Page Number shapes are standard text shapes.

```
<define name="presentation-shape-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:class">
            <ref name="presentation-classes"/>
        </attribute>
    </optional>
</define>
<define name="presentation-classes">
    <choice>
        <value>title</value>
        <value>outline</value>
        <value>subtitle</value>
        <value>text</value>
        <value>graphic</value>
        <value>object</value>
        <value>chart</value>
        <value>table</value>
        <value>orgchart</value>
        <value>page</value>
        <value>notes</value>
        <value>handout</value>
        <value>header</value>
        <value>footer</value>
        <value>date-time</value>
        <value>page-number</value>
    </choice>
</define>
```

Placeholder

The `presentation:placeholder` attribute defines if a shape is a placeholder or a presentation object with actual content.

```
<define name="presentation-shape-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:placeholder">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

User-Transform

The `presentation:user-transformed` attribute specifies whether the size and position of the shape is set by the user or is set by the corresponding presentation shape on the master page.

```
<define name="presentation-shape-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:user-transformed">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.7 Presentation Animations

In a presentation document, shapes can be animated. Each presentation page can have an optional `<presentation:animations>` element, which is a container for animation effects. The animation is executed when the page is displayed during a presentation.

This specification allows multiple effects for one and the same shape within a page. Applications may have restrictions regarding the number and combination of effects applicable to a shape, for instance may support only one show and one hide effect per shape with an additional show and hide text and one dim and sound effect.

```
<define name="presentation-animations">
    <element name="presentation:animations">
        <zeroOrMore>
            <choice>
                <ref name="presentation-animation-elements"/>
                <ref name="presentation-animation-group"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
<define name="presentation-animation-elements">
    <choice>
        <ref name="presentation-show-shape"/>
        <ref name="presentation-show-text"/>
        <ref name="presentation-hide-shape"/>
        <ref name="presentation-hide-text"/>
        <ref name="presentation-dim"/>
        <ref name="presentation-play"/>
    </choice>
</define>
```

9.7.1 Sound

The element `<presentation:sound>` may be contained in all animation effect elements that support sounds. The sound file referenced by the XLink attributes is played when the effect is executed.

```
<define name="presentation-sound">
    <element name="presentation:sound">
        <ref name="presentation-sound-attlist"/>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <choice>
                    <value>simple</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onRequest">
                <choice>
                    <value>onRequest</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show">
                <choice>
                    <value>new</value>
                    <value>replace</value>
                </choice>
            </attribute>
        </optional>
        <empty/>
    </element>
</define>
```

The attribute that may be associate with the <`presentation:sound`> element is:

- Play full

Play Full

If the value of the attribute `presentation:play-full` is `true`, the next effect starts after the sound is played. If the value of this attribute is `false`, the next effect starts when the current effect is finished.

```
<define name="presentation-sound-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:play-full">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.7.2 Show Shape

The element `<presentation:show-shape>` makes a shape visible. If there is a `<presentation:show-shape>` element for one shape, this shape is automatically invisible before the effect is executed.

```
<define name="presentation-show-shape">
    <element name="presentation:show-shape">
        <ref name="common-presentation-effect-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<presentation:show-shape>` element are:

- Shape
- Effect
- Direction
- Speed
- Delay
- Start Scale
- Path

Shape

The attribute `draw:shape-id` specifies the shape of this effect using a shape ID.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <attribute name="draw:shape-id">
        <ref name="IDREF"/>
    </attribute>
</define>
```

Effect

The attribute `presentation:effect` specifies the type of effect.

- `none`: no effect is used.
- `fade`: the shape fades from its visible or hidden state to a hidden or visible state.
- `move`: the shape moves from or to its final position.
- `stripes`: the shape is faded in or out by drawing or removing horizontal or vertical stripes that change their size.
- `open`: the shape is drawn or removed line by line, either horizontally or vertically, starting at the center of the shape.
- `close`: the shape is drawn or removed line by line, either horizontally or vertically, starting at the edge of the shape.

- `dissolve`: the shape is faded in or out by drawing or removing small blocks in a random fashion.
- `wavyline`: the shape is faded in our out by drawing or removing small blocks in a snake like fashion.
- `random`: an effect is chosen at random to fade the shape in or out.
- `lines`: the shape is faded in our out by drawing or removing line by line, either horizontally or vertically, in a random fashion.
- `laser`: this effect is only available for text shapes; the characters of the text are moved one by one from the top edge of the screen to their final position.
- `appear`: the shape is faded in by just switching its state from invisible to visible.
- `hide`: the shape is faded out by just switching its state from visible to invisible.
- `move-short`: like the move effect, but the moving shape is clipped to its final bounding rectangle during fade.
- `checkerboard`: the shape is faded in or out by drawing or removing checkerboard like blocks that increase in size over time.
- `rotate`: the shape rotates horizontally or vertically for a short amount of time during this effect.
- `stretch`: the shape is faded in or out by changing its size during this effect.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:effect" a:defaultValue="none">
            <ref name="presentationEffects"/>
        </attribute>
    </optional>
</define>
<define name="presentationEffects">
    <choice>
        <value>none</value>
        <value>fade</value>
        <value>move</value>
        <value>stripes</value>
        <value>open</value>
        <value>close</value>
        <value>dissolve</value>
        <value>wavyline</value>
        <value>random</value>
        <value>lines</value>
        <value>laser</value>
        <value>appear</value>
        <value>hide</value>
        <value>move-short</value>
        <value>checkerboard</value>
        <value>rotate</value>
        <value>stretch</value>
    </choice>
</define>
```

Direction

The attribute `presentation:direction` specifies the direction of the effect. This is relevant for some effects only.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:direction" a:defaultValue="none">
            <ref name="presentationEffectDirections"/>
        </attribute>
    </optional>
</define>
<define name="presentationEffectDirections">
    <choice>
        <value>none</value>
        <value>from-left</value>
        <value>from-top</value>
        <value>from-right</value>
        <value>from-bottom</value>
        <value>from-center</value>
        <value>from-upper-left</value>
        <value>from-upper-right</value>
        <value>from-lower-left</value>
        <value>from-lower-right</value>
        <value>to-left</value>
        <value>to-top</value>
        <value>to-right</value>
        <value>to-bottom</value>
        <value>to-upper-left</value>
        <value>to-upper-right</value>
        <value>to-lower-right</value>
        <value>to-lower-left</value>
        <value>path</value>
        <value>spiral-inward-left</value>
        <value>spiral-inward-right</value>
        <value>spiral-outward-left</value>
        <value>spiral-outward-right</value>
        <value>vertical</value>
        <value>horizontal</value>
        <value>to-center</value>
        <value>clockwise</value>
        <value>counter-clockwise</value>
    </choice>
</define>
```

Speed

The attribute `presentation:speed` specifies the speed of the effect.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:speed" a:defaultValue="medium">
            <ref name="presentationSpeeds"/>
        </attribute>
    </optional>
</define>
<define name="presentationSpeeds">
    <choice>
        <value>slow</value>
        <value>medium</value>
        <value>fast</value>
    </choice>
</define>
```

Delay

The attribute `presentation:delay` specifies the delay before a presentation effect starts after the previous one has been finished.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:delay">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

Start Scale

Some effects scale a shape during execution of the effect. The attribute `presentation:start-scale` specifies the start size of the shape as a percentage of its original size.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:start-scale" a:defaultValue="100%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Path

The attribute `presentation:path-id` applies to `move` effects. The attribute specifies the shape-id of a polygon shape. The effect moves along the lines of the specified polygon. The referenced polygon is not visible during the presentation.

```
<define name="common-presentation-effect-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:path-id"/>
    </optional>
</define>
```

9.7.3 Show Text

The element `<presentation:show-text>` makes the text of a shape visible. If there is a `<show-text>` element for one shape, the text of the shape is automatically invisible before the effect is executed.

```
<define name="presentation-show-text">
    <element name="presentation:show-text">
        <ref name="common-presentation-effect-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the <`presentation:show-text`> element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see section 9.7.2

9.7.4 Hide Shape

The element `<presentation:hide-shape>` makes a shape invisible.

```
<define name="presentation-hide-shape">
    <element name="presentation:hide-shape">
        <ref name="common-presentation-effect-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the <`presentation:hide-shape`> element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see section 9.7.2

9.7.5 Hide Text

The element `<presentation:hide-text>` makes the text of a shape invisible.

```
<define name="presentation-hide-text">
    <element name="presentation:hide-text">
        <ref name="common-presentation-effect-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the <`presentation:hide-text`> element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see section 9.7.2

9.7.6 Dim

The element `<presentation:dim>` fills a shape in a single color.

```
<define name="presentation-dim">
    <element name="presentation:dim">
        <ref name="presentation-dim-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the <`presentation:dim`> element are:

- Shape – see section 9.7.2
- Color

```
<define name="presentation-dim-attlist" combine="interleave">
    <attribute name="draw:shape-id">
        <ref name="IDREF"/>
    </attribute>
</define>
```

Color

The attribute `draw:color` specifies the color that is used to fill the shape when the shape is dimmed.

```
<define name="presentation-dim-attlist" combine="interleave">
    <attribute name="draw:color">
        <ref name="color"/>
    </attribute>
</define>
```

9.7.7 Play

The element `<presentation:play>` starts the animation of a shape that supports animation.

```
<define name="presentation-play">
    <element name="presentation:play">
        <ref name="presentation-play-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the <`presentation:play`> element are:

- Shape ID and Speed – see section 9.7.2

```
<define name="presentation-play-attlist" combine="interleave">
    <attribute name="draw:shape-id">
        <ref name="IDREF"/>
    </attribute>
    <optional>
        <attribute name="presentation:speed" a:defaultValue="medium">
            <ref name="presentationSpeeds"/>
        </attribute>
    </optional>
</define>
```

9.7.8 Effect groups

The element `<presentation:animation-group>` allows to specify that multiple effects should happen at the same time.

```
<define name="presentation-animation-group">
    <element name="presentation:animation-group">
        <zeroOrMore>
            <ref name="presentation-animation-elements"/>
        </zeroOrMore>
    </element>
</define>
```

9.8 SMIL Presentation Animations

This chapter describes [SMIL20] based shape animations for presentation documents. This kind of animations can be used instead of the ones specified by the `<presentation:animations>` elements if one of the following items is required:

- Multiple animations per shape.
- A mixture of animations starting on user interaction and starting automatically per page.

- Multiple animations running at the same time.
- Additional effects 'programmed' in XML by combining basic animation elements.
- Document transformations to SVG including [SMIL20].

9.8.1 Recommended Usage Of SMIL

The following sections describe the usage of SMIL animation elements that enables an office application to present the animation elements in a simple and easy to use UI to the user. This UI may contain a single main sequence of effects, and in addition to this, multiple sequences of effects that are started as interactions on drawing shapes. An *effect* is a combination of one or more animation elements that animate a single shape and or a shape's paragraphs.

It is recommended, that in user interfaces, effects can be created by using presets that have localized and meaningful names. This way, the user will not work on a hierarchy of SMIL animation elements, but on one dimensional lists of effects, which are much easier to handle for the office application users.

Slide Animation

Each `<draw:page>` element may optionally have an `<anim:par>` element that defines the animation of that page during a running slideshow. This `<anim:par>` element should contain one `<anim:seq>` element which is the main sequence for shape effects and zero or more `<anim:seq>` elements that define interactive sequences for shapes that contain animation interactions. The animation elements are executed after the slide has executed its initial transition.

Main Sequence

The main sequence is a `<anim:seq>` element which contains the effects that should start after the slide has executed its initial transition. Since this is a sequential container, its child nodes are executed one after each other. If a child node's `smil:begin` attribute has the value `indefinite`, then the execution is stalled until the user advances the slideshow by a mouse or key interaction.

The first level of child nodes in the main sequence should be `<anim:par>` elements that group animation elements that are started with the same user interaction. The second level of child nodes should be `<anim:par>` elements that group animations elements that start at the same time. The third level of child nodes should be `<anim:par>` elements that group the animation elements for a single effect.

The following example shows a main sequence with the effects A, B, C and D. Effect A is started on user interaction, effect B is started simultaneously with A. Effect C is started 4 seconds after the effects A and B. Effect D is started on the next user interaction:

```
<amin:par> <!-- timing root-->
   <anim:seq> <!-- main sequence-->
      <anim:par smil:begin="indefinite">
         <!-- first user interaction -->
         <anim:par smil:begin="0s" smil:dur="4s">
            <!-- first group of effects to execute -->
            <anim:par> <!-- effect a -->
               <!-- nodes for effect a-->
            </anim:par>
            <anim:par> <!-- effect b -->
               <!-- nodes for effect b-->
```

```
                </anim:par>
            </anim:par>
            <anim:par smil:begin="4s">
                <!-- second group of effects to execute -->
                <anim:par> <!-- effect c -->
                    <!-- nodes for effect c-->
                </anim:par>
            </anim:par>
        </anim:par>
        <anim:par>
            <!-- second user interaction-->
            <anim:par smil:begin="indefinite">
                <!-- first group of effects to execute -->
                <anim:par> <!-- effect d -->
                    <!--- nodes for effect d-->
                </anim:par>
            </anim:par>
        </anim:par>
    </anim:seq>
</anim:par>
```

Interactive Sequence

An interactive sequence is a `<anim:seq>` element that should have the same structure as a main sequence. The only difference is that the `<anim:par>` element in the first level has a `smil:begin` attribute with a value like `[shape-id].click`, where `[shape-id]` identifies a drawing shapes by its `draw:id` attribute. These animation elements are triggered when the user interacts with the element defined by `[shape-id]`.

9.8.2 Document Dependent SMIL Animation Attribute Values

This section describes the attribute values of the document type dependent attributes specified in section 13 if they are used within presentation documents.

Iteration Target Element

For presentation documents, the `smil:targetElement` attribute of the `<anim:iterate>` element (see section 13.4.4) can reference drawing shape or paragraph elements. If the `anim:sub-item` attribute of `<anim:iterate>` has the value `whole`, the iteration includes the drawing shape's background and its text. If the `anim:sub-item` attribute's value is `text`, only the shape's text is iterated.

Iteration Type

For presentation documents, the `anim:iterate-type` attribute of the `<anim:iterate>` element (see section 13.4.4) can have the following values:

- `by-paragraph:` the target shape is iterated by paragraphs.
- `by-word`: the target shape or paragraph is iterated by words.
- `by-letter`: the target shape or paragraph is iterated by letters.

Target Element

For presentation documents, the `smil:targetElement` specified in section 13.3.1 can reference drawing shapes by their `draw:id` attribute value and paragraphs by their `text:id` attribute value.

Target Attribute

For presentation documents, the `smil:attributeName` attribute specified in section 13.3.1 can have the following values:

- `x`: animates the elements x position, values are given in screen space where 0 is the left edge and 1 is the right edge.
- `y`: animates the elements y position, values are given in screen space where 0 is the top and 1 is the bottom.
- `width`: animates the elements width, values are given in screen space where 0 is no width and 1 is the same width as the screen.
- `height`: animates the elements height, values are given in screen space where 0 is no height and 1 is the same height as the screen.
- `color:` animates the elements color, this animates both fill,line and char color. Values can be RGB or HSL
- `rotate:` animates the elements rotation, this animates both the shapes and text animation.
- `skewX:` animates the elements horizontal skew.
- `fillColor`: animates the elements fill color.
- `fillStyle`: animates the elements fill style.
- `lineColor`: animates the elements line color.
- `lineStyle`: animates the elements line style.
- `charColor`: animates the elements char color.
- `charWeight`: animates the elements text weight.
- `charUnderline`: animates the elements text underline.
- `charFontName`: animates the elements text font.
- `charHeight`: animates the elements text height.
- `charPosture`: animates the elements text posture.
- `visibility`: animates the elements visibility.
- `opacity`: animates the elements opacity.

Target Element Sub Item

For presentation documents, the `anim:sub-item` attribute specified in section 13.3.1 can have the following values:

- `whole`: animates both the shape and its text.

- `background`:animates only the shapes background and not its text.
- `text`: animates only the text.

Formula

For presentation documents, the `anim:formula` attribute specified in section 13.3.2 may contain the following additional identifiers:

- e,: this is the Euler constant.
- `x`: this is the animated elements left edge in screen space where 0 is the left edge of the screen and 1 is the right edge.
- `y`: this is the animated elements top edge in screen space, where 0 is the top edge of the screen and 1 is the bottom edge.
- `width`: this is the animated elements width in screen space, where 0 is no width and 1 is the screens width.
- `height`: this is the animated elements height in screen space, where 0 is no height and 1 is the screens height.

Command

For presentation documents, The `anim:command` attribute of the <anim:command> element (see section 13.6.1) can have the following values:

- `custom`: the command is user defined.
- `verb`: the command targets an OLE2 shape. The parameter `verb` is the verb number that will be executed at the OLE2 shape.
- `play`: the command targets a media shape and starts its playback. The optional parameter `media-time` defines the playback start time in seconds. If this parameter is not set, playback starts at the last position.
- `toggle-pause`: the command targets a media shape and toggles its playback state from play to paused or from paused to play.
- `stop`: the command targets a media shape and stops its playback.
- `stop-audio`: the command has no target and stops all running audio playback.

9.8.3 SMIL Presentation Animation Attributes

The attributes described in this section can be attached to the animation elements described in section 13.4, 13.5 and 13.6 if they are used inside presentation documents. They don't influence the actual animation behavior, but help office application user interfaces in presenting animation effect settings to the user.

Node Type

The `presentation:node-type` attribute specifies a node type for an animation element. This attribute does not alter the element's behavior but helps the application to quickly identify an elements purpose inside an animation element hierarchy. The value of this attribute can be:

- `default`: this animation element has no special meaning for the application. This is the default setting.
- `on-click`: this animation element is the root element of an effect that starts with a user click.
- `with-previous`: this animation element is the root element of an effect that starts with the previous effect.
- `after-previous`: this animation element is the root element of an effect that starts after the previous effect.
- `timing-root`: this animation element is the root element for the animation of a page.
- `main-sequence`: this animation element is the root element for the main sequence of effects of a page
- `interactive-sequence`: this animation element is the root element for a sequence of effects that are started when the user interactively clicks on a special element inside a page.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:node-type" a:defaultValue="default">
            <choice>
                <value>default</value>
                <value>on-click</value>
                <value>with-previous</value>
                <value>after-previous</value>
                <value>timing-root</value>
                <value>main-sequence</value>
                <value>interactive-sequence</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Preset Id

The `presentation:preset-id` attribute specifies the name of the preset that was used to create this animation element.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:preset-id">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Preset Sub Type

The `presentation:preset-sub-type` attribute specifies the sub type of the preset that was used to create this animation element.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:preset-sub-type">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Preset Class

The `presentation:preset-class` attribute specifies the class of the preset that was used to create this animation element. The value of this attribute can be:

- `custom`: the preset was a user defined one. This is the default setting.
- `entrance`: the preset was an entrance effect.
- `exit`: the preset was an exit effect.
- `emphasis`: the preset was an emphasis effect.
- `motion-path`: the preset was a motion path.
- `ole-action`: the preset was an ole action.
- `media-call`: the preset was a media call.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:preset-class" a:defaultValue="custom">
            <choice>
                <value>custom</value>
                <value>entrance</value>
                <value>exit</value>
                <value>emphasis</value>
                <value>motion-path</value>
                <value>ole-action</value>
                <value>media-call</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Master Element

The `presentation:master-element` attribute specifies the id of an animation element. Office application user interfaces may only display animation elements that don't have a `presentation:master-element` attribute, and may consider the ones that have a `presentation:master-element` to be a part of the animation element that is referenced.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:master-element">
            <ref name="IDREF"/>
        </attribute>
    </optional>
</define>
```

Group Id

The `presentation:group-id` attribute specifies a group id. This id can be used to group animation elements within the user interface, where a group consists of all animation elements that have the same group id. This can be used for instance to group the animation elements that animate the paragraphs of a single shape.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:group-id">
```

```
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

9.9 Presentation Events

Many objects inside a presentation document support special presentation events. For example, a user can advance the presentation one frame when he clicks on an object with a corresponding event. Presentation events are contained with a graphic object's event listener table. See section 9.2.21 for details.

```
<define name="presentation-event-listener">
    <element name="presentation:event-listener">
        <ref name="presentation-event-listener-attlist"/>
        <optional>
            <ref name="presentation-sound"/>
        </optional>
    </element>
</define>
```

Event Name

The `script:event-name` attribute specifies the name of the event. See section 12.4.1 for details.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <attribute name="script:event-name">
        <ref name="string"/>
    </attribute>
</define>
```

Event Action

The kind of action that is executed when the event is triggered can be selected with the `presentation:action` attribute. The following actions are available:

- `none`: no action is performed when this event is triggered.
- `previous-page`: the presentation jumps to the previous page.
- `next-page`: the presentation jumps to the next page.
- `first-page`: the presentation jumps to the first page of the current document.
- `last-page`: the presentation jumps to the last page of the current document.
- `hide`: the object that contains this event is hidden if the event is triggered.
- `stop`: if a slide show is active, it will be stopped.
- `execute`: another application is lunched when this event is triggered. The application can be set with an xlink.
- `show`: the target of an URL is opened when this event is triggered. The URL can be set with an xlink.
- `verb`: if the object that contains this event supports the execution of [OLE] verbs, the verb with the id set in the `presentation:verb` attribute is executed.

- `fade-out`: the object that contains this event is faded out when this event is triggered. The attributes `presentation:effect`, `presentation:direction`, `presentation:speed` and `presentation:start-scale` can be used to set the effect.
- `sound`: an audio effect is started when the effect is triggered. The audio effect is described by a `<presentation:sound>` child element.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <attribute name="presentation:action">
        <choice>
            <value>none</value>
            <value>previous-page</value>
            <value>next-page</value>
            <value>first-page</value>
            <value>last-page</value>
            <value>hide</value>
            <value>stop</value>
            <value>execute</value>
            <value>show</value>
            <value>verb</value>
            <value>fade-out</value>
            <value>sound</value>
        </choice>
    </attribute>
</define>
```

Event Effect

See `presentation:effect` attribute in section 9.7.2.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:effect" a:defaultValue="none">
            <ref name="presentationEffects"/>
        </attribute>
    </optional>
</define>
```

Effect Direction

See `presentation:direction` attribute in section 9.7.2.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:direction" a:defaultValue="none">
            <ref name="presentationEffectDirections"/>
        </attribute>
    </optional>
</define>
```

Effect Speed

See `presentation:speed` attribute in section 9.7.2.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:speed" a:defaultValue="medium">
            <ref name="presentationSpeeds"/>
        </attribute>
    </optional>
```

```
</define>
```

Start Scale

See `presentation:start-scale` attribute in section 9.7.2.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:start-scale" a:defaultValue="100%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Link

Depending on the action selected by the `presentation:action` attribute, this `xlink:href` attribute either selects a document bookmark or an application.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <choice>
                <value>simple</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:show" a:defaultValue="embed">
            <choice>
                <value>embed</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate" a:defaultValue="onRequest">
            <choice>
                <value>onRequest</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Verb

The [OLE] verb defined by the `presentation:verb` attribute is executed for event listeners of type `verb` at the object that contains this event.

```
<define name="presentation-event-listener-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:verb">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

9.10 Presentation Text Fields

This section describes text fields that are specific to the text of drawing shapes that are contained presentations.

9.10.1 Header Field

Header fields display a header text specified in a header field declaration (see section 9.11.2). Which header field declaration is used is specified by the `presentation:use-header-name` attribute of the draw page where the field occurs. If the field is contained in a presentation shape inside a master page (see section 9.6.1), then the `presentation:use-header-name` attribute of the drawing page for which the drawing shape is displayed is used (see section 9.1.4).

This field is mainly used inside master pages. Since its value may differ for the individual drawing pages that make use of a master page, the current field value is not available.

```
<define name="paragraph-content" combine="choice">
    <element name="presentation:header">
        <empty/>
    </element>
</define>
```

9.10.2 Footer Field

Footer fields display a footer text specified in a footer field declaration (see section 9.11.3). Which footer field declaration is used is specified by the `presentation:use-footer-name` attribute of the draw page where the field occurs. If the field is contained in a presentation drawing shape inside a master page (see section 9.6.1), then the `presentation:use-footer-name` attribute of the drawing page for which the drawing shape is displayed is used (see section 9.1.4).

This field is mainly used inside master pages. Since its value may differ for the individual drawing pages that make use of a master page, the current field value is not available.

```
<define name="paragraph-content" combine="choice">
    <element name="presentation:footer">
        <empty/>
    </element>
</define>
```

9.10.3 Date and Time Field

Date and time fields display a date/time text as specified in the date/time field declaration(see section 9.11.4). Which date-time field declaration is used is specified by the `presentation:use-date-time-name` attribute of the draw page where the field occurs. If the field is contained in a presentation drawing shape inside a master page (see section 9.6.1), then the `presentation:use-date-time-name` attribute of the drawing page for which the drawing shape is displayed is used (see section 9.1.4).

This field is mainly used inside master pages. Since its value may differ for the individual drawing pages that make use of a master page, the current field value is not available.

```
<define name="paragraph-content" combine="choice">
    <element name="presentation:date-time">
        <empty/>
    </element>
</define>
```

9.11 Presentation Document Content

9.11.1 Presentation Declarations

Some presentation specific text fields need per-document declarations before they can be used. For example, header fields require that the header text that is displayed is declared separately. These declarations are collected at the beginning of a text document.

```
<define name="presentation-decls">
    <zeroOrMore>
        <ref name="presentation-decl"/>
    </zeroOrMore>
</define>
```

9.11.2 Header field declaration

The `<presentation:header-decl>` element specifies the text of a header field. See section 9.10.1 for details.

```
<define name="presentation-decl" combine="choice">
    <element name="presentation:header-decl">
        <ref name="presentation-header-decl-attlist"/>
        <text/>
    </element>
</define>
```

Name

The `presentation:name` attribute specifies the name of the header declaration.

```
<define name="presentation-header-decl-attlist" combine="interleave">
    <attribute name="presentation:name">
        <ref name="string"/>
    </attribute>
</define>
```

9.11.3 Footer field declaration

The `<presentation:footer-decl>` element specifies the text of a footer field. See section 9.10.2 for details.

```
<define name="presentation-decl" combine="choice">
    <element name="presentation:footer-decl">
        <ref name="presentation-footer-decl-attlist"/>
        <text/>
    </element>
</define>
```

Name

The `presentation:name` attribute specifies the name of the footer declaration.

```
<define name="presentation-footer-decl-attlist" combine="interleave">
    <attribute name="presentation:name">
        <ref name="string"/>
    </attribute>
</define>
```

9.11.4 Date and Time field declaration

The `<presentation:date-time-decl>` element specifies the text of a date-time field. See section 9.10.3 for details.

```
<define name="presentation-decl" combine="choice">
    <element name="presentation:date-time-decl">
        <ref name="presentation-date-time-decl-attlist"/>
        <text/>
    </element>
</define>
```

Name

The `presentation:name` attribute specifies the name of the date-time declaration.

```
<define name="presentation-date-time-decl-attlist" combine="interleave">
    <attribute name="presentation:name">
        <ref name="string"/>
    </attribute>
</define>
```

Source

The `presentation:source` attribute specifies whether the current date/time or the fixed content of the the field declaration is displayed.

```
<define name="presentation-date-time-decl-attlist" combine="interleave">
    <attribute name="presentation:source">
        <choice>
            <value>fixed</value>
            <value>current-date</value>
        </choice>
    </attribute>
</define>
```

Date and time formatting style

The date style referenced by the `style:data-style-name` attribute is used to format the date and time of the `presentation:date-time` fields if the field is not fixed.

```
<define name="presentation-date-time-decl-attlist" combine="interleave">
    <optional>
        <attribute name="style:data-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

9.11.5 Presentation Settings

The settings for a presentation are stored in the element `<presentation:settings>` inside an `<office:presentation>` element. These settings affect the behavior if the document is displayed in a presentation.

```
<define name="presentation-settings">
    <optional>
        <element name="presentation:settings">
            <ref name="presentation-settings-attlist"/>
            <zeroOrMore>
                <ref name="presentation-show"/>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

The attributes that may be associated with the `<presentation:settings>` element are:

- Start page
- Show
- Full screen
- Endless
- Pause
- Show logo
- Force manual
- Mouse visible
- Mouse as pen
- Start with navigator
- Animation
- Transition on click
- Stay on top

Start page

The attribute `presentation:start-page` specifies the name of the page on which the presentation starts. If this attribute is set, it overrides the `presentation:show` attribute.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:start-page">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Show

The attribute `presentation:show` specifies the name of a show definition (see section 9.11.6) that is used for the presentation. If the `presentation:start-page` attribute is set, it overrides the value of this attribute.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:show">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Full Screen

The attribute `presentation:full-screen` determines whether the presentation is displayed in full screen mode or in a window.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:full-screen" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Endless

The attribute `presentation:endless` switches indefinite repetition of a presentation on and off.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:endless" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Pause

If a presentation is repeated indefinitely, the attribute `presentation:pause` specifies a time duration for displaying a pause screen before the presentation is played again. If this attribute is not set or has a value of 0, a pause screen is not displayed in endless mode. The value of this attribute must conform to the time period format described in §3.2.6 of [xmlschema-2].

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:pause">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

Show Logo

The attribute `presentation:show-logo` specifies whether or not a presentation application shows its logo on the pause screen.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:show-logo" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Force Manual

If set, the attribute `presentation:force-manual` overrides all `presentation:transition-type` properties that are specified within a presentation page (see section 15.36.1) and sets it to `manual`.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:force-manual" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Mouse Visible

The attribute `presentation:mouse-visible` specifies whether or not the mouse pointer is visible during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:mouse-visible" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Mouse As Pen

The attribute `presentation:mouse-as-pen` specifies if the mouse pointer is displayed as a pen or a pointer. If the mouse is displayed as a pen the user can draw sketches on the pages during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:mouse-as-pen" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Start With Navigator

The attribute `presentation:start-with-navigator` specifies whether or not the navigator window is initially displayed during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:start-with-navigator"
                   a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Animations

The attribute `presentation:animations` enables or disables the playback of bitmap animations during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:animations" a:defaultValue="enabled">
            <choice>
                <value>enabled</value>
                <value>disabled</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Transition On Click

The attribute `presentation:transition-on-click` enables or disables a manual transition by a mouse click on the slide during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:transition-on-click"
                   a:defaultValue="enabled">
            <choice>
                <value>enabled</value>
                <value>disabled</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Stay On Top

If the attribute `presentation:stay-on-top` is set to `true`, the presentation window is displayed on top of other windows during a presentation.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:stay-on-top" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Show End-Of-Presentation Slide

The attribute `presentation:show-end-of-presentation-slide` defines whether the presentation application should show an additional slide at the end of the presentation, telling the user that the presentation is finished.

This slides content itself is not defined within the document, but is generated by the application automatically.

```
<define name="presentation-settings-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:show-end-of-presentation-slide"
                   a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

9.11.6 Show Definitions

A presentation document can contain one or more `<presentation:show>` elements. A `<presentation:show>` element customizes the order in which the pages are displayed during a presentation. It can be also used to omit pages from the presentation or to repeat pages during the presentation.

This element is optional.

```
<define name="presentation-show">
    <element name="presentation:show">
        <ref name="presentation-show-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<presentation:show>` element are:

- Name
- Pages

Name

The attribute `presentation:name` uniquely identifies a `<presentation:show>` element.

```
<define name="presentation-show-attlist" combine="interleave">
    <attribute name="presentation:name">
        <ref name="string"/>
    </attribute>
</define>
```

Pages

The attribute `presentation:pages` contains a comma separated list of page names. The pages are displayed in the order in which they are listed during a presentation that uses this show. Pages can be included more than once.

```
<define name="presentation-show-attlist" combine="interleave">
    <attribute name="presentation:pages"/>
</define>
```

10 Chart Content

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

- Introduction to Chart Documents
- Chart
- Title, Subtitle and Footer
- Legend
- Plot Area
- Wall
- Floor
- Axis
- Series
- Categories
- Data Point
- Mean Value
- Error Indicator
- Regression Curves

10.1 Introduction to Chart Documents

Chart documents are always contained within other XML documents. There are two types of chart container documents:

- Documents that do not provide data for the chart: The chart data is contained in a `<table:table>` element inside the `<chart:chart>` element.
- Documents that provide data for the chart: The chart data may be contained in a `<table:table>` element in the parent document, for example, in a spreadsheet or text document.

The chart data is specified by the `<chart:plot-area>` element's `table:cell-range-address` attribute. The `<chart:plot-area>` element represents the visualization container of all data series in the chart.

10.2 Chart

The `<chart:chart>` element represents an entire chart, including titles, a legend, and the graphical object that visualizes the underlying data called the plot area. The data underlying the chart is represented by a table element. This element may also exist for embedded charts that get the data from the container document. In this case the chart can be rendered without getting the data from the container document.

```
<define name="chart-chart">
```

```
    <element name="chart:chart">
        <ref name="chart-chart-attlist"/>
        <optional>
            <ref name="chart-title"/>
        </optional>
        <optional>
            <ref name="chart-subtitle"/>
        </optional>
        <optional>
            <ref name="chart-footer"/>
        </optional>
        <optional>
            <ref name="chart-legend"/>
        </optional>
        <ref name="chart-plot-area"/>
        <optional>
            <ref name="table-table"/>
        </optional>
    </element>
</define>
```

Class

The `chart:class` attribute specifies the chart type. The chart type is represented by a namespaced token, meaning an identifier prefixed by an XML namespace prefix, just like any attribute or element name in this specification. This specification defines a number of chart types in the chart namespace (URN: urn:oasis:names:tc:opendocument:xmlns:chart:1.0). Additional chart types may be supported by using a different namespace.

The chart type may be specified more precisely with formatting properties that may be attached to chart styles. For example, a 3D bar chart with horizontal bars is specified by setting the class attribute to `chart:bar` and by adding the properties for three dimensional and horizontal arrangement in the corresponding style.

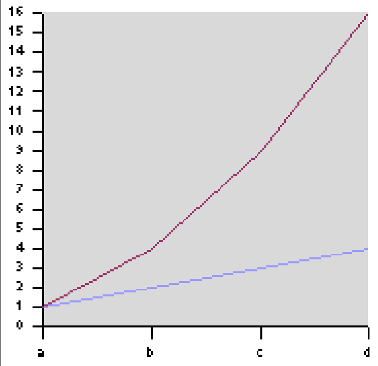
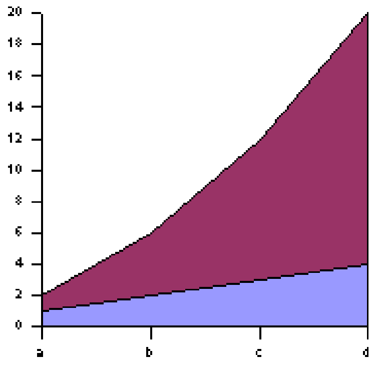
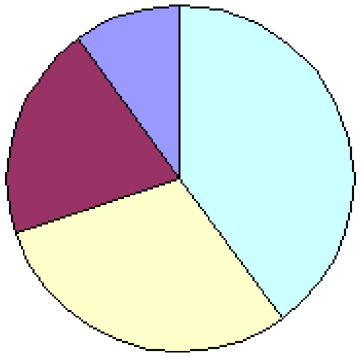
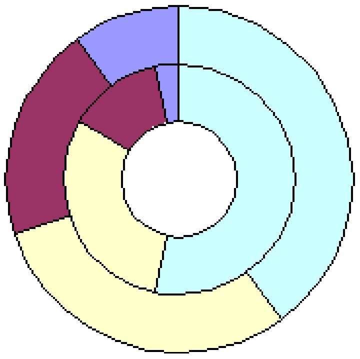
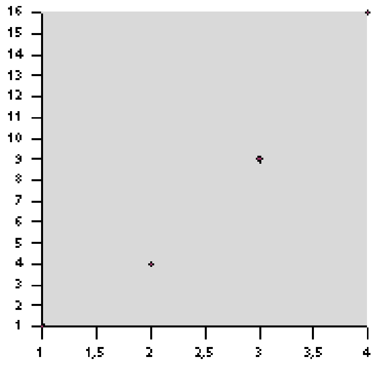
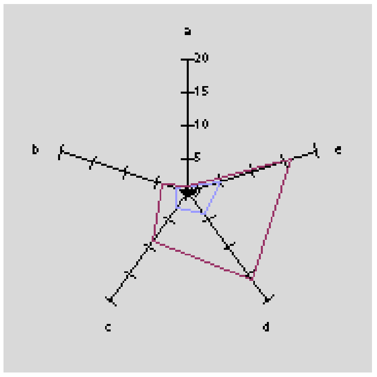
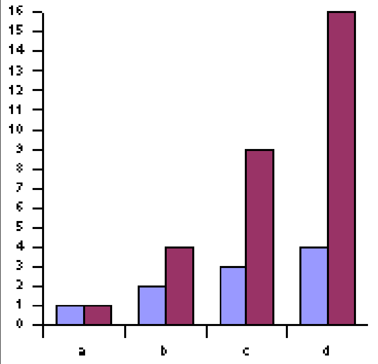
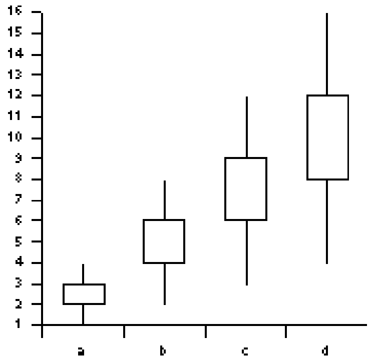
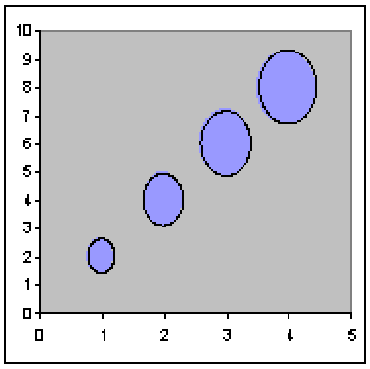
```
<define name="chart-chart-attlist" combine="interleave">
    <attribute name="chart:class">
        <ref name="namespacedToken"/>
    </attribute>
</define>
```

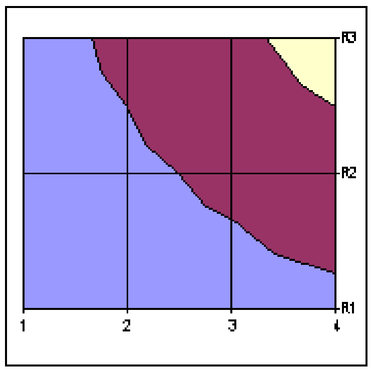
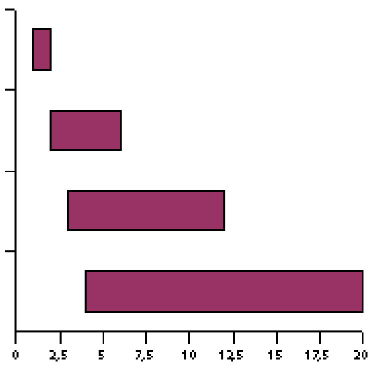
The pre-defined chart types are:

- **line** – the data points of each data series are connected through lines.
- **area** – the area below a data series is filled, and additional data series are stacked.
- **circle** – a circular chart is segmented according to the relative weights of the data points.
- **ring** – each data series is represented as a concentric rings, with each ring rendered as if it was part of a circle chart of the series.
- **scatter** – a pair of data series is used to determine x and y positions for each data point.
- **radar** – a radial plot of the data points, where the value of each point determines the distance from the chart origin. The data points of a series are connected, thus forming a closed line around the center.
- **bar** – each data point is depicted by a bar whose length is proportional to the data value.
- **stock** – four data series are interpreted as opening, minimum, maximum and closing stock values.

- **bubble** – the first two of three data series are interpreted a positions as in a scatter chart, where the area of each data point is sized relative to the value in the third data series.
- **surface** – the data points are interpreted as tabular data, where each value defines a 'height' at a specific grid location. The graph may visualize these using colors for height intervals, creating color bands similar to geographical maps.
- **gantt** – a pair of data series is used to determine the start and end positions for horizontal bars

Example: The following table shows examples for the pre-defined chart types. Those charts that use one or two data series use two data series with the values 1;2;3;4 and 1;4;9;16 and the labels a;b;c;d. Those chart types that use more than two data series (stock and bubble) use the data series 1;2;3;4 and multiples thereof. The radar chart uses two data series with five data points.

 **chart:line**	 **chart:area**	 **chart:circle**
 **chart:ring**	 **chart:scatter**	 **chart:radar**
 **chart:bar**	 **chart:stock**	 **chart:bubble**

 chart:surface	 chart:gantt	

Size

The svg:width and svg:height (see section 9.2.15) attributes define the extent of the entire chart. If they are omitted, the size of the chart is determined by the size of the window in which the chart is displayed.

```
<define name="chart-chart-attlist" combine="interleave">
    <ref name="common-draw-size-attlist"/>
</define>
```

Column and Row Mapping

The chart:column-mapping and chart:row-mapping attributes contain, if provided, a list of indexes of series. The numbers define a reordering of data that comes from a container document that provides the data for the chart. The numbering begins with 1. A list of ascending numbers beginning with 1 has no effect. To exchange two series, their numbers must be exchanged in the list. For example, 1 3 2 4 exchanges the second and the third series.

The chart:column-mapping and chart:row-mapping attributes must not be used simultaneously.

```
<define name="chart-chart-attlist" combine="interleave">
    <optional>
        <attribute name="chart:column-mapping">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
<define name="chart-chart-attlist" combine="interleave">
    <optional>
        <attribute name="chart:row-mapping">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Style Name

The chart:style-name attribute references a chart style. See section 14.16 for details.

Within the style applied to the <chart:chart> element, fill properties (described in section 15.14) and the stroke properties (described in section 15.13) as well as the scale text property described in section 15.29.1 can be used.

```
<define name="chart-chart-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.3 Title, Subtitle and Footer

10.3.1 Title

The `<chart:title>` element represents a main title object in a chart document. 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 10.8. In this case the title is displayed beside the axis object.

```
<define name="chart-title">
    <element name="chart:title">
        <ref name="chart-title-attlist"/>
        <optional>
            <ref name="text-p"/>
        </optional>
    </element>
</define>
```

Table Range

A chart title may be bound to a table cell, causing the current content of the given cell to be displayed in the chart title.

```
<define name="chart-title-attlist" combine="interleave">
    <optional>
        <attribute name="table:cell-range">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

Position and Size

The common positioning attributes for drawing objects can be used on `<chart:title>` elements.

```
<define name="chart-title-attlist" combine="interleave">
    <ref name="common-draw-position-attlist"/>
</define>
```

Style Name

The `chart:style-name` attribute specifies a chart style for the `<chart:title>` element. Within the referenced style, fill and stroke properties may be used. They are applied to the surrounding title box. See sections 15.14 and 15.13 for more information. In addition to this, text properties may be used. They are applied to the title text itself. See section 15.4.

```
<define name="chart-title-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.3.2 Subtitle

The `<chart:subtitle>` element represents a subtitle which can be used for additional title information in a chart.

The structure of the `<chart:subtitle>` element is the same as that of the `<chart:title>` element. The attributes that may be associated with the `<chart:subtitle>` element are the same as those that may be associated with the `<chart:title>` element. See section 10.3.1 for more information.

```
<define name="chart-subtitle">
    <element name="chart:subtitle">
        <ref name="chart-title-attlist"/>
        <optional>
            <ref name="text-p"/>
        </optional>
    </element>
</define>
```

10.3.3 Footer

The `<chart:footer>` element represents a footer below the chart's plot area.

The structure of the subtitle element is the same as that of the `<chart:title>` title element. See section 10.3.1 for more information.

```
<define name="chart-footer">
    <element name="chart:footer">
        <ref name="chart-title-attlist"/>
        <optional>
            <ref name="text-p"/>
        </optional>
    </element>
</define>
```

10.4 Legend

The `<chart:legend>` element determines whether or not a legend is displayed in the chart. The legend's position may be specified either as a relative or as an absolute position. The size of the legend is calculated automatically and therefore cannot be set as attribute.

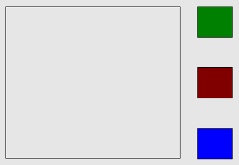
```
<define name="chart-legend">
    <element name="chart:legend">
        <ref name="chart-legend-attlist"/>
        <empty/>
    </element>
</define>
```

Legend Placement

The legend can be placed automatically, next to the plot area, or in one of the corners. This placement is determined by the `chart:legend-position` attribute, which may have the values `start`, `end`, `top`, `bottom` for legend positions next to the plot area and `top-start`, `bottom-start`, `top-end` or `bottom-end` for legend positions in the corners. If the legend is placed next to the plot area, in any of the four directions `start`, `end`, `top` `bottom`, an additional alignment attribute `chart:legend-align` determines which border (`start`, `end`) or axis (`center`) of the legend and the plot area are to be aligned.

```
<define name="chart-legend-attlist" combine="interleave">
    <choice>
        <group>
            <attribute name="chart:legend-position">
                <choice>
                    <value>start</value>
                    <value>end</value>
                    <value>top</value>
                    <value>bottom</value>
                </choice>
            </attribute>
            <optional>
                <attribute name="chart:legend-align">
                    <choice>
                        <value>start</value>
                        <value>center</value>
                        <value>end</value>
                    </choice>
                </attribute>
            </optional>
        </group>
        <attribute name="chart:legend-position">
            <choice>
                <value>top-start</value>
                <value>bottom-start</value>
                <value>top-end</value>
                <value>bottom-end</value>
            </choice>
        </attribute>
        <empty/>
    </choice>
</define>
```

Example: If `chart:legend-position="right"`, the legend will be positioned to the right of the chart's plot area. The `chart:legend-align` values of `start`, `center`, and `end` will yield legend positions as depicted by the green, red, and blue boxes, respectively.



The legend position can also be given in absolute coordinates, as with any drawing object. If both a drawing position and legend placement options are available, the legend placement takes precedence and the position should reflect the automatic placement.

```
<define name="chart-legend-attlist" combine="interleave">
    <ref name="common-draw-position-attlist"/>
</define>
```

Legend Expansion

The legend needs to be expanded to accommodate additional legend items. The `style:legend-expansion` attribute determines in which direction the legend expands. Legend expansion of `wide` and `high` causes the legend to be expanded horizontally and vertically. An expansion `balanced` causes expansion into both directions. An expansion value of `custom` with a numeric `style:legend-expansion-aspect-ratio` causes the legend to be expanded such that the given ratio between width and height is observed.

```
<define name="chart-legend-attlist" combine="interleave">
    <choice>
        <attribute name="style:legend-expansion">
            <choice>
                <value>wide</value>
                <value>high</value>
                <value>balanced</value>
            </choice>
        </attribute>
        <group>
            <attribute name="style:legend-expansion">
                <value>custom</value>
            </attribute>
            <attribute name="style:legend-expansion-aspect-ratio">
                <ref name="double"/>
            </attribute>
        </group>
        <empty/>
    </choice>
</define>
```

Legend Styling

Additional styling information for the chart legend can be referenced through the `chart:style-name` attribute. The style may specify fill and stroke properties. They are applied to the legend object. See sections 15.14 and 15.13 for more information. In addition to this, the style may specify text properties. They are applied to the text inside the legend object. See section 15.4.

```
<define name="chart-legend-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.5 Plot Area

The `<chart: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.

```
<define name="chart-plot-area">
    <element name="chart:plot-area">
        <ref name="chart-plot-area-attlist"/>
        <zeroOrMore>
            <ref name="dr3d-light"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="chart-axis"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="chart-series"/>
        </zeroOrMore>
        <optional>
            <ref name="chart-stock-gain-marker"/>
        </optional>
        <optional>
            <ref name="chart-stock-loss-marker"/>
        </optional>
        <optional>
            <ref name="chart-stock-range-line"/>
        </optional>
        <optional>
            <ref name="chart-wall"/>
        </optional>
        <optional>
            <ref name="chart-floor"/>
        </optional>
    </element>
</define>
```

Plot Area Positioning

The plot area's position and size are determined the common positioning and sizing attributes for drawing objects. If the position and size attributes are not specified, the values are calculated by the render application.

```
<define name="chart-plot-area-attlist" combine="interleave">
    <ref name="common-draw-position-attlist"/>
    <ref name="common-draw-size-attlist"/>
</define>
```

Plot Area Style

The `chart:style-name` attribute that is set for the `<chart:plot-area>` element is used for all data elements contained inside the plot area, unless extra styles are specified in one of those sub-elements. These data elements can be `<chart:series>` and `<chart:data-point>` elements.

If the chart is three-dimensional, 3D scene properties may be applied to the plot area. See the section 15.22 - 15.26 for more information.

```
<define name="chart-plot-area-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Plot Area Data Attributes

If a chart is embedded in a document that provides the data for the chart, the `table:cell-range-address` attribute reflects the ranges from which all the data for the chart comes. The range given here is interpreted by the chart as consecutive series.

```
<define name="chart-plot-area-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="table:cell-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

If the first row or column, or both contains labels, this is stated by the `chart:data-source-has-labels` attribute.

```
<define name="chart-plot-area-attlist" combine="interleave">
    <optional>
        <attribute name="chart:data-source-has-labels" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>row</value>
                <value>column</value>
                <value>both</value>
            </choice>
        </attribute>
    </optional>
</define>
```

The `chart:series-source` formatting property specified in section 15.34.1 determines whether the data table contains the data series in column-wise or row-wise fashion.

10.5.1 3D Plot Area

The plot area may be displayed as an 3D scene as specified in section 9.4.1. All 3D attributes that can be applied to the `<dr3d:scene>` element can be applied to the `<chart:plot-area>` element, including the `dr3d:transform` attribute. It represents the rotation of a chart scene, that is the three-dimensional plot area. See section 9.4.1 for more information. In addition to this, the `<chart:plot-area>` element may contain a `<dr3d:light>` element as specified in section 9.4.2.

```
<define name="chart-plot-area-attlist" combine="interleave">
    <ref name="dr3d-scene-attlist"/>
    <ref name="common-dr3d-transform-attlist"/>
</define>
```

10.6 Wall

The `<chart:wall>` element can be contained in the `<chart:plot-area>` element. It specifies a chart's wall. For two-dimensional charts, the wall spans the entire plot area. For three-dimensional charts, the wall usually consists of two perpendicular rectangles.

```
<define name="chart-wall">
    <element name="chart:wall">
        <ref name="chart-wall-attlist"/>
        <empty/>
    </element>
</define>
```

Width

The `svg:width` attributes specifies the width of the wall for three-dimensional charts.

```
<define name="chart-wall-attlist" combine="interleave">
    <optional>
        <attribute name="svg:width">
```

```
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Style

The `<chart:wall>` element may have a `chart:style-name` attribute to specify further styling information. They style may contain fill and stroke properties. See sections 15.14 and 15.13 for more information.

```
<define name="chart-wall-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.7 Floor

The `<chart:floor>` element can be contained in the `<chart:plot-area>` element. For three-dimensional charts, the `<chart:floor>` element is present in addition to the `<chart:wall>` element.

```
<define name="chart-floor">
    <element name="chart:floor">
        <ref name="chart-floor-attlist"/>
        <empty/>
    </element>
</define>
```

Size

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. The `svg:width` attribute can be used to set the width of the floor.

```
<define name="chart-floor-attlist" combine="interleave">
    <optional>
        <attribute name="svg:width">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Style

The `<chart:floor>` element may have a `chart:style-name` attribute to specify further styling information. Fill and stroke properties can be applied to a floor. See sections 15.14 and 15.13 for more information.

```
<define name="chart-floor-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
```

```
</define>
```

10.8 Axis

The `<chart: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.

```
<define name="chart-axis">
    <element name="chart:axis">
        <ref name="chart-axis-attlist"/>
        <optional>
            <ref name="chart-title"/>
        </optional>
        <optional>
            <ref name="chart-categories"/>
        </optional>
        <zeroOrMore>
            <ref name="chart-grid"/>
        </zeroOrMore>
    </element>
</define>
```

Dimension

The `chart:dimension` attribute specifies along which physical axis on the chart the values of the current axis are displayed.

A chart may contain more than one axis with the same dimension. For example, it may have two axes with dimension y. Data series may be attached to either axis. This way, data may be grouped for different scaling. To attach a specific axis to a data series, the axis has to be referenced by the `<chart:series>` element's `chart:axis-name` attribute. If an axis is not references by a data series, it becomes a copy of an existing axis with the same dimension.

The position of an axis in a chart is determined by the rendering application and depends on the chart type. In a chart with horizontal bars, the rendering application usually paints the axis with dimension x on the bottom of the plot area. If there are two axes with dimension y, a rendering application might paint the second axis at the top of the plot area.

```
<define name="chart-axis-attlist" combine="interleave">
    <attribute name="chart:dimension">
        <choice>
            <value>x</value>
            <value>y</value>
            <value>z</value>
        </choice>
    </attribute>
</define>
```

Name

The `chart:name` attribute can be used to assign a name to this axis, so it can be referenced from e.g., a data series.

```
<define name="chart-axis-attlist" combine="interleave">
    <optional>
        <attribute name="chart:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Style

A `chart:style-name` attribute can be associated with an axis. Stroke properties can be applied to axes; see section 15.13. These properties affect all lines of the axis object. Text properties can also be applied to axes; see section 15.4. These properties affect the appearance of all text objects. The axis properties described in section 15.31 can also be used.

The chart style that is referenced by the `chart:style-name` attribute may specify a data style that is used to format the axis' labels. See section 14.1 for details.

```
<define name="chart-axis-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Example: Bar chart

In this example, there are two axes with dimension y. One of these axes has the name `primary-value`. A data series has been attached to that named axis. There is no data attached to the second axis, therefore an axis name has not been specified, 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:dimension="x"
                    chart:axis-name="x"/>
        <chart:axis chart:dimension="y"
                    chart:axis-name="primary-value"/>
        <chart:axis chart:dimension="y"/>
        <chart:series chart:values-address="Sheet1.A1:.A7"
                    chart:attached-axis="primary-value"/>
    </chart:plot-area>
</chart:chart>
```

10.8.1 Grid

The `<chart:grid>` element can be contained in a `<chart:axis>` element. It adds a grids to the axis.

```
<define name="chart-grid">
    <element name="chart:grid">
        <ref name="chart-grid-attlist"/>
    </element>
</define>
```

Class

The `chart:class` attribute specifies whether major or minor tick marks are used. If a major grid is applied to an axis, the major tick marks are extended to grid lines. If a grid is minor, any minor tick marks assigned to the axis are used.

```
<define name="chart-grid-attlist" combine="interleave">
    <optional>
        <attribute name="chart:class" a:defaultValue="major">
            <choice>
                <value>major</value>
                <value>minor</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Style Name

The `<chart:grid>` element may have a `chart:style-name` attribute to specify further styling information. Stroke properties can be applied to grids, which affect the lines of the grid. See section 15.13 for information on these stroke properties.

```
<define name="chart-grid-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.9 Series

The `<chart:series>` element represents a data series in a chart. If the chart requires more input data like scatter and bubble charts, `<chart:domain>` sub-elements must be defined that mainly contain the `cell-range-address` of the corresponding data.

```
<define name="chart-series">
    <element name="chart:series">
        <ref name="chart-series-attlist"/>
        <zeroOrMore>
            <ref name="chart-domain"/>
        </zeroOrMore>
        <optional>
            <ref name="chart-mean-value"/>
        </optional>
        <optional>
            <ref name="chart-regression-curve"/>
        </optional>
        <optional>
            <ref name="chart-error-indicator"/>
        </optional>
        <zeroOrMore>
            <ref name="chart-data-point"/>
        </zeroOrMore>
    </element>
</define>
```

Cell Range

The `chart:values-cell-range-address` attribute allows a range to be specified that contains the values that should be visualized by this data series.

```
<define name="chart-series-attlist" combine="interleave">
    <optional>
        <attribute name="chart:values-cell-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
</define>
```

The `chart:label-cell-address` attribute allows a name to be provided for the series.

```
<define name="chart-series-attlist" combine="interleave">
    <optional>
        <attribute name="chart:label-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

Class

The `chart:class` attribute can be used to assign a chart type to be used for rendering the data of this `<chart:series>` element. A `chart:class` attribute for a `<chart:series>` element overrides the `chart:class` attribute for the entire chart. This allows the creation of charts with multiple sub-charts, e.g., a bar chart with one or more data series rendered as lines. For more information on the available chart classes, see section 10.2.

```
<define name="chart-series-attlist" combine="interleave">
    <optional>
        <attribute name="chart:class">
            <ref name="namespacedToken"/>
        </attribute>
    </optional>
</define>
```

Attached Axis

The `chart:attached-axis` attribute can be used to assign the data series to a `<chart:axis>` element.

```
<define name="chart-series-attlist" combine="interleave">
    <optional>
        <attribute name="chart:attached-axis">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Style Name

Styling attributes for the data series can be assigned through the `chart:style-name` attribute. Fill and stroke properties may be applied for `<chart:series>` element, see sections 15.14 and 15.13 for information. Text properties can also be applied to the descriptive text underneath the series, see section 15.4 for information.

```
<define name="chart-series-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.9.1 Domain

For scatter and bubble charts, one ore more `<chart:domain>` elements must be specified for the `<chart:series>` elements.

For scatter charts, one `<chart:domain>` element is required. Its `cell-range-address` attribute references the x coordinate values for the scatter chart.

For bubble charts, two `<chart:domain>` elements are required. Their `cell-range-address` attributes reference the x and y coordinate values for the bubble chart

For both chart types, there must be at least one `<chart:series>` element with the necessary number of `<chart:domain>` sub-elements. All other `<chart:series>` elements can omit these. In this case, the first domain that is specified is used.

```
<define name="chart-domain">
    <element name="chart:domain">
    <optional>
        <attribute name="table:cell-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
    </element>
</define>
```

10.10 Categories

The element `<chart:categories>` element represents the range of cell addresses that contains the captions for the categories contained in each series.

The element may contain a `table:cell-range-address` that denotes the region from which the category labels are taken from. If this attribute or the `<chart:categories>` element is omitted the application will evaluate the `chart:data-source-has-labels` attribute.

```
<define name="chart-categories">
    <element name="chart:categories">
    <optional>
        <attribute name="table:cell-range-address">
            <ref name="cellRangeAddress"/>
        </attribute>
    </optional>
    </element>
</define>
```

10.11 Data Point

If a single data point in a data series should have a specific appearance, the `<chart:data-point>` element is used to apply the required properties.

```
<define name="chart-data-point">
    <element name="chart:data-point">
```

```
        <ref name="chart-data-point-attlist"/>
        <empty/>
    </element>
</define>
```

Repetition

The `chart:repeated` attribute serves as a simplification if more than one consecutive data-points have the same properties. For example, the following XML-fragments have an identical meaning:

```
<chart:series chart:style-name="ch9">
   <chart:data-point/>
   <chart:data-point/>
   <chart:data-point/>
   <chart:data-point/>
</chart:series>
```

and

```
<chart:series chart:style-name="ch9">
   <chart:data-point chart:repeated="4"/>
</chart:series>
```

```
<define name="chart-data-point-attlist" combine="interleave">
    <optional>
        <attribute name="chart:repeated">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Style

The `chart:style-name` attribute referenced a chart style. Fill and stroke properties can be applied to each data point object, see sections 15.14 and 15.13. Text properties can also be applied to the descriptive text located underneath the data points, see section 15.4.

```
<define name="chart-data-point-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>

```

10.12 Mean Value

The formatting properties of the mean-value line are stored in the `<chart:mean-value>` element, which may be part of a `<chart:series>` element.

```
<define name="chart-mean-value">
    <element name="chart:mean-value">
        <ref name="chart-mean-value-attlist"/>
        <empty/>
    </element>
</define>
```

Style Name

The `chart:style-name` attribute references a chart style that contains the formatting properties for the mean-value line.

```
<define name="chart-mean-value-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.13 Error Indicator

The formatting properties of error-indicators are stored in the `<chart:error-indicator>` elements which may be part of a series.

```
<define name="chart-error-indicator">
    <element name="chart:error-indicator">
        <ref name="chart-error-indicator-attlist"/>
        <empty/>
    </element>
</define>
```

Style Name

The `chart:style-name` attribute references a chart style that contains the formatting properties for the error indicator.

```
<define name="chart-error-indicator-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.14 Regression Curves

The formatting properties of regression-lines are stored in the `<chart:regression-curve>` elements which may be part of a series.

```
<define name="chart-regression-curve">
    <element name="chart:regression-curve">
        <ref name="chart-regression-curve-attlist"/>
        <empty/>
    </element>
</define>
```

Style Name

The `chart:style-name` attribute referenced a chart style that contains the formatting properties for the error indicator. The chart style especially may contain the regression type property specified in section 15.35.1.

```
<define name="chart-regression-curve-attlist" combine="interleave">
    <optional>
        <attribute name="chart:style-name">
```

```
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

10.14.1 Stock Chart Markers

The properties of a stock chart, i.e., the different colors for filling the candlestick-bars or the line-styles of the lines pointing to the high and low values (the range-line), are stored in separate elements.

The candlestick-bars for stocks that have a higher close-value than open-value take their formatting from the `<chart:stock-gain-marker>` element's properties, whereas stocks which close value is lower than the open-value, use the properties stored in `<chart:stock-loss-marker>`.

```
<define name="chart-stock-gain-marker">
    <element name="chart:stock-gain-marker">
        <ref name="common-stock-marker-attlist"/>
    </element>
</define>
<define name="chart-stock-loss-marker">
    <element name="chart:stock-loss-marker">
        <ref name="common-stock-marker-attlist"/>
    </element>
</define>
<define name="chart-stock-range-line">
    <element name="chart:stock-range-line">
        <ref name="common-stock-marker-attlist"/>
    </element>
</define>
```

Style Name

The `chart:style-name` attribute referenced a chart style that contains the formatting properties for stock markers.

```
<define name="common-stock-marker-attlist">
    <optional>
        <attribute name="chart:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

11 Form Content

A form is a container for user interface controls which a user interacts with. For example, buttons, text boxes, check boxes, and drop-down lists are user interface controls that can be contained in a form. In the XML file format, the following basic rules apply to user interface controls and forms:

- All controls must be located in a form.
- All controls that are not hidden have to be associated with an absolute or relative position. These visual aspects of the control are represented by drawing shapes that contain a reference to the control. See section 9.2.12 for details.
- Forms may be nested.
- Forms are not connected with the text flow and layout of a document. This does not apply to controls.
- Forms can be data-aware. The controls reflect the content of a database.

Forms define rules for the following form behavior:

- Submitting the form, which is similar to [HTML4].

  Note: Form submission is only supported for non nested forms that contain only controls that can be converted to HTML.
- Connecting to a data source. When this happens, the controls in a form become data-aware.
- Submitting and binding according to the [XForms] data model.

Forms are contained in the `<office:forms>` section of an XML document. This element may contain an arbitrary sequence of `<form:form>` or `<xforms:model>` elements. Note that controls are always declared inside a `<form:form>` element, while an `<xforms:model>` element contains only the XForms data model. Thus, the `<office:forms>` element may contain only `<form:form>` elements but no `<xforms:model>` element, while an `<xforms:model>` would typically be accompanied by an additional `<form:form>` element.

```
<define name="office-forms">
    <optional>
        <element name="office:forms">
            <ref name="office-forms-attlist"/>
            <zeroOrMore>
                <choice>
                    <ref name="form-form"/>
                    <ref name="xforms-model"/>
                </choice>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

For ease of use when using (filling out) forms, applications may focus controls initially so that the user can immediately type into the first form control. To achieve this behavior, the `form:automatic-focus` flag may be set to `true`.

```
<define name="office-forms-attlist" combine="interleave">
    <optional>
        <attribute name="form:automatic-focus" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Application which support both creation and usage (filling out) of forms, the `form:apply-design-mode` flag determines whether the application is supposed to present the forms in this document in editable or fill-out state.

```
<define name="office-forms-attlist" combine="interleave">
    <optional>
        <attribute name="form:apply-design-mode" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1 Form

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

This element is contained in either an `<office:forms>` or a `<form:form>` element. It contains the controls and sub forms of the form, a `<form:properties>` element which defines the properties of the form, and an `<office:events-listeners>` element that contains the events for the form.

```
<define name="form-form">
    <element name="form:form">
        <ref name="common-form-control-attlist"/>
        <ref name="form-form-attlist"/>
        <optional>
            <ref name="form-properties"/>
        </optional>
        <optional>
            <ref name="office-event-listeners"/>
        </optional>
        <zeroOrMore>
            <choice>
                <ref name="controls"/>
                <ref name="form-form"/>
            </choice>
        </zeroOrMore>
        <optional>
            <ref name="form-connection-resource"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<form:form>` are as follows:

- Name. See section 11.4.
- Service name. See section 11.4.
- Action
- Target frame
- Method
- Encoding Type

- Allow deletes
- Allow inserts
- Allow updates
- Apply filter
- Command type
- Command
- Data source
- Master fields
- Detail fields
- Escape processing
- Filter
- Ignore result
- Navigation mode
- Order
- Tabbing cycle

11.1.1 Action

The `xlink:href` attribute represents the IRI of the processing agent for the form.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <value>simple</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onRequest">
                <value>onRequest</value>
            </attribute>
        </optional>
    </optional>
</define>
```

11.1.2 Target Frame

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

This attribute can have one of the following values:

- `_self`: The form replaces the content of the current frame.
- `_blank`: The form is displayed in a new frame.

- `_parent`: The form is displayed in the parent frame of the current frame.
- `_top`: The form 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 form is displayed in the named frame. If the named frame does not exist, a new frame with that name is created.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="office:target-frame" a:defaultValue="_blank">
            <ref name="targetFrameName"/>
        </attribute>
    </optional>
</define>
```

11.1.3 Method

The `form:method` attribute specifies the HTTP method to use to submit the data in the form to the server. The value of this attribute can be `get` or `post`. The default value is `get`. These values are not case sensitive.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:method" a:defaultValue="get">
            <choice>
                <value>get</value>
                <value>post</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

11.1.4 Encoding Type

If the value of the `form:method` attribute is `post`, the `form:enctype` attribute specifies the content type used to submit the form to the server. The default value of this attribute is `application/x-www-form-urlencoded`. Other suitable MIME types are also acceptable.

See §17.3 of [HTML4] for more information.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:enctype"
                    a:defaultValue="application/x-www-form-urlencoded">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.1.5 Allow Deletes

The `form:allow-deletes` attribute specifies whether or not data records can be deleted. It applies only if the form is data-aware.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:allow-deletes" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.6 Allow Inserts

The `form:allow-inserts` attribute specifies whether or not new data records can be inserted. It applies only if the form is data-aware.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:allow-inserts" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.7 Allow Updates

The `form:allow-updates` attribute specifies whether or not data records can be updated.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:allow-updates" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.8 Apply Filter

The `form:apply-filter` attribute specifies whether or not filters should be applied to the form. See also the Filter attribute.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:apply-filter" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.9 Command Type

The `form:command-type` attribute specifies the type of command to execute on the data source. The value of this attribute can be one of the following:

- `table`: The command contains a table name. The form retrieves all of the data in the table.
- `query`: The command contains the name of query. The form retrieves and executes the query.
- `command` : The command contains an SQL statement. The form executes the SQL statement.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:command-type" a:defaultValue="command">
            <choice>
```

```
                <value>table</value>
                <value>query</value>
                <value>command</value>
            </choice>
        </attribute>
    </optional>
</define>
```

11.1.10 Command

The `form:command` attribute specifies the command to execute on the data source.

The value is interpreted differently, depending to the value of the Command Type attribute of the form. It can be the name of a database table, the name of a query object or an SQL statement.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:command"/>
    </optional>
</define>
```

11.1.11 Data Source

The `form:datasource` attribute specifies the name of a data source to use for the form.

The value of this attribute can be one of the following:

- A URL specifying a database connection.
- A data source name that the office application can use to establish database connections.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:datasource">
            <choice>
                <ref name="anyURI"/>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

11.1.12 Master Fields

The `form:master-fields` attribute is used for nested data-aware forms. It specifies the names of the columns in the result set represented by the parent form. Usually, they denote the foreign key fields of the parent form. The values of the columns are used to parameterize the data for the nested form. Each time the parent form changes the current row, the nested form queries the database again based on the values of the master fields.

The attribute contains a comma separated list of field names.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:master-fields">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.1.13 Detail Fields

The `form:detail-fields` attribute is used for nested database forms. It specifies the names of the columns in detail forms that are related to columns in the parent form. The columns are used as parameters in the command for the nested form to retrieve the details for a matching master form record.

This attribute contains a comma separated list of field names.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:detail-fields">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.1.14 Escape Processing

If the value of the `form:command-type` attribute is `command`, the `form:escape-processing` attribute specifies whether or not the application processes the command before passing it to the database driver.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:escape-processing" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.15 Filter

The `form:filter` attribute specifies a filter for the command to base the form on. No matter whether the form is based on a `query`, a `table`, or a `command`, the filter is always conjunctively added to any possible existing filter. The filter usually forms a SQL “WHERE” clause, without the “WHERE” keyword.

The `form:apply-filter` attribute specifies whether or not the filter is actually applies to the command.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:filter">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.1.16 Ignore Result

The `form:ignore-result` attribute specifies whether or not to discard all results that are retrieved from the underlying data source. If `true`, a database-bound form will discard any data it queries from the database, and thus only inserting and editing of new records is available. Essentially, this allows a mode of operation where only new data can be inserted into a database.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:ignore-result" a:defaultValue="false">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.1.17 Navigation Mode

The `form:navigation-mode` attribute specifies how the records in a database form are navigated.

The value of this attribute can be one of the following:

- `none`: A dedicated navigation bar is not provided by the user interface. The form must be navigated using the TAB and SHIFT/TAB keys on the keyboard.
- `current`: A navigation bar is provided and the navigation is performed on the current form.
- `parent`: A navigation bar is provided and the navigation is performed on the parent form of the current form.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:navigation-mode">
            <ref name="navigation"/>
        </attribute>
    </optional>
</define>

<define name="navigation">
    <choice>
        <value>none</value>
        <value>current</value>
        <value>parent</value>
    </choice>
</define>
```

11.1.18 Order

The `form:order` attribute specifies a sort criteria for the command. No matter whether the form is based on a `query`, a `table`, or a `command`, the sorting is always conjunctively added to any possible existing sorting. The attribute value usually forms an SQL “ORDER BY” clause, without the “ORDER BY” keyword.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:order">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.1.19 Tabbing Cycle

The `form:tab-cycle` attribute specifies how the application responds when the user presses the TAB key in the controls in a form. The behavior of the application depends on whether or not the form is bound to a data source.

The value of this attribute can be one of the following:

- `records`: If a user presses the TAB key in the last control of the form, the focus moves to the first control specified in the tab order of the same form, and moves the form to the next record.
- `current`: If a user presses the TAB key in the last control of the form, the focus moves to the first control specified in the tab order of the same form, while the record pointer of the form is not touched.
- `page`: If a user presses the TAB key in the last control of a form, the focus moves to the first control specified in the tab order for the next form.

```
<define name="form-form-attlist" combine="interleave">
    <optional>
        <attribute name="form:tab-cycle">
            <ref name="tab-cycles"/>
        </attribute>
    </optional>
</define>
<define name="tab-cycles">
    <choice>
        <value>records</value>
        <value>current</value>
        <value>page</value>
    </choice>
</define>
```

11.1.20 Connection Resource

The `<form:connection-resource>` element specifies the source database by an [XLink]. Its `xlink:href` attribute either references a file containing a database, or it contains information on how to make a connection to a database, for instance a [JDBC] URL.

```
<define name="form-connection-resource">
    <element name="form:connection-resource">
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <empty/>
    </element>
</define>
```

11.2 XForms Model

The form model described in section 11.1 implies a data model where each control defines a name-value-pair, with the name being determined by the control id and the value being editable through the control. No interaction between controls is possible (save for macro programming). For applications where this kind of form logic does not suffice, W3C has introduced XForms (see [XForms]), a standard for XML-based forms.

XForms is designed to be embedded in another XML format. It consists of two major parts, the XForms model which contains the form logic plus form data, and the XForms controls, which can be bound to a data model. In the OASIS Open Office 1.0 we embed the W3C XForms model as defined by the `<xforms:model>` element into the `<office:forms>` forms container. The controls (see 11.3) will be left as is, except that they receive an `xforms:bind` attribute, which allows to bind any OpenDocument control to a previously defined XForms model.

11.2.1 XForms Model

We import the XForms model defined in [XForms]. In order to avoid duplication of the XForms schema here, we only specify the XForms model element and allow arbitrary content.

```
<define name="xforms-model">
    <element name="xforms:model">
        <ref name="anyAttListOrElements"/>
    </element>
</define>
```

11.3 Controls

Controls are used to interact with forms. Each control in a form is identified by a name, though the names must not necessarily be unique.

Controls are connected to a the surrounding document (and its text flow, if applicable) by binding them to a shape that acts as a placeholder for the control. See section 9.2.12 for details.

In addition to the attributes defined in this file format, controls may have application-specific additional attributes. These attributes are stored in the `<form:properties>` element in each control. Control events are specified in the `<office:event-listeners>` element.

When a user submits a form for processing, the names of some controls are paired with the current values of the controls and the pairs are submitted with the form. These controls are called successful controls. See section 17.13.2 of [HTML4]for more information.

The file format provides elements for the following standard controls:

- Text
- Text area
- Password
- File
- Formatted text
- Number
- Date
- Time
- Fixed text
- Combo box
- List box
- Button
- Image
- Check box
- Radio button
- Frame
- Image frame

- Hidden
- Grid

It is also possible to define application-specific controls. These controls are described by the `<form:generic-control>` element.

11.3.1 Text

The `<form:text>` element defines a control for displaying and inputting text.

```
<define name="column-controls" combine="choice">
    <element name="form:text">
        <ref name="form-text-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="controls" combine="choice">
    <ref name="column-controls"/>
</define>
<define name="form-text-attlist">
    <ref name="form-control-attlist"/>
    <ref name="common-current-value-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-convert-empty-attlist"/>
    <ref name="common-data-field-attlist"/>
</define>
<define name="form-control-attlist">
    <ref name="common-form-control-attlist"/>
    <ref name="common-control-id-attlist"/>
    <ref name="xforms-bind-attlist"/>
</define>
<define name="common-form-control-content">
    <optional>
        <ref name="form-properties"/>
    </optional>
    <optional>
        <ref name="office-event-listeners"/>
    </optional>
</define>
```

The attributes that may be associated with the `<form:text>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Convert Empty and Data Field. See section 11.5.22 for information about these attributes.

11.3.2 Text Area

The `<form:textarea>` element defines a control for displaying and inputting text on multiple lines.

The `<form:textarea>` element may be used with plain text values (specified by the `form:current-value` attribute) as well as with formatted text (specified as paragraph content). If both, the `form:current-value` and one or more `<text:p>` elements are present, it is up to the application reading the document to decide which information is used.

```
<define name="column-controls" combine="choice">
    <element name="form:textarea">
        <ref name="form-textarea-attlist"/>
        <ref name="common-form-control-content"/>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
<define name="form-textarea-attlist">
    <ref name="form-control-attlist"/>
    <ref name="common-current-value-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-convert-empty-attlist"/>
    <ref name="common-data-field-attlist"/>
</define>
```

The attributes that may be associated with the `<form:textarea>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Convert Empty and Data Field. See section 11.5.22 for information about these attributes.

11.3.3 Password

The `<form:password>` element defines a control that hides the text that a user inputs using an echo character, for example, an asterisk. This type of control is usually used for inputting sensitive information such as a password.

```
<define name="controls" combine="choice">
    <element name="form:password">
        <ref name="form-password-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-password-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-convert-empty-attlist"/>
</define>
```

The attributes that may be associated with the `<form:password>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Maximum Length, Printable, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Echo Char

Echo Char

The `form:echo-char` attribute specifies the character that the form uses to mask the text which a user inputs in a password control.

```
<define name="form-password-attlist" combine="interleave">
    <optional>
        <attribute name="form:echo-char" a:defaultValue="*">
            <ref name="character"/>
        </attribute>
    </optional>
</define>
```

11.3.4 File

The `<form:file>` element defines a control for selecting a file.

```
<define name="controls" combine="choice">
    <element name="form:file">
        <ref name="form-file-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-file-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-current-value-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
</define>
```

The attributes that may be associated with the `<form:file>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.

11.3.5 Formatted Text

The `<form:formatted-text>` element defines a control for inputting formatted text, which follows a certain formatting in both input and display.

```
<define name="column-controls" combine="choice">
    <element name="form:formatted-text">
        <ref name="form-formatted-text-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
```

```
<define name="form-formatted-text-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-current-value-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-convert-empty-attlist"/>
    <ref name="common-data-field-attlist"/>
</define>
```

The attributes that may be associated with the `<form:formatted-text>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Convert Empty and Data Field. See section 11.5.22 for information about these attributes.
- Maximum Value
- Minimum Value
- Validation

Maximum Value

The `form:max-value` attribute specifies the maximum value that a user can enter.

```
<define name="form-formatted-text-attlist" combine="interleave">
    <optional>
        <attribute name="form:max-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Minimum Value

The `form:min-value` attribute specifies the minimum value that a user can enter.

```
<define name="form-formatted-text-attlist" combine="interleave">
    <optional>
        <attribute name="form:min-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Validation

The `form:validation` attribute specifies whether or not the text that the user enters is validated during input.

```
<define name="form-formatted-text-attlist" combine="interleave">
    <optional>
        <attribute name="form:validation" a:defaultValue="false">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.3.6 Number

The `<form:number>` element describes a control which allows the user to enter a floating point number. The attributes that may be associated on this control are similar to those of the `<form:formatted-text>`, except that the data type is fixed to numeric data.

```
<define name="column-controls" combine="choice">
    <element name="form:number">
        <ref name="form-number-attlist"/>
        <ref name="common-numeric-control-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="common-numeric-control-attlist">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-convert-empty-attlist"/>
    <ref name="common-data-field-attlist"/>
</define>
```

The attributes that may be associated with the `<form:number>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Convert Empty and Data Field. See section 11.5.22 for information about these attributes.
- Value and Current Value
- Minimum and Maximum Value

Value

The attributes for value and current value are the same as those for other fields, except that they can contain only floating point data.

```
<define name="form-number-attlist" combine="interleave">
    <optional>
        <attribute name="form:value">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
<define name="form-number-attlist" combine="interleave">
    <optional>
        <attribute name="form:current-value">
            <ref name="double"/>
        </attribute>
    </optional>
```

```
</define>
```

Minimum and Maximum

The attributes for minimum and maximum value define the smallest and largest numerical values that are acceptable for this control.

```
<define name="form-number-attlist" combine="interleave">
    <optional>
        <attribute name="form:min-value">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
<define name="form-number-attlist" combine="interleave">
    <optional>
        <attribute name="form:max-value">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

11.3.7 Date And Time

The controls for date and time are the same as those for number values, except that they accept date and time values, respectively. They support the same attributes as the numerical field, except for the different data types of their value attributes.

```
<define name="column-controls" combine="choice">
    <element name="form:date">
        <ref name="form-date-attlist"/>
        <ref name="common-numeric-control-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="controls" combine="choice">
    <element name="form:time">
        <ref name="form-time-attlist"/>
        <ref name="common-numeric-control-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
```

The attributes that may be associated with the `<form:date>` and `<form:time>` elements are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Convert Empty and Data Field. See section 11.5.22 for information about these attributes.
- Value and Current Value
- Minimum and Maximum Value

Value

The attributes for value and current value are the same as those for `<form:number>`, except that they can contain only date or time data, respectively.

```
<define name="form-date-attlist" combine="interleave">
    <optional>
        <attribute name="form:value">
            <ref name="date"/>
        </attribute>
    </optional>
</define>
<define name="form-time-attlist" combine="interleave">
    <optional>
        <attribute name="form:value">
            <ref name="time"/>
        </attribute>
    </optional>
</define>
<define name="form-date-attlist" combine="interleave">
    <optional>
        <attribute name="form:current-value">
            <ref name="date"/>
        </attribute>
    </optional>
</define>
<define name="form-time-attlist" combine="interleave">
    <optional>
        <attribute name="form:current-value">
            <ref name="time"/>
        </attribute>
    </optional>
</define>
```

Minimum and Maximum

The attributes for minimum and maximum value define the smallest and largest dates (or times) that are acceptable for this control.

```
<define name="form-date-attlist" combine="interleave">
    <optional>
        <attribute name="form:min-value">
            <ref name="date"/>
        </attribute>
    </optional>
</define>
<define name="form-time-attlist" combine="interleave">
    <optional>
        <attribute name="form:min-value">
            <ref name="time"/>
        </attribute>
    </optional>
</define>
<define name="form-date-attlist" combine="interleave">
    <optional>
        <attribute name="form:max-value">
            <ref name="date"/>
        </attribute>
    </optional>
</define>
<define name="form-time-attlist" combine="interleave">
    <optional>
        <attribute name="form:max-value">
            <ref name="time"/>
        </attribute>
    </optional>
```

```
</define>
```

11.3.8 Fixed Text

The `<form:fixed-text>` element describes a control which attaches additional information to controls, or merely displays information in the application. Relations between a labeling and a labeled control can be established by specifying the `form:for` attribute of the label. Only one label may be associated with the same control.

```
<define name="controls" combine="choice">
    <element name="form:fixed-text">
        <ref name="form-fixed-text-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-fixed-text-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="for"/>
    <ref name="common-disabled-attlist"/>
    <ref name="label"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-title-attlist"/>
</define>
```

The attributes that may be associated with the `<form:fixed-text>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, For, Label, Printable, and Title. See section 11.5 for information about these attributes.
- Multi-Line

Multi-Line

The `form:multi-line` attribute specifies whether or not the label is displayed on multiple lines.

```
<define name="form-fixed-text-attlist" combine="interleave">
    <optional>
        <attribute name="form:multi-line" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.3.9 Combo Box

The `<form:combobox>` element defines a control which allows displaying and editing of text, and containing a list of possible values for this text.

```
<define name="column-controls" combine="choice">
    <element name="form:combobox">
        <ref name="form-combobox-attlist"/>
        <ref name="common-form-control-content"/>
        <zeroOrMore>
            <ref name="form-item"/>
        </zeroOrMore>
    </element>
</define>
<define name="form-combobox-attlist" combine="interleave">
```

```
    <ref name="form-control-attlist"/>
    <ref name="common-current-value-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="dropdown"/>
    <ref name="common-maxlength-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="size"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-convert-empty-attlist"/>
    <ref name="common-data-field-attlist"/>
    <ref name="list-source"/>
    <ref name="list-source-type"/>
</define>
```

The attributes that may be associated with the `<form:combobox>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Dropdown, Max Length, Printable, Read only, Size, Tab Index, Tab Stop, Title, and Value. See section 11.5 for information about these attributes.
- Convert Empty, Data Field, List Source, and List Source Type. See section 11.5.22 for information about these attributes.
- Automatic Completion

Automatic Completion

The `form:auto-complete` attribute specifies whether, when the user enters text in the combobox that matches one of the list items in the combobox, the application automatically completes the text for the user.

```
<define name="form-combobox-attlist" combine="interleave">
    <optional>
        <attribute name="form:auto-complete">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Item

The `<form:item>` element defines a list item for a combobox control.

```
<define name="form-item">
    <element name="form:item">
        <ref name="form-item-attlist"/>
        <text/>
    </element>
</define>
<define name="form-item-attlist" combine="interleave">
    <ref name="label"/>
</define>
```

The attribute that may be associated associate with the `<form:item>` element is:

- Label. See section 11.5 for information about this attribute.

11.3.10 List Box

The `<form:listbox>` element defines an input control that allows a user to select one or more items from a list. It is an alternative representation for a group of radio buttons.

```
<define name="column-controls" combine="choice">
    <element name="form:listbox">
        <ref name="form-listbox-attlist"/>
        <ref name="common-form-control-content"/>
        <zeroOrMore>
            <ref name="form-option"/>
        </zeroOrMore>
    </element>
</define>
<define name="form-listbox-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="dropdown"/>
    <ref name="common-printable-attlist"/>
    <ref name="size"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="bound-column"/>
    <ref name="common-data-field-attlist"/>
    <ref name="list-source"/>
    <ref name="list-source-type"/>
</define>
```

The attributes that may be associated with the `<form:listbox>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Dropdown, Printable, Read only, Size, Tab Index, Tab Stop, and Title. See section 11.5 for information about these attributes.
- Bound Column, Data Field, List Source, and List Source Type. See section 11.5.22 for information about these attributes.
- Multiple
- XForms source

Multiple

The `form:multiple` attribute determines whether or not a user can select multiple items from a list box.

```
<define name="form-listbox-attlist" combine="interleave">
    <optional>
        <attribute name="form:multiple" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

XForms source

The `form:xforms-list-source` allows to dynamically create the list of choices by binding the list content to XForms (see section 11.2, as well as [XForms]). The attribute references an `<xforms:bind>` element, and creates a list entry for each node in the node-set defined by that attribute.

```
<define name="form-listbox-attlist" combine="interleave">
    <optional>
        <attribute name="form:xforms-list-source">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Option

The `<form:option>` element defines the list items for a list box control. An item can be preselected and can contain a related value.

```
<define name="form-option">
    <element name="form:option">
        <ref name="form-option-attlist"/>
        <text/>
    </element>
</define>
<define name="form-option-attlist" combine="interleave">
    <ref name="current-selected"/>
    <ref name="selected"/>
    <ref name="label"/>
    <ref name="common-value-attlist"/>
</define>
```

The attributes that may be associated with the `<form:option>` element are:

- Current Selected, Selected, Label, and Value. See section 11.5 for information about these attributes.

11.3.11 Button

The `<form:button>` element defines a button. When pressed, a button usually triggers an action.

```
<define name="controls" combine="choice">
    <element name="form:button">
        <ref name="form-button-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-button-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="button-type"/>
    <ref name="common-disabled-attlist"/>
    <ref name="label"/>
    <ref name="image-data"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="target-frame"/>
    <ref name="target-location"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-form-relative-image-position-attlist"/>
</define>
```

The attributes that may be associated with the `<form:button>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.

- Button Type, Control ID, Disabled, Image Data, Printable, Tab Index, Tab Stop, Target Frame, Target Location, Title, Value and relative image position. See section 11.5 for information about these attributes.
- Default Button
- Toggle
- Focus on Click
- XForms Submission

Default Button

The `form:default-button` attribute determines whether or not the button is the default button on the form. If a user clicks the default button or presses Return while an input control is focused, the application takes the same action.

If a form contains more than one default button, the behavior of the application is undefined.

```
<define name="form-button-attlist" combine="interleave">
    <optional>
        <attribute name="form:default-button" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Toggle

The `form:toggle` attribute specifies whether a form button control, when it is operated (via mouse or keyboard), should be toggled between a "pressed" and a "not pressed" state. If this attribute is set to `false`, the button controls behaves like a push button.

```
<define name="form-button-attlist" combine="interleave">
    <optional>
        <attribute name="form:toggle" a:default-value="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Focus on click

The `form:focus-on-click` attribute specifies whether a form button control should grab the focus when it is clicked with the mouse.

```
<define name="form-button-attlist" combine="interleave">
    <optional>
        <attribute name="form:focus-on-click">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

XForms Submission

Buttons may be used to trigger an XForms submission by adding an `form:xforms-submission` attribute. If such a button is triggered, a previously declared XForms submission with the given name is executed.

```
<define name="form-button-attlist" combine="interleave">
    <optional>
        <attribute name="form:xforms-submission">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.3.12 Image

The `<form:image>` element defines a graphical button control. This element corresponds to the input element of type image in HTML 4.01. Note: HTML 4.01 only allows the button type to be “submit” for an image button. In office application file format, an image button can be of any type.

```
<define name="controls" combine="choice">
    <element name="form:image">
        <ref name="form-image-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-image-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="button-type"/>
    <ref name="common-disabled-attlist"/>
    <ref name="image-data"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="target-frame"/>
    <ref name="target-location"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
</define>
```

The attributes that may be associated with the <form:image> element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Button Type, Control ID, Disabled, Image Data, Printable, Tab Index, Tab Stop, Target Frame, Target Location, Title, and Value. See section 11.5 for information about these attributes.

11.3.13 Check Box

The `<form:checkbox>` element defines an on/off control which a user can toggle. The control is `on` when the value of the `form:current-state` attribute associated with the control element is `checked`. When a user submits a form, only the controls whose current state is checked are successful.

```
<define name="column-controls" combine="choice">
    <element name="form:checkbox">
        <ref name="form-checkbox-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-checkbox-attlist" combine="interleave">
```

```
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="label"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-data-field-attlist"/>
    <ref name="common-form-visual-effect-attlist"/>
    <ref name="common-form-relative-image-position-attlist"/>
</define>
```

The attributes that may be associated with the `<form:checkbox>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Label, Printable, Tab Index, Tab Stop, Title, Value, Visual Effect and Relative Image Position. See section 11.5 for information about these attributes.
- Data Field. See section 11.5.22 for information about this attribute.
- Current State
- Is Tristate
- State

Current State

The `form:current-state` attribute specifies the current state of the check box control.

The value of this attribute can be one of the following:

- `unchecked`: The check box is not checked.
- `checked`: The check box is checked. The value of the control is submitted with the form.
- unknown: This value is only available when the control is in tristate mode (See the "Is Tristate" attribute) . This value may, for instance, be used in connection with a database field binding to indicate that the value is NULL.

```
<define name="states">
    <choice>
        <value>unchecked</value>
        <value>checked</value>
        <value>unknown</value>
    </choice>
</define>
<define name="form-checkbox-attlist" combine="interleave">
    <optional>
        <attribute name="form:current-state">
            <ref name="states"/>
        </attribute>
    </optional>
</define>
```

Is Tristate

The `form:is-tristate` attribute specifies that the check box can have three states instead of the common two states.

```
<define name="form-checkbox-attlist" combine="interleave">
    <optional>
        <attribute name="form:is-tristate" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

State

The `form:state` attribute specifies the default state of the check box control. This state is used to initialize the control.

```
<define name="form-checkbox-attlist" combine="interleave">
    <optional>
        <attribute name="form:state" a:defaultValue="unchecked">
            <ref name="states"/>
        </attribute>
    </optional>
</define>
```

11.3.14 Radio Button

The `<form:radio>` element describes controls which act like check boxes except that when several radio buttons share the same control name they are mutually exclusive. When one button is on, all of the other buttons with the same name are off. If no radio button is initially on, the way in which the application chooses which button to turn on initially is undefined.

If a group of radio buttons is bound to one database field, the reference value of the selected radio button is written into the database field.

```
<define name="controls" combine="choice">
    <element name="form:radio">
        <ref name="form-radio-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-radio-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="current-selected"/>
    <ref name="common-disabled-attlist"/>
    <ref name="label"/>
    <ref name="common-printable-attlist"/>
    <ref name="selected"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
    <ref name="common-data-field-attlist"/>
    <ref name="common-form-visual-effect-attlist"/>
    <ref name="common-form-relative-image-position-attlist"/>
</define>
```

The attributes that may be associated with the `<form:radio>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Selected, Disabled, Label, Printable, Selected, Tab Index, Tab Stop, Title, Value, Visual Effect and Relative Image Position. See section 11.5 for information about these attributes.
- Data Field. See section 11.5.22 for information about this attribute.

11.3.15 Frame

The `<form:frame>` element defines a frame, which may be used to arrange controls visually. This element does not have a value and it does not allow any user input.

```
<define name="controls" combine="choice">
    <element name="form:frame">
        <ref name="form-frame-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-frame-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="for"/>
    <ref name="label"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-title-attlist"/>
</define>
```

The attributes that may be associated with the `<form:frame>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, For, Label, Printable, and Title. See section 11.5 for information about these attributes.

11.3.16 Image Frame

The `<form:image-frame>` element defines a graphical control. The control displays an image, whose location is described in the control.

```
<define name="controls" combine="choice">
    <element name="form:image-frame">
        <ref name="form-image-frame-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-image-frame-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="image-data"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-readonly-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-data-field-attlist"/>
</define>
```

The attributes that may be associated with the `<form:image-frame>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Image Data, Printable, Read only, and Title. See section 11.5 for information about these attributes.
- Data Field. See section 11.5.22 for information about this attribute.

11.3.17 Hidden

The `<form:hidden>` element defines a control that does not have a visual representation. This element is usually used as a container for information.

```
<define name="controls" combine="choice">
    <element name="form:hidden">
        <ref name="form-hidden-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-hidden-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-value-attlist"/>
</define>
```

The attributes that may be associated with the `<form:hidden>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Value. See section 11.5 for information about this attribute.

11.3.18 Grid

The `<form:grid>` element defines a control that displays table data. This control is data-aware and is bound to a form which retrieves data from a data source. The actual data to display in a grid control is determined by the parent form, which is data-aware and thus based on a certain row set. The rows in the grid contain these data rows.

Each column in the grid is specified by a `<form:column>` element. Each column is bound to a field in the form's row set.

```
<define name="controls" combine="choice">
    <element name="form:grid">
        <ref name="form-grid-attlist"/>
        <ref name="common-form-control-content"/>
        <zeroOrMore>
            <ref name="form-column"/>
        </zeroOrMore>
    </element>
</define>
<define name="form-grid-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
</define>
```

The attributes that may be associated with the `<form:grid>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Disabled, Printable, Tab Index, Tab Stop, and Title. See section 11.5 for information about these attributes.

Column

The `<form:column>` element defines a column in a grid control. The column contains a control that displays the grid data for the column.

```
<define name="form-column">
    <element name="form:column">
        <ref name="form-column-attlist"/>
        <oneOrMore>
            <ref name="column-controls"/>
        </oneOrMore>
    </element>
</define>
<define name="form-column-attlist" combine="interleave">
    <ref name="common-form-control-attlist"/>
    <ref name="label"/>
    <ref name="text-style-name"/>
</define>
```

The attributes that may be associated with the `<form:column>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Label. See section 11.5 for information about this attribute.
- Column Style

Column Style

The `form:text-style-name` attribute specifies paragraph style that is applied to all controls with the column. See also section 9.2.12. Unlike other paragraph styles, this style may reference a data style.

```
<define name="text-style-name">
    <optional>
        <attribute name="form:text-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

11.3.19 Value Range

The new `<form:value-range>` element defines a control which allows the user to select a value from a continuous number range. Possible representations include scroll bars and spin buttons.

```
<define name="controls" combine="choice">
    <element name="form:value-range">
        <ref name="form-value-range-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-value-range-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
    <ref name="common-disabled-attlist"/>
    <ref name="common-printable-attlist"/>
    <ref name="common-tab-attlist"/>
    <ref name="common-title-attlist"/>
    <ref name="common-value-attlist"/>
</define>
```

The attributes that may be associated with a `<form:value-range>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.
- Control ID, Current Value, Disabled, Printable, Read only, Tab Index, Tab Stop, Title and Value. See section 11.5 for information about these attributes.
- Maximum Value

- Minimum Value
- Step Size
- Page Step Size
- Repeat Delay
- Orientation

Maximum Value

The `form:max-value` attribute specifies the maximum value that a user can enter.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:max-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Minimum Value

The `form:min-value` attribute specifies the minimum value that a user can enter.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:min-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Step Size

The `form:step-size` attribute specifies the increment to be used for a control representing a value.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:step-size" a:defaultName="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Page Step Size

The `form:page-step-size` attribute specifies a second-level increment to be used for a control representing a value. In the user interface, this is usually associated with the user pressing the "Page Up" or "Page Down" key.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:page-step-size">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Repeat Delay

The `form:delay-for-repeat` attribute specifies a time-out to be used before a pressed mouse button results in repeating an action.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:delay-for-repeat">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

Orientation

The `form:orientation` attribute specifies the orientation of the control, which could be either horizontal or vertical.

```
<define name="form-value-range-attlist" combine="interleave">
    <optional>
        <attribute name="form:orientation">
            <choice>
                <value>horizontal</value>
                <value>vertical</value>
            </choice>
        </attribute>
    </optional>
</define>
```

11.3.20 Generic Control

The `<form:generic-control>` element defines a placeholder for a generic control. The generic control can contain any properties and any events. The application detects the type of the control and instantiates the correct control.

```
<define name="controls" combine="choice">
    <element name="form:generic-control">
        <ref name="form-generic-control-attlist"/>
        <ref name="common-form-control-content"/>
    </element>
</define>
<define name="form-generic-control-attlist" combine="interleave">
    <ref name="form-control-attlist"/>
</define>
```

The attributes that may be associated with the `<form:generic-control>` element are:

- Name and Service Name. See section 11.4 for information about these attributes.

11.4 Common Form and Control Attributes

11.4.1 Name

The `form:name` attribute specifies the name of the form or control element. This may be used to give a form or control element an identity, which is important for scripting and for submitting the content of controls.

```
<define name="common-form-control-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="form:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.4.2 Control Implementation

A control may be given a control type attribute, which determines which concrete rendition or implementation the user agent should instantiate. For easy extensibility, the value of this attribute is a namespaced token, i.e., it is token using a namespace prefix, much like attributes in XML.

```
<define name="common-form-control-attlist" combine="interleave">
    <optional>
        <attribute name="form:control-implementation">
            <ref name="namespacedToken"/>
        </attribute>
    </optional>
</define>
```

11.4.3 Bind to XForms

Any control can be bound to an XForms form (see section 11.2, as well as [XForms]) by using the `xforms:bind` attribute. With buttons the bind attribute refers to an `<xforms:submission>` element with the given ID. Pushing the button causes the appropriate XForms submission action to be performed. For all other control types, the `xforms:bind` attribute refers to an `<xforms:bind>` element with the given ID. Any such bound control reads and writes its data as determined by the appropriate bind element.

```
<define name="xforms-bind-attlist">
    <optional>
        <attribute name="xforms:bind">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.5 Common Control Attributes

11.5.1 Button Type

The `form:button-type` attribute specifies the type of a button. This attribute is supported for the following elements:

- `<form:button>`
- `<form:image>`

The value of this attribute can be one of the following:

- `submit`: Pressing the button submits the form.
- `reset`: Pressing the button resets every control in the form to its default value.
- `push`: Pressing the button does not perform any action by default. The use then can add scripts to the button. and the script is run when the button is pressed.

- `url`: Pressing the button loads the URL that is specified in the `form:target-url` attribute.

```
<define name="types">
    <choice>
        <value>submit</value>
        <value>reset</value>
        <value>push</value>
        <value>url</value>
    </choice>
</define>
<define name="button-type">
    <optional>
        <attribute name="form:button-type" a:defaultValue="push">
            <ref name="types"/>
        </attribute>
    </optional>
</define>
```

11.5.2 Control ID

All controls except Hidden Controls have a visual representation in the host document. Thus, they need an absolute or relative position, describing the location in the document. The position is represented by a shape that contains a reference to the control element within the form element.

The `form:id` attribute is used to uniquely identify a control element. Every control that is not hidden must have such an attribute associated with it, which in turn can be used to reference the control.

This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:image-frame>`
- `<form:grid>`

```
<define name="common-control-id-attlist">
    <attribute name="form:id">
```

```
        <ref name="ID"/>
    </attribute>
</define>
```

11.5.3 Current Selected

The `form:current-selected` attribute determines the current state of a radio button or option element.

This attribute is supported for the following elements:

- `<form:option>`
- `<form:radio>`

```
<define name="current-selected">
    <optional>
        <attribute name="form:current-selected" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.4 Value and Current Value

Every control has a default value and a current value. The current value changes with user interaction; the default value of a control does not. In general, the default value is specified in a `form:value` attribute.

The default value is used during special events, such as resetting the form, which transfers the default value of every control to its current value. If a control does not have a default value, the result of resetting the form is undefined.

Besides storing the current value together with the control, it is also possible to bind controls to other value providers, which act as value sink and source, such as database fields (in data-aware forms) or e.g., cells in a spreadsheet document the controls live in. In this case, the current value is not stored with the control itself, but in the external instance, which may or may not store it together with the document. See section 11.5.22 for more details on database properties.

Default Value

The `form:value` attribute specifies the default value of an input control. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:option>`
- `<form:button>`

- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:hidden>`

```
<define name="common-value-attlist">
    <optional>
        <attribute name="form:value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Current Value

The `form:current-value` attribute specifies the current status of an input control. It overrides the value of a `form:value` attribute, if one is present.

This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`

```
<define name="common-current-value-attlist">
    <optional>
        <attribute name="form:current-value">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.5.5 Disabled

The `form:disabled` attribute specifies whether or not a control can accept user input. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`

- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:image-frame>`
- `<form:grid>`

Controls that are disabled are not included in the tabbing navigation sequence and can not be focused.

```
<define name="common-disabled-attlist">
    <optional>
        <attribute name="form:disabled" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.6 Dropdown

The `form:dropdown` attribute specifies whether the list in a combo box or list box is always visible or is only visible when the user clicks the drop-down button. This attribute is supported for the following elements:

- `<form:combobox>`
- `<form:listbox>`

If the value is `true`, the list is always visible. If the value is `false`, the list is only visible when the user clicks the drop-down button.

```
<define name="dropdown">
    <optional>
        <attribute name="form:dropdown" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.7 For

The `form:for` attribute specifies the IDs of the controls with which control element is labeling. This attribute is supported for the following elements:

- `<form:fixed-text>`
- `<form:frame>`

This attribute contains a comma separated list of control IDs.

```
<define name="for">
    <optional>
        <attribute name="form:for">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.5.8 Image Data

The `form:image-data` attribute links the control to an external file containing image data. This attribute is supported for the following elements:

- `<form:button>`
- `<form:image>`
- `<form:image-frame>`

```
<define name="image-data">
    <optional>
        <attribute name="form:image-data">
            <ref name="anyURI"/>
        </attribute>
    </optional>
</define>
```

11.5.9 Label

The `form:label` attribute contains a label for a control such as a radio button or check box. This attribute is supported for the following elements:

- `<form:fixed-text>`
- `<form:item>`
- `<form:option>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:column>`

```
<define name="label">
    <optional>
        <attribute name="form:label">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

11.5.10 Maximum Length

The `form:max-length` attribute specifies the maximum number of characters that a user can enter in an input control. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`

- `<form:formatted-text>`
- `<form:combobox>`

The default value of this attribute is unlimited, which allows a user to enter an unlimited number of characters.

```
<define name="common-maxlength-attlist">
    <optional>
        <attribute name="form:max-length">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

11.5.11 Printable

The `form:printable` attribute specifies whether or not a control is printed when a user prints the document in which the control is contained. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:image-frame>`
- `<form:grid>`

```
<define name="common-printable-attlist">
    <optional>
        <attribute name="form:printable" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.12 Read only

The `form:readonly` attribute specifies whether or not a user can modify the value of a control. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:image-frame>`

Read-only controls are included in the tabbing navigation sequence.

```
<define name="common-readonly-attlist">
    <optional>
        <attribute name="form:readonly" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.13 Selected

The `form:selected` attribute specifies the default state of a radio button or option. When the control is initialized, it is in the default state specified by this attribute. This attribute is supported for the following elements:

- `<form:option>`
- `<form:radio>`

In a group of radio buttons that share the same name, only one radio button can have this attribute set to true.

```
<define name="selected">
    <optional>
        <attribute name="form:selected" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.14 Size

The `form:size` attribute specifies the number of rows that are visible at a time in a combo box list or a list box list. This attribute is supported for the following elements:

- `<form:combobox>`
- `<form:listbox>`

```
<define name="size">
    <optional>
```

```
        <attribute name="form:size">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

11.5.15 Tab Index

The `form:tab-index` attribute specifies the tabbing navigation order of a control within a form. The tabbing order is the order in which controls are given focus when a user navigates through the form using the TAB key on the keyboard. The tabbing order can include elements that are nested in other elements. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:grid>`

The rules for tabbing are similar to the tabbing rules used in HTML 4.0.

Controls that can be given focus are navigated in the order described in the following rules:

1. The controls that have a positive value for the `form:tab-index` attribute are navigated first.
2. The navigation starts at the control with lowest `form:tab-index` value and ends at the control with the highest value. Values do not have to be sequential and they do not have to begin with a particular value.
3. Controls that have the same values for the form:tab-index attribute are navigated according their position in the form.
4. Controls that do not contain the `form:tab-index` attribute or contain the attribute with a value of 0 are navigated next. These controls are navigated according to their position in the form.
5. Controls that have the `form:disabled` attribute set to `true` are not included in the navigation, independent on their `form:tab-index` value.

```
<define name="common-tab-attlist" combine="interleave">
    <optional>
        <attribute name="form:tab-index" a:defaultValue="0">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

11.5.16 Tab Stop

The `form:tab-stop` attribute specifies whether or not a control is included in the tabbing navigation order. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:grid>`

If the value is false, the control is not included in the tabbing navigation.

```
<define name="common-tab-attlist" combine="interleave">
    <optional>
        <attribute name="form:tab-stop" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

11.5.17 Target Frame

The `office:target-frame` attribute specifies the link target frame of the area. This attribute is supported for the following elements:

- `<form:button>`
- `<form:image>`

```
<define name="target-frame">
    <optional>
        <attribute name="office:target-frame" a:defaultValue="_blank">
            <ref name="targetFrameName"/>
        </attribute>
    </optional>
</define>
```

11.5.18 Target Location

An `xlink:href` attribute specifies the URL that is loaded if a button is clicked. This attribute is supported for the following elements:

- `<form:button>`
- `<form:image>`

This attribute is only evaluated if the value of the `form:button-type` attribute is `location`.

```
<define name="target-location">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
    </optional>
</define>
```

11.5.19 Title

The `form:title` attribute contains additional information about a control. The value of the attribute can be used as a tool tip. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:image>`
- `<form:image-frame>`
- `<form:grid>`

```
<define name="common-title-attlist">
    <optional>
        <attribute name="form:title"/>
    </optional>
</define>
```

11.5.20 Visual Effect

The `form:visual-effect` attributes specifies a visual affect to apply to a control. The attribute values can be `flat` for a flat visual effect and `3d` for a 3D effect. This attribute is supported for the following elements:

- `<form:checkbox>`
- `<form:radio>`

```
<define name="common-form-visual-effect-attlist" combine="interleave">
    <optional>
        <attribute name="form:visual-effect">
            <choice>
                <value>flat</value>
                <value>3d</value>
            </choice>
        </attribute>
    </optional>
</define>
```

11.5.21 Relative Image Position

The `form:image-position` and `form:image-align` together specify the position of an image to be displayed in a form control, relative to the label text.

If the `form:image-position` attribute has the value `center`, the image shown in a control should be centered relative to the control's text.

If the `form:image-position` attribute has one of the values `start`, `end`, `top`, `bottom`, the image is to be placed before, after, above, or below the text. In this case, the `form:image-align` attribute specifies which border (`start`, `end`) or axis (`center`) of the image and the text are to be aligned. If the `form:image-position` attribute is not present, it is assumed to be `center`. The `form:image-position` and `form:image-align` attributes are supported for the following elements:

- `<form:button>`
- `<form:checkbox>`
- `<form:radio>`

```
<define name="common-form-relative-image-position-attlist"
        combine="interleave">
    <choice>
        <optional>
            <attribute name="form:image-position" a:defaultValue="center">
                <value>center</value>
            </attribute>
        </optional>
        <group>
            <attribute name="form:image-position">
                <choice>
                    <value>start</value>
                    <value>end</value>
                    <value>top</value>
                    <value>bottom</value>
                </choice>
            </attribute>
            <optional>
```

```
                <attribute name="form:image-align" a:defaultValue="center">
                    <choice>
                        <value>start</value>
                        <value>center</value>
                        <value>end</value>
                    </choice>
                </attribute>
            </optional>
        </group>
    </choice>
</define>
```

11.5.22 Database Binding Attributes

A control may be bound to a database fields. In this case, the controls becomes data-aware. The control acquires the values of a database field by going through a result set that is provided by the form. Each time there is a row change in the form, the value of the control may change. The value changes are stored in the associated database field.

Bound Column

The `form:bound-column` attribute specifies the column values of the list source result set that are used to fill the data field values. This attribute is supported for the `<form:listbox>` element.

```
<define name="bound-column">
    <optional>
        <attribute name="form:bound-column">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Convert Empty To Null

The `form:convert-empty-to-null` attribute specifies whether or not empty current values are regarded as NULL This attribute is important for data-aware controls to determine which values to store for the bound database field. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:formatted-text>`
- `<form:combobox>`

If the value of the attribute is `true`, an empty string in the control is regarded as the dedicated NULL value. If the value of the attribute is `false`, an empty string in the control is regarded as an empty string.

```
<define name="common-convert-empty-attlist">
    <optional>
        <attribute name="form:convert-empty-to-null" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Data Field

The `form:data-field` attribute specifies the name of a result set column. The result set is determined by the form which the control belongs to. This attribute is supported for the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:image-frame>`

```
<define name="common-data-field-attlist">
    <optional>
        <attribute name="form:data-field">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

List Source

The `form:list-source` attribute specifies the source used to populate the list in a list box or combo box. The first column of the list source result set populates the list. This attribute is supported for the following elements:

- `<form:combobox>`
- `<form:listbox>`

```
<define name="list-source">
    <optional>
        <attribute name="form:list-source">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

List Source Type

The `form:list-source-type` attribute specifies the type of data source that is used to populates the list data in a list box or combo box. This attribute is supported for the following elements:

- `<form:combobox>`
- `<form:listbox>`

The value of this attribute can be one of the following:

- `table`: The list is populated using the content of a database table.

- `query`: The list is populated by executing a query.
- `sql`: The list is populated by executing an SQL statement.
- `sql-pass-through`: The list is populated by executing any type of statement that is passed directly to a database driver, without being interpreted by the application.
- `value-list`: The list is populated with values specified by the user using the `form:value` attribute in the `<form:option>` element. This setting is only applicable to list boxes.
- `table-fields`: The list is populated using the field names in a database table.

```
<define name="list-source-type">
    <optional>
        <attribute name="form:list-source-type">
            <choice>
                <value>table</value>
                <value>query</value>
                <value>sql</value>
                <value>sql-pass-through</value>
                <value>value-list</value>
                <value>table-fields</value>
            </choice>
        </attribute>
    </optional>
</define>
```

11.6 Event Listeners

Forms and form controls may have event listeners attached. The event listeners that are attached to, for example, a list box or button, are represented by an event listener element as described in section 12.4. This element is contained within the form or form control element, for example, the `<form:listbox>` element or the `<form:button>` element.

Section 12.4.1 contains guidelines for event names that may be used within forms and form controls. In addition to those, the following events may be used for forms and form controls.

Value of script:event-name Attribute	***Applies To***	***Description of Event***
`form:approveaction`	Button or image.	Occurs before the “on performaction” event takes place. Allows the user to veto the action.
`form:performaction`	Button or image.	Occurs when the control action is to be performed. The common interpretation of this event is “pressing the button”.
`form:textchange`	All controls that allow text input.	Occurs when a user changes the text in a control.
`form:itemstatechange`	Check box or radio button.	Occurs when the state of a check box or radio button changes.
`form:mousedrag`	All controls.	Occurs when a user presses and holds one of the mouse buttons and moves the mouse pointer onto a control.
`form:approvereset`	same objects as for form:on-reset	Occurs before the on-reset event takes place. Allows the user to veto the reset event.

Value of script:event-name Attribute	*Applies To*	*Description of Event*
`form:approveupdate`	All controls that can be bound to a database field, that is controls that contain the data-field attribute.	Occurs before the on-update event takes place. Allows the user to veto the update.
`form:update`	All controls that can be bound to a database field, that is controls that contain the data-field attribute.	Occurs when the content of a control that is bound to a database field is committed.
`form:load`	Forms.	Occurs when the form establishes a connection to the data source.
`form:startrealod`	Forms.	Occurs when the form is about to refresh a data source connection.
`form:reload`	Forms.	Occurs when the form has refreshed a data source connection.
`form:startunload`	Forms.	Occurs when the form is about to drop a data source connection.
`form:unload`	Forms.	Occurs when the form has dropped a data source connection.
`form:confirmdelete`	Forms.	Occurs when the user is about to delete a record.
`form:approverowchange`	Forms.	Occurs before the “on rowchange” event takes place. Allows the user to veto the change.
`form:rowchange`	Forms.	Occurs after changes to a row are complete, such as deletions, updates, and insertions.
`form:approvecursormove`	Forms.	Occurs before the form is moved to another row. Allows the user to veto the move.
`form:cursormove`	Forms.	Occurs after the form is moved to another row.
`form:supplyparameter`	Forms.	Occurs when the form needs to fill parameters to connect to a data source.
`form:error`	Forms, combo boxes and list boxes.	Occurs when a database-related error occurs.
`form:adjust`	Value Range	Occurs when the value of a Value Range element has been adjusted.

11.7 Properties

The `<form:properties>` element may be used to store the following settings for controls and forms:

- Settings that are not known by the document format.
- Settings that are provided by external vendors.
- Settings that are specific to the application.

Properties consist of a name/value pair. The name identifies the property. The value can be given in a fundamental data type or as a list of fundamental data types.

11.7.1 Property Set

The `<form:properties>` element contains the property elements. Properties are encoded using the `form:property` element, except for list properties, which make use of the `form:list-property` element.

```
<define name="form-properties">
    <element name="form:properties">
        <oneOrMore>
            <ref name="form-property"/>
        </oneOrMore>
    </element>
</define>
```

11.7.2 Property

The `<form:property>` element describes a single property, and contains its name, type and value.

```
<define name="form-property" combine="choice">
    <element name="form:property">
        <ref name="form-property-name"/>
        <ref name="form-property-value-and-type-attlist"/>
    </element>
</define>
```

Property Name

The `form:property-name` attribute specifies the name of a property element.

```
<define name="form-property-name" combine="interleave">
    <attribute name="form:property-name">
        <ref name="string"/>
    </attribute>
</define>
```

Property Value and Type

The value and type of form properties are represented through the common `office:value-type` and suitable value attributes. See section 6.7.1for more information on these attributes.

In addition to these value types, form properties can also be empty. This is represented by the special value type `void`. Such properties have no value attribute.

```
<define name="form-property-value-and-type-attlist" combine="interleave">
    <choice>
        <ref name="common-value-and-type-attlist"/>
        <attribute name="office:value-type">
            <value>void</value>
        </attribute>
    </choice>
</define>
```

11.7.3 List Property

The `<form:list-property>` element specifies a property that contains a list of values. A value type attribute determines which types are allowed on the list elements. The element contains a sequence of list value elements, each of which contains a value attribute suitable to the value type given in the `<form:list-property>` element. The value attributes are the same as those used elsewhere in the specification, except that the type attribute is attached to the container element, which the value attributes are attached to the list values. (See section 6.7.1 for more information on vale and value type attributes.)

```
<define name="form-property" combine="choice">
    <element name="form:list-property">
        <ref name="form-property-name"/>
        <ref name="form-property-type-and-value-list"/>
    </element>
</define>
```

List Value

The list value element contains value attributes for the value type given in the containing `<form:list-property>` element.

```
<define name="form-property-type-and-value-list">
    <choice>
        <group>
            <attribute name="office:value-type">
                <value>float</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:value">
                        <ref name="double"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>percentage</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:value">
                        <ref name="double"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>currency</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:value">
                        <ref name="double"/>
                    </attribute>
                    <optional>
                        <attribute name="office:currency">
                            <ref name="string"/>
                        </attribute>
                    </optional>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>date</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:date-value">
                        <ref name="dateOrDateTime"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>time</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:time-value">
                        <ref name="duration"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>boolean</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:boolean-value">
                        <ref name="boolean"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <group>
            <attribute name="office:value-type">
                <value>string</value>
            </attribute>
            <zeroOrMore>
                <element name="form:list-value">
                    <attribute name="office:string-value">
                        <ref name="string"/>
                    </attribute>
                </element>
            </zeroOrMore>
        </group>
        <attribute name="office:value-type">
            <value>void</value>
        </attribute>
    </choice>
</define>
```

Example: Form properties

The following contains a string property “Name” with value “Name 1”, and a string list property “Items” containing the strings “Item 1”, “Item 2”, “Item 3”.

```
<form:properties>
    <form:property form:property-name="Name"
                   office:value-type="string"
                   office:string-value="Name 1">
    <form:list-property form:property-name="Items"
                   office:value-type="string" >
        <form:list-value office:string-value="Item 1"/>
        <form:list-value office:string-value="Item 2"/>
        <form:list-value office:string-value="Item 3"/>
    </form:list-property>
</form:properties>
```

12 Common Content

12.1 Annotation

The `<office:annotation>` element specifies an OpenDocument annotation. The annotation's text is contained in `<text:p>` and `<text:list>` elements.

```
<define name="office-annotation">
    <element name="office:annotation">
        <ref name="office-annotation-attlist"/>
        <ref name="draw-caption-attlist"/>
        <ref name="common-draw-position-attlist"/>
        <ref name="common-draw-size-attlist"/>
        <ref name="common-draw-shape-with-text-and-styles-attlist"/>
        <optional>
            <ref name="dc-creator"/>
        </optional>
        <optional>
            <ref name="dc-date"/>
        </optional>
        <optional>
            <ref name="meta-date-string"/>
        </optional>
        <zeroOrMore>
            <choice>
                <ref name="text-p"/>
                <ref name="text-list"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

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

- Display
- Position, size, style, layer, z-index, id, and transformation (see section 9.2.15)
- Text anchor, table background, draw end position (see section 9.2.16)
- Caption point, round corners (see section 9.2.10)

Display

The `office:display` attribute specifies whether or not the annotation is visible.

```
<define name="office-annotation-attlist" combine="interleave">
    <optional>
        <attribute name="office:display">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Caption Attributes

The following attributes can be attached to the `<office:annotation>` element to influence how it is displayed: `svg:x`, `svg:y`, `svg:width`, `svg:height`, `draw:caption-point-x`, `draw:caption-point-y`, `draw:corner-radius`, `table:end-cell-address`, `table:end-x`, `table:end-y`, `text:anchor-type`, `text:anchor-page-number`, `draw:layer`, `draw:style-name`, `draw:text-style-name`, `draw:transform`, `draw:name`, `draw:z-index` and `draw:id`. Their meaning is the same as if they are applied to a `<draw:caption>` element (see section 9.2.10). The use of these attributes is optional.

12.1.1 Creator

The optional `<dc:creator>` element described in section 3.1.7 specifies the author of the annotation.

12.1.2 Creation Date and Time

The optional `<dc:date>` element described in section 3.1.9 specifies the creation date and time of the annotation.

12.1.3 Creation Date and Time String

If the application only has a date string and cannot parse this string, it may write the string into the `<meta:date-string>` element.

```
<define name="meta-date-string">
    <element name="meta:date-string">
        <ref name="string"/>
    </element>
</define>
```

12.2 Number Format

The OpenDocument 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

12.2.1 Prefix and Suffix

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

If the prefix and suffix do not contain alphanumeric characters, an [XSLT] `format` attribute can be created from the OpenDocument attributes by concatenating the values of the `style:num-prefix`, `style:num-format`, and `style:num-suffix` attributes.

```
<define name="common-num-format-prefix-suffix-attlist" combine="interleave">
    <optional>
        <attribute name="style:num-prefix">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:num-suffix">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

12.2.2 Format Specification

The `style:num-format` attribute specifies the format of the number in the same way as the [XSLT] `format` attribute. 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,...

The value of this attribute can be "1", "a", "A", "i", or "I". For some elements, the attribute value also can be empty. In this case, no number is displayed.

```
<define name="common-num-format-attlist" combine="interleave">
    <choice>
        <attribute name="style:num-format">
            <choice>
                <value>1</value>
                <value>i</value>
                <value>I</value>
                <ref name="string"/>
                <empty/>
            </choice>
        </attribute>
        <group>
            <attribute name="style:num-format">
                <choice>
                    <value>a</value>
                    <value>A</value>
                </choice>
            </attribute>
            <ref name="style-num-letter-sync-attlist"/>
        </group>
        <empty/>
    </choice>
</define>
```

12.2.3 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:

- A new digit is inserted. Its start value is A, and it 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.
- A new digit is inserted 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**.

The `style:num-letter-sync` specifies whether letter synchronization shall take place.

```
<define name="style-num-letter-sync-attlist" combine="interleave">
    <optional>
        <attribute name="style:num-letter-sync">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

12.3 Change Tracking Metadata

Meta-data for change tracking is contained inside an `<office:change-info>` element. It contains the author and creation date of a tracked change, as well as an optional comment.

```
<define name="office-change-info">
    <element name="office:change-info">
        <ref name="dc-creator"/>
        <ref name="dc-date"/>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
```

Creator

The `<dc:creator>` element as described in section 3.1.7 specifies the name of the author who changed the document.

Date and Time

The `<dc:date>` element as described in section 3.1.9 specifies the date and time when the change took place.

Comment

An additional comment may be included as `<text:p>` elements.

12.4 Event Listener Tables

Many objects such as controls, images, text boxes, or an entire document support events. An event binds the occurrence of a particular condition to an action that is executed if the condition arises. For example, if a user places the cursor over a graphic, this condition triggers an action that is supported by the office application. This event, called "on-mouse-over", can be associated with a macro that is executed whenever the condition occurs, that is, whenever a user places the cursor over a graphic.

The XML representation of events and event tables is structured as follows:

- All of the event elements that are associated with an object are located in a container element called `<office:event-listeners>`.
- Each event-to-action association is recorded in one `<script:event-listener>` element.
- Depending on the type of action that the event triggers, the following elements are used:
  - The `<script:event-listener>` element represents events that are bound to a macro or script.

- The `<presentation:event-listener>` element represents events that are bound to an action that is specific to a presentation, for example, go to the next page. Presentation events are described in section .

The `<office:event-listeners>` element specifies the table of events that are associated with an object.

```
<define name="office-event-listeners">
    <element name="office:event-listeners">
        <zeroOrMore>
            <choice>
                <ref name="script-event-listener"/>
                <ref name="presentation-event-listener"/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

12.4.1 Event Listener

The `<script:event-listener>` element binds an event to a macro.

```
<define name="script-event-listener" combine="interleave">
    <element name="script:event-listener">
        <ref name="script-event-listener-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<script:event-listener>` element are:

- Event name
- Script language
- Macro Name and Location

Event Name

The `script:event-name` attribute specifies the name of the event. Since the available events, their names and their meanings are application and script language dependent, the name should be preceded by a namespace prefix, so that the corresponding namespace together with the event name can be used to identify the semantic of the event.

Where appropriate, it is recommended to use the event names described in [DOMEvents2]. The corresponding namespace is "http://www.w3.org/2001/xml-events".

Note: Event names defined in [DOMEvents2] are not namespaced. If used in OpenDocument, they should be preceded by a namespace prefix as described above. [DOMEvents3], which is a work in progress, specifies namespaced event names. After completion of this specification, it is recommended to use event names as specified in [DOMEvents3].

The following table describes events defined in [DOMEvents2] that are typically supported by office application and have an equivalent event in HTML. The namespace that should be used for these events is “http://www.w3.org/2001/xml-events”. The namespace prefix used in this specification is “dom”.

Value of script:event-name Attribute	***Equivalent HTML Event***	***Description of Event***
`dom:change`	`onchange`	Occurs when a control is no longer focused and the value of the control was modified since it was given focus.
`dom:DOMFocusIn`	`onfocus`	Occurs when a control is given focus using the mouse or the TAB key.
`dom:DOMFocusOut`	`onblur`	Occurs when a control is no longer focused as a result of moving the mouse or by tabbing navigation. It may be used with the same elements as form:on-focus.
`dom:mouseover`	`onmouseover`	Occurs when the mouse pointer is moved over the control.
`dom:mousemove`	`onmousemove`	Occurs when the mouse pointer is moved onto a control.
`dom:mousedown`	`onmousedown`	Occurs when a mouse button is pressed on a control.
`dom:mouseup`	`onmouseup`	Occurs when a mouse button is released on a control.
`on-mouseout`	`onmouseout`	Occurs when the mouse pointer is moved away from a control.
`dom:reset`	`onreset`	Occurs when a form is reset.
`dom:submit`	`onsubmit`	Occurs when a form is submitted.

12.4.2 Event Types

In addition to the HTML event types, the XML file format for office applications allows additional events to be handled at run time.

```
<define name="script-event-listener-attlist" combine="interleave">
    <attribute name="script:event-name">
        <ref name="string"/>
    </attribute>
</define>
```

Script Language

The `script:language` attribute specifies the scripting language in which the macro or script which is associated with the event is written. See also section 2.5.1.

```
<define name="script-event-listener-attlist" combine="interleave">
    <attribute name="script:language">
        <ref name="string"/>
    </attribute>
</define>
```

Macro Name and Location

The macro code that should be called for the event can be either specified by an IRI in [XLink] notation, or a simple name specified by a `script:macro-name` attribute. If an XLink is used, the IRI may have an arbitrary protocol, for instance one that encodes the name of a macro library name together with macro name defined in this library. Both, the XLink IRI as well as a simple name, are script language dependent.

```
<define name="script-event-listener-attlist" combine="interleave">
    <choice>
        <attribute name="script:macro-name">
            <ref name="string"/>
        </attribute>
        <group>
            <attribute name="xlink:href">
                <ref name="anyURI"/>
            </attribute>
            <optional>
                <attribute name="xlink:type" a:defaultValue="simple">
                    <value>simple</value>
                </attribute>
            </optional>
            <optional>
                <attribute name="xlink:actuate" a:defaultValue="onRequest">
                    <value>onRequest</value>
                </attribute>
            </optional>
        </group>
    </choice>
</define>
```

12.5 Mathematical Content

Mathematical content is represented by MathML 2.0 (see [MathML])

```
<define name="math-math">
    <element name="math:math">
        <ref name="mathMarkup"/>
    </element>
</define>

<!-- To avoid inclusion of the complete MathML schema, anything -->
<!-- is allowed within a math:math top-level element           -->
<define name="mathMarkup">
    <zeroOrMore>
        <choice>
            <attribute>
                <anyName/>
            </attribute>
            <text/>
            <element>
                <anyName/>
                <ref name="mathMarkup"/>
            </element>
        </choice>
    </zeroOrMore>
</define>
```

12.6 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.

All elements making use of DDE connections must contain their content (or its presentation), so that documents using DDE can still be properly displayed on machines which do not support the DDE mechanism, or where the DDE target is not available. Applications should preserve the DDE connection information even if they cannot make use of it, so that other applications can make use the DDE facilities.

DDE only is available on some operating systems. In order to create portable documents, authors are advised to use this feature in their documents with great care.

12.6.1 Container for DDE Connection Declarations

Within text and spreadsheet documents, DDE connection declarations are contained in one declaration element. For text documents, the element is `<text:dde-connection-decls>` as described in section 4.8. For spreadsheet documents, it is `<table:dde-links>` as described in section 8.10.

12.6.2 Declaring DDE Connections for Text Fields

Every DDE connection used by a text field 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.

```
<define name="text-dde-connection-decl">
    <element name="text:dde-connection-decl">
        <ref name="text-dde-connection-decl-attlist"/>
        <ref name="common-dde-connection-decl-attlist"/>
    </element>
</define>
```

The attributes that may be associated with the `<text:dde-connection-decl>` element are:

- Connection name
- DDE target application
- DDE target topic
- DDE target item
- Automatic update flag

Connection Name

The `office:name` attribute specifies the name by which the connection will be referred.

```
<define name="text-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:name">
        <ref name="string"/>
    </attribute>
</define>
```

Target Application

The `office:dde-application` attribute specifies the name of the target application to use for the DDE connection.

```
<define name="common-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-application">
        <ref name="string"/>
    </attribute>
</define>
```

> **Example**: The target name for the OpenOffice.org software is `soffice`. Therefore, internal DDE links have the attribute `text:dde-application="soffice"`.

Target Topic

The `office:dde-topic` attribute specifies the name of the topic to use for the DDE connection.

```
<define name="common-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-topic">
        <ref name="string"/>
    </attribute>
</define>
```

> **Example**: The OpenOffice.org software interprets the DDE topic as the name of the file.

Target Item

The `office:dde-item` attribute specifies which information the target application should deliver.

```
<define name="common-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-item">
        <ref name="string"/>
    </attribute>
</define>
```

> **Example**: If the target application for the DDE connection is the OpenOffice.org Writer software, the item represents the name of a bookmark. OpenOffice.org delivers the current text content to the requesting application.

Automatic Update

Office applications by default automatically update DDE links. If a manual update of the link is preferred, the `text:automatic-update` attribute my be used to specify that the DDE connection links should only be updated at the request of the user.

If the value of this attribute is `true`, then the application is expected to automatically update the DDE links. If this value of this attribute is `false`, the DDE links are updated on user request only.

```
<define name="common-dde-connection-decl-attlist" combine="interleave">
    <optional>
        <attribute name="office:automatic-update" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
```

```
</define>
```

12.6.3 Declaring DDE Connections for Tables

The DDE connection data of tables is contained in an `<office:dde-source>` element. The usage of this element differs between spreadsheet and text document tables. For text document tables, the element is contained within the table's `<table:table>` element directly. For spreadsheet documents, it is contained in a `<table:dde-link>` element, that describes a single DDE connection.

The `<table:dde-link>` element contains the DDE source data in the `<office:dde-source>` element and a simple table element that might be used to cache the data of the DDE source. The table does not need a name and does not contain style information. Only the data contained in the cell attributes is used. The cells themselves remain empty.

```
<define name="table-dde-link">
    <element name="table:dde-link">
        <ref name="office-dde-source"/>
        <ref name="table-table"/>
    </element>
</define>
```

The `<office:dde-source>` element supports `office:dde-application`, `office:dde-topic`, `office:dde-item` and `office:automatic-update` attributes as described in section 12.6.2. In addition to this, it supports the following attributes

- Connection name
- Conversion mode

```
<define name="office-dde-source">
    <element name="office:dde-source">
        <ref name="office-dde-source-attlist"/>
        <ref name="common-dde-connection-decl-attlist"/>
    </element>
</define>
```

Connection Name

The `office:name` attribute specifies the name by which the connection can be referred.

```
<define name="office-dde-source-attlist" combine="interleave">
    <optional>
        <attribute name="office:name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Conversion Mode

The `office:conversion-mode` attribute specifies the method by which the DDE server converts its data into numbers. There are three possible values:

- `into-default-style-data-style`: Numbers are converted into the data style which is set on the default style.
- `into-english-number`: numbers are converted into the English default format.

- `keep-text`: Numbers are not converted. They are treated as text.

```
<define name="office-dde-source-attlist" combine="interleave">
    <optional>
        <attribute name="office:conversion-mode"
                    a:defaultValue="into-default-style-data-style">
            <choice>
                <value>into-default-style-data-style</value>
                <value>into-english-number</value>
                <value>keep-text</value>
            </choice>
        </attribute>
    </optional>
</define>
```

13 SMIL Animations

This section describes [SMIL20] based elements and attribute that can be used within the OpenDocument format for animation effects.

13.1 Basic Animation Elements

The basic animation elements are directly derived from basic animation elements specified §3.5 and §12.5 of [SMIL20], and in section §19.2 of [SVG].

13.1.1 Animate

The `<anim:animate>` element behaves like the [SMIL20] `<smil:animate>` element. See §3.5.1 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:animate">
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-named-target-attlist"/>
        <ref name="common-anim-values-attlist"/>
        <ref name="common-anim-spline-mode-attlist"/>
        <ref name="common-spline-anim-value-attlist"/>
        <ref name="common-timing-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
    </element>
</define>
```

13.1.2 Set

The `<anim:set>` element behaves like the [SMIL20] `<smil:set>` element. See §3.5.2 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:set">
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-named-target-attlist"/>
        <ref name="common-anim-set-values-attlist"/>
        <ref name="common-timing-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
    </element>
</define>
```

13.1.3 Animate Motion

The `<anim:animateMotion>` element behaves as the [SVG] `<svg:animateMotion>` element. See §19.2.12 of [SVG] and §3.5.3 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:animateMotion">
        <ref name="anim-animate-motion-attlist"/>
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-named-target-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
        <ref name="common-anim-values-attlist"/>
        <ref name="common-timing-attlist"/>
        <ref name="common-spline-anim-value-attlist"/>
    </element>
</define>
```

The Motion Path

The [SVG] `svg:path` attribute can be used to specify a path along which the element is animated. See §19.2.12 of [SVG] for details.

```
<define name="anim-animate-motion-attlist" combine="interleave">
    <optional>
        <attribute name="svg:path">
            <ref name="pathData"/>
        </attribute>
    </optional>
</define>
```

Origin

The [SVG] `svg:origin` attribute can be used to specify an origin. See §19.2.12 of [SVG] for details.

```
<define name="anim-animate-motion-attlist" combine="interleave">
    <optional>
        <attribute name="svg:origin">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Calc Mode

The [SMIL20] `smil:calcMode` attribute is used to specify the interpolation mode of the animation. See §19.2.12 of [SVG] for details.

```
<define name="anim-animate-motion-attlist" combine="interleave">
    <optional>
        <attribute name="smil:calcMode" a:defaultValue="paced">
            <choice>
                <value>discrete</value>
                <value>linear</value>
                <value>paced</value>
                <value>spline</value>
            </choice>
        </attribute>
    </optional>
</define>
```

13.1.4 Animate Color

The `<anim:animateColor>` element behaves like the [SMIL20] `<smil:animateColor>` element. See §3.5.4 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:animateColor">
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-named-target-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
        <ref name="common-anim-values-attlist"/>
        <ref name="common-anim-spline-mode-attlist"/>
        <ref name="common-spline-anim-value-attlist"/>
        <ref name="anim-animate-color-attlist"/>
        <ref name="common-timing-attlist"/>
    </element>
</define>
```

Color Interpolation

The `anim:color-interpolation` attribute specifies the color space that is used for color interpolation.

```
<define name="anim-animate-color-attlist" combine="interleave">
    <optional>
        <attribute name="anim:color-interpolation" a:defaultValue="rgb">
            <choice>
                <value>rgb</value>
                <value>hsl</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Color Interpolation Direction

The `anim:color-interpolation-direction` attribute specify the direction that is used for color interpolation. This is only valid for the HSL color space.

```
<define name="anim-animate-color-attlist" combine="interleave">
    <optional>
        <attribute name="anim:color-interpolation-direction"
                                a:defaultValue="clockwise">
            <choice>
                <value>clockwise</value>
                <value>counter-clockwise</value>
            </choice>
        </attribute>
    </optional>
</define>
```

13.1.5 Animate Transform

The `<anim:animateTransform>` element is based on the [SVG] `<svg:animateTransform>` element. See §19.2.14 of [SVG] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:animateTransform">
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-named-target-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
        <ref name="common-anim-values-attlist"/>
        <ref name="anim-animate-transform-attlist"/>
        <ref name="common-timing-attlist"/>
    </element>
</define>
```

Transformation Type

The [SVG] `svg:type` attribute is used to specify the transformation type. See §19.2.14 of [SVG] for details.

```
<define name="anim-animate-transform-attlist" combine="interleave">
    <attribute name="svg:type">
        <choice>
            <value>translate</value>
            <value>scale</value>
            <value>rotate</value>
            <value>skewX</value>
            <value>skewY</value>
        </choice>
    </attribute>
</define>
```

13.1.6 Transition Filter

The `<anim:transitionFilter>` element is based on the [SMIL20] `<smil:transitionFilter>` element. See §12.5.1 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:transitionFilter">
        <ref name="common-anim-target-attlist"/>
        <ref name="common-anim-add-accum-attlist"/>
        <ref name="common-anim-values-attlist"/>
        <ref name="common-anim-spline-mode-attlist "/>
        <ref name="anim-transition-filter-attlist"/>
        <ref name="common-timing-attlist"/>
    </element>
</define>
```

Transition Type

The [SMIL20] `smil:type` attribute is used to specify the transition type or family. See §12.8 of [SMIL20] for a list of supported types.

```
<define name="anim-transition-filter-attlist" combine="interleave">
    <attribute name="smil:type">
        <ref name="string"/>
    </attribute>
</define>
```

Transition Subtype

The [SMIL20] `smil:subtype` attribute can be used to specify the transition subtype. See §12.8 of [SMIL20] for a list of supported subtypes.

```
<define name="anim-transition-filter-attlist" combine="interleave">
    <optional>
        <attribute name="smil:subtype">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Transition Direction

The [SMIL20] `smil:direction` attribute can be used to specify the transition direction. See §12.4.1 of [SMIL20] for details.

```
<define name="anim-transition-filter-attlist" combine="interleave">
    <optional>
        <attribute name="smil:direction" a:defaultValue="forward">
            <choice>
                <value>forward</value>
                <value>reverse</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Fade Color

The [SMIL20] `smil:fadeColor` attribute can be used to specify the transition fade color for transitions that makes use of a start or end color. See §12.5.1 of [SMIL20] for details.

```
<define name="anim-transition-filter-attlist" combine="interleave">
    <optional>
        <attribute name="smil:fadeColor">
            <choice>
                <value>forward</value>
                <value>reverse</value>
            </choice>
        </attribute>
    </optional>
</define>
```

The Transition Mode

The [SMIL20] `smil:mode` attribute is used to specify if the animated element will be transition in or out. See §12.5.1 of [SMIL20] for details.

```
<define name="anim-transition-filter-attlist" combine="interleave">
    <optional>
        <attribute name="smil:mode" a:defaultValue="in">
            <choice>
                <value>in</value>
                <value>out</value>
            </choice>
        </attribute>
    </optional>
</define>
```

13.2 Animation Model Attributes

The animation model uses the same concepts and syntax as specified in §3 of [SMIL20].

13.3 Common Animation Attributes

Element Id

The `anim:id` attribute defines an ID that is used to identify the element inside a document.

```
<define name="common-anim-attlist" combine="interleave">
    <optional>
        <attribute name="anim:id">
            <ref name="ID"/>
        </attribute>
    </optional>
</define>
```

13.3.1 Animation Target Attributes

Target Element

The [SMIL20] `smil:targetElement` attribute is used to specify the target element to be animated. See §3.4.1 of [SMIL20] for details. See section 9.8.2 for details about the usage of this attribute in presentation documents.

```
<define name="common-anim-target-attlist" combine="interleave">
    <optional>
        <attribute name="smil:targetElement">
            <ref name="IDREF"/>
        </attribute>
    </optional>
</define>
```

Target Attribute

The [SMIL20] `smil:attributeName` attribute is used to specify a target attribute by name. See §3.4.1 of [SMIL20] for details. See section 9.8.2 for details about the usage of this attribute in presentation documents.

```
<define name="common-anim-named-target-attlist" combine="interleave">
    <attribute name="smil:attributeName">
        <ref name="string"/>
    </attribute>
</define>
```

Target Element Sub Item

The `anim:sub-item` attribute specifies an optional sub item of the target element. Possible values for this element depend on the document type and the target element type. See section 9.8.2 for details about the usage of this attribute in presentation documents.

```
<define name="common-anim-target-attlist" combine="interleave">
    <optional>
        <attribute name="anim:sub-item">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

13.3.2 Animation Function Attributes

Value List

The [SMIL20] `smil:values` attribute specifies the values used to animate the target element. See $3.4.2 of [SMIL20] for details.

```
<define name="common-anim-values-attlist" combine="interleave">
    <optional>
        <attribute name="smil:values">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Calc Mode

The [SMIL20] `smil:calcMode` attribute is used to specify the interpolation mode of the animation function. See $3.4.2 of [SMIL20] for details.

```
<define name="common-anim-spline-mode-attlist" combine="interleave">
    <optional>
        <attribute name="smil:calcMode" a:defaultValue="discrete">
            <choice>
                <value>discrete</value>
                <value>linear</value>
                <value>paced</value>
                <value>spline</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Key Times

The [SMIL20] `smil:keyTimes` attribute specifies the pacing of the animation. See $3.7.1 of [SMIL20] for details.

```
<define name="common-spline-anim-value-attlist" combine="interleave">
    <optional>
        <attribute name="smil:keyTimes">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Key Splines

The [SMIL20] `smil:keySplines` attribute specifies a cubic Bézier function that controls interval pacing. See $3.7.1 of [SMIL20] for details.

```
<define name="common-spline-anim-value-attlist" combine="interleave">
    <optional>
        <attribute name="smil:keySplines">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Accumulation

The [SMIL20] `smil:accumulate` attribute specifies the accumulation of the animation function. See $3.4.3 of [SMIL20] for details.

```
<define name="common-anim-add-accum-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="smil:accumulate">
            <choice>
                <value>none</value>
                <value>sum</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Additive

The [SMIL20] `smil:additive` attribute specifies if the additive of the animation function. See $3.4.3 of [SMIL20] for details.

```
<define name="common-anim-add-accum-attlist" combine="interleave">
    <optional>
        <attribute name="smil:additive">
            <choice>
                <value>replace</value>
                <value>sum</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Formula

The `anim:formula` attribute specifies a formula that is used as the animation function. The identifier '$' will be replaced by a value between 0 and 1 (inclusive) that represents the proportional offset into the animation element's duration. For specific document types, additional identifiers may exist. The following is the minimum supported grammar:

```
identifier = '$' | 'pi'

function = 'abs'|'sqrt'|'sin'|'cos'|'tan'|'atan'|'acos'|'asin'|'exp'|'log'
binary_function = 'min'|'max'

basic_expression =
    number |
    identifier |
    function '(' additive_expression ')' |
    binary_function
        '(' additive_expression ',' additive_expression ')' |
    '(' additive_expression ')'

unary_expression =
    '-' basic_expression |
    basic_expression

multiplicative_expression =
    unary_expression
    (   ( '*' unary_expression )* |
        ( '/' unary_expression )* )

additive_expression =
    multiplicative_expression
    (   ( '+' multiplicative_expression )* |
        ( '-' multiplicative_expression )* )
```

See section 9.8.2 for details about additional identifiers for presentation documents.

If a anim:formula attribute is given, it overrides the smil:values, smil:to, smil:from and smil:by attributes as specified in the next section.

```
<define name="common-anim-values-attlist" combine="interleave">
    <optional>
        <attribute name="anim:formula">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Simple Animation Functions

In addition to describing an animation with a list of values, a simplified version using the [SMIL20] smil:from, smil:to and smil:by attributes can be used. See §3.4.4 of [SMIL20] for details.

```
<define name="common-anim-set-values-attlist" combine="interleave">
    <optional>
        <attribute name="smil:to">
            <ref name="string"/>
        </attribute>
    </optional>
</define>

<define name="common-anim-values-attlist" combine="interleave">
    <ref name="common-anim-set-values-attlist"/>
    <optional>
        <attribute name="smil:from">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="smil:by">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

13.4 Animation Timing

The animation timing uses the same concepts and syntax as specified in §10 and §11 of [SMIL20] chapters.

13.4.1 Animation Timing Attributes

Element Start

The [SMIL20] smil:begin attribute can be used to specify the begin time of an element. See §10.3.1 of [SMIL20] for details.

```
<define name="common-begin-end-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:begin">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Element End

The [SMIL20] `smil:end` attribute can be used to specify the end time of an element. See §10.3.1 of [SMIL20] for details.

```
<define name="common-begin-end-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:end">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Element Duration

The [SMIL20] `smil:dur` attribute can be used to specify the duration of an element. See §10.3.1 of [SMIL20] for details.

```
<define name="common-dur-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:dur">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Element End Synchronization

The [SMIL20] `smil:endsync` attribute can be used to control the implicit duration of time containers, as a function of their children. See §10.3.1 of [SMIL20] for details.

```
<define name="common-endsync-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:endsync">
            <choice>
                <value>first</value>
                <value>last</value>
                <value>all</value>
                <value>media</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Repeating Elements

The [SMIL20] `smil:repeatCount` and `smil:repeatDur` attributes specifies the behavior of repeated animations. See §10.3.1 of [SMIL20] for details.

```
<define name="common-repeat-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:repeatDur">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="smil:repeatCount">
            <choice>
                <ref name="nonNegativeInteger"/>
                <value>indefinite</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Fill

The [SMIL20] `smil:fill` attribute specifies the behavior of an element after an animation is finished. See §10.3.1 of [SMIL20] for details.

```
<define name="common-fill-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:fill">
            <choice>
                <value>remove</value>
                <value>freeze</value>
                <value>hold</value>
                <value>auto</value>
                <value>default</value>
                <value>transition</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Fill Default

The [SMIL20] `smil:fillDefault` attribute specifies the default behavior for the `smil:fill` attribute. See §10.3.1 of [SMIL20] for details.

```
<define name="common-fill-default-attlist" combine="interleave">
    <optional>
        <attribute name="smil:fillDefault">
            <choice>
                <value>remove</value>
                <value>freeze</value>
                <value>hold</value>
                <value>transition</value>
                <value>auto</value>
                <value>inherit</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Restart

The [SMIL20] `smil:restart` attribute can be used to specify the restart behavior of an element. See §10.3.1 of [SMIL20] for details.

```
<define name="common-restart-timing-attlist" combine="interleave">
    <optional>
        <attribute name="smil:restart" a:defaultValue="default">
            <choice>
                <value>never</value>
                <value>always</value>
                <value>whenNotActive</value>
                <value>default</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Restart Default

The [SMIL20] `smil:restartDefault` attribute can be used to specify the default restart behavior of an element. See §10.3.1 of [SMIL20] for details.

```
<define name="common-restart-default-attlist" combine="interleave">
    <optional>
        <attribute name="smil:restartDefault" a:defaultValue="inherit">
            <choice>
                <value>never</value>
                <value>always</value>
                <value>whenNotActive</value>
                <value>inherit</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Accelerate

The [SMIL20] `smil:accelerate` attribute can be used to specify a simple acceleration of element time. See §11.1.2 of [SMIL20] for details.

```
<define name="common-time-manip-attlist" combine="interleave">
    <optional>
        <attribute name="smil:accelerate" a:defaultValue="0.0">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Decelerate

The [SMIL20] `smil:decelerate` attribute can be used to specify a simple deceleration of element time. See §11.1.2 of [SMIL20] for details.

```
<define name="common-time-manip-attlist" combine="interleave">
    <optional>
        <attribute name="smil:decelerate" a:defaultValue="0.0">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Auto Reverse

The [SMIL20] `smil:autoreverse` attribute can be used to specify an automatic playback in reverse. See §11.1.2 of [SMIL20] for details.

```
<define name="common-time-manip-attlist" combine="interleave">
    <optional>
        <attribute name="smil:autoReverse" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

13.4.2 Parallel Animations

The `<anim:par>` element is based on the [SMIL20] `<smil:par>` element and defines a parallel time container. See §10.3.2 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:par">
        <ref name="common-anim-attlist"/>
        <ref name="common-timing-attlist"/>
        <ref name="common-endsync-timing-attlist"/>
        <zeroOrMore>
            <ref name="animation-element"/>
        </zeroOrMore>
    </element>
</define>

<define name="common-basic-timing-attlist" combine="interleave">
   <ref name="common-begin-end-timing-attlist"/>
   <ref name="common-dur-timing-attlist"/>
   <ref name="common-repeat-timing-attlist"/>
</define>

<define name="common-timing-attlist" combine="interleave">
   <ref name="common-basic-timing-attlist"/>
   <ref name="common-restart-timing-attlist"/>
   <ref name="common-restart-default-attlist"/>
   <ref name="common-fill-timing-attlist"/>
   <ref name="common-fill-default-attlist"/>
   <ref name="common-time-manip-attlist"/>
</define>
```

13.4.3 Sequential Animations

The `<anim:seq>` element is based on the [SMIL20] `<smil:seq>` element and defines a sequential time container. See §10.3.2 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:seq">
        <ref name="common-anim-attlist"/>
        <ref name="common-endsync-timing-attlist"/>
        <ref name="common-timing-attlist"/>
        <zeroOrMore>
            <ref name="animation-element"/>
        </zeroOrMore>
    </element>
</define>
```

13.4.4 Iterative Animations

The `<anim:iterate>` element defines a parallel time container. The difference to a `<anim:par>` element is that the `<anim:iterate>` element does not specify effects for its target element itself. Instead of this, it iterates over possible child elements of the target element and executes all its child effects with the children of the target element as target.

```
<define name="animation-element" combine="choice">
    <element name="anim:iterate">
        <ref name="common-anim-attlist"/>
        <ref name="anim-iterate-attlist"/>
        <ref name="common-timing-attlist"/>
        <ref name="common-endsync-timing-attlist"/>
        <zeroOrMore>
            <ref name="animation-element"/>
        </zeroOrMore>
    </element>
</define>
```

The Target Element

The [SMIL20] `smil:targetElement` and `anim:sub-item` attributes specify the target element to whose children the effects should be applied. See section 9.8.2 for details about the attribute's usage in presentation documents.

```
<define name="anim-iterate-attlist" combine="interleave">
    <ref name="common-anim-target-attlist"/>
</define>
```

The Iterate Type

The `anim:iterate-type` attribute specifies how the iteration targets child elements are iterated. Possible values depends on the document type and the target element type. See section 9.8.2 for details about the attribute's usage in presentation documents.

```
<define name="anim-iterate-attlist" combine="interleave">
    <optional>
        <attribute name="anim:iterate-type">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

The Iterate Interval

The `anim:iterate-interval` attribute specifies the delay between the execution of the child effects of this element. The effects of the next iterated child of the target element are started when the given time is elapsed since the effects for the previous child has been started. An iterate interval of zero seconds would have the same behavior as using a `<anim:par>` element.

```
<define name="anim-iterate-attlist" combine="interleave">
    <optional>
        <attribute name="anim:iterate-interval">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

13.5 Media Elements

13.5.1 Audio

The `<anim:audio>` element is based on the [SMIL20] `<smil:audio>` element. It allows the playback of audio streams during an animation. See §7.3.1 of [SMIL20] for details.

```
<define name="animation-element" combine="choice">
    <element name="anim:audio">
        <ref name="common-anim-attlist"/>
        <ref name="anim-audio-attlist"/>
        <ref name="common-basic-timing-attlist"/>
    </element>
</define>
```

Source

The `xlink:href` attribute specifies the IRI of the audio stream.

```
 <define name="anim-audio-attlist" combine="interleave">
    <optional>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
    </optional>
</define>
```

Audio Level

The `anim:audio-level` attribute specifies the volume during playback. Its value is a number in the range 0 (inaudible) to 1 (the system volume).

```
<define name="anim-audio-attlist" combine="interleave">
    <optional>
        <attribute name="anim:audio-level">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

13.6 Special Elements

13.6.1 Command

The `<anim:command>` element is used to send generic commands to the application during an animation. The available command types and its parameters depend on the document type and the type of the target element. See section 9.8.2 for details about the element's usage in presentation documents.

```
<define name="animation-element" combine="choice">
    <element name="anim:command">
        <ref name="common-anim-attlist"/>
        <ref name="anim-command-attlist"/>
        <ref name="common-begin-end-timing-attlist"/>
        <ref name="common-anim-target-attlist"/>
        <zeroOrMore>
            <element name="anim:param">
                <attribute name="anim:name"/>
                <attribute name="anim:value"/>
            </element>
        </zeroOrMore>
    </element>
</define>
```

Command

The `anim:command` attribute specifies the command that will be executed at the application when this animation element is started.

```
<define name="anim-command-attlist" combine="interleave">
```

```
    <attribute name="anim:command">
        <ref name="string"/>
    </attribute>
</define>
```

14 Styles

Many objects in an office 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 the OpenDocument format, formatting properties are only stored within styles. This differs to the user interface of typical office applications, where formatting properties may be assigned to an object directly, or indirectly by applying 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 OpenDocument 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. The format and layout of the document get separated 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.

Within this chapter, the various style types are explained.

14.1 Style Element

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 graphic styles.

The individual style families that make use of these element are described separately. Within this section, the common attributes of the style element are described.

```
<define name="style-style">
    <element name="style:style">
        <ref name="style-style-attlist"/>
        <ref name="style-style-content"/>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>
```

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

- Style name
- Display name

- Style family
- Parent style
- Next style
- List style
- Master page name
- Automatically update
- Data style name
- Class
- Outline numbering level

Style Name

The `style:name` attribute identifies the name of the style. This attribute, combined with the `style:family` attribute, uniquely identifies a style. The `<office:styles>`, `<office:automatic-styles>` and `<office:master-styles>` elements each must not contain 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, it cannot be assumed that the same name is generated each time.

In an XML document, the name of each style is a unique name that may be independent of the language selected for an office applications user interface. Usually these names are the ones used for the English version of the user interface.

```
<define name="style-style-attlist" combine="interleave">
    <attribute name="style:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `style:display-name` attribute specifies the name of the style as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Style Family

The `style:family` attribute identifies the family of the style, for example, paragraph, text, or frame. It might have one of the following values: `paragraph`, `text`, `section`, `table`, `table-column`, `table-row`, `table-cell`, `table-page`, `chart`, `default`, `drawing-page`, `graphic`, `presentation`, `control` and `ruby`.

Parent Style

The style:parent-style-name attribute specifies the name of the parent style. 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 has to exist.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:parent-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Next Style

The style:next-style-name attribute specifies the style to used for the next paragraph if a paragraph break is inserted in the user interface. By default, the current style is used as the next style.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:next-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

List Style

A paragraph style and styles of other families that may contain paragraph properties (for instance graphic styles) can have an associated list style. This applies to automatic and common styles.

The list style specified by the style:list-style-name attribute is only applied to headings and 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.

The style:list-style-name attribute's value can be empty. In this case, an association with a list style that is inherited from a parent style will be removed.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:list-style-name">
            <choice>
                <ref name="styleName"/>
                <empty/>
            </choice>
        </attribute>
    </optional>
</define>
```

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.

This attribute is ignored if it is associated with a paragraph style that is applied to a paragraph within a table.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:master-page-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

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, there might be a paragraph style that contains a formatting property specifying that paragraph text is centered, and this paragraph style is applied to a paragraph. If the user manually changes the formatting of that paragraph text to be right-aligned and the value of the `style:auto-update` attribute is `true`, then 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. This attribute can have a value of `true` or `false`.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:auto-update" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Data Style Name

Table cell style can have an associated data style. This applies to automatic and common styles. The data style is referenced by the `style:data-style-name` attribute. See section 14.7 for details about data styles.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:data-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Class

A style may belong to an arbitrary class of styles. The class is an arbitrary string. The class has no meaning within the file format itself, but it can for instance be evaluated by user interfaces to show a list of styles where the styles are grouped by its name.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:class">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Outline Numbering Level

For style with family `paragraph`, the `style:default-outline-level` attribute specifies a default outline level. It takes a number like the `text:outline-level` attribute of the heading element `<text:h>`. If this attribute is existing for a paragraph style, and if the paragraph style is assigned to a paragraph by an user interface action, then office applications should convert the paragraph into a heading of the given level. However, the attribute has no effect to the differentiation of headings and paragraphs in the file format itself. The differentiation between headings and paragraphs still takes place by using either a `<text:h>` or a `<text:p>` element. If a `<text:p>` element references a paragraph style that has a `style:default-outline-level` attribute, the paragraph remains a paragraph and will not become a heading.

```
<define name="style-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:default-outline-level">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Formatting Properties

If a style has formatting attributes assigned, the style element contains one ore more formatting property container elements. See section 15 for detailed information about these element.

Sample Style

Example: OpenDocument representation of the “Text body” paragraph style

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

14.1.1 Style Mappings

The `<style:map>` element specifies the mapping to another style, if certain conditions exist. If a style contains such mappings, it is called an conditional style. There is one element for every condition that the style uses.

Conditional styles usually are supported by paragraph styles contained in text documents and table cell styles contained in spreadsheets only. Conditional styles are also supported by data styles.

```
<define name="style-map">
    <element name="style:map">
        <ref name="style-map-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be 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.

The value of this attribute is a Boolean expression. The syntax of the expression is similar to the XPath syntax. If an application detects a condition that it does not recognize, it must ignore the entire `<style:map>` element.

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()`

The following conditions are valid for paragraph styles:

- `is-true-formula(`*`formula`*`)`
- `cell-content-is-between(`*`value`*`, `*`value`*`)`
- `cell-content-is-not-between(`*`value`*`, `*`value`*`)`
- `cell-content()` *`operator value`*, where *`operator`* is one of; `'<'`, `'>'`, `'<='`, `'>='`, `'='` or `'!='`, and value is a *`numberValue`*, a *`string`* or a *`formula`*.
- A *`numberValue`* is a whole or decimal number. The number cannot contain comma separators for numbers of 1000 or greater.
- A *`string`* comprises one or more characters surrounded by quotation marks.
- A *`formula`* is a formula (see 8.1.3) without the equals (=) sign at the beginning.

The following conditions are valid for data styles:

- `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`.

The conditions that apply for different types of styles may differ.

```
<define name="style-map-attlist" combine="interleave">
    <attribute name="style:condition">
        <ref name="string"/>
    </attribute>
</define>
```

Applied Style

The `style:apply-style-name` attribute specifies the style to apply when the condition specified by the `style:condition` attribute is `true`. If the referenced style is undefined or is an automatic style, an error occurs.

```
<define name="style-map-attlist" combine="interleave">
    <attribute name="style:apply-style-name">
```

```
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

Base Cell Address

For table cell styles, the `style:base-cell-address` attribute specifies the base cell for relative addresses in formulas. 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.

```
<define name="style-map-attlist" combine="interleave">
    <optional>
        <attribute name="style:base-cell-address">
            <ref name="cellAddress"/>
        </attribute>
    </optional>
</define>
```

Example: Style mapping

```
<style:style style:name="Text body" style:family="paragraph"
             style:parent-style-name="Standard"
             style:next-style-name="Text body">
    <style:paragraph-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>
```

14.2 Default Styles

A default style specifies default formatting properties for a certain style family. These defaults are used if a formatting property is neither specified by an automatic nor a common style. Default styles exist for all style families that are represented by the `<style:style>` element specified in section 14.1.

Default styles are represented by the `<style:default-style>` element. The only attribute supported by this element is `style:family`. Its meaning equals the one of the same attribute for the `<style:style>` element, and the same properties child elements are supported depending on the style family.

```
<define name="style-default-style">
    <element name="style:default-style">
        <ref name="style-style-content"/>
    </element>
</define>
```

14.3 Page Layout

The `<style:page-layout>` element specifies the physical properties of a page. This element contains a `<style:page-layout-properties>` element which specifies the formatting properties of the page and two optional elements that specify the properties of headers and footers.

```
<define name="style-page-layout">
```

```
    <element name="style:page-layout">
        <ref name="style-page-layout-attlist"/>
        <optional>
            <ref name="style-page-layout-properties"/>
        </optional>
        <optional>
            <ref name="style-header-style"/>
        </optional>
        <optional>
            <ref name="style-footer-style"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<style:page-layout>` element are:

- Name
- Page usage

Name

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

```
<define name="style-page-layout-attlist" combine="interleave">
    <attribute name="style:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Page Usage

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

```
<define name="style-page-layout-attlist" combine="interleave">
    <optional>
        <attribute name="style:page-usage" a:defaultValue="all">
            <choice>
                <value>all</value>
                <value>left</value>
                <value>right</value>
                <value>mirrored</value>
            </choice>
        </attribute>
    </optional>
</define>
```

14.3.1 Header and Footer Styles

The header and footer style elements `<style:header-style>` and `<style:footer-style>` specify the formatting properties for headers and footers on a page. These elements must be contained within a page layout element. The contain a `<style:header-footer-properties>` element that contains the formatting properties of the header or footer.

```
<define name="style-header-style">
    <element name="style:header-style">
        <optional>
            <ref name="style-header-footer-properties"/>
        </optional>
    </element>
</define>
<define name="style-footer-style">
    <element name="style:footer-style">
        <optional>
            <ref name="style-header-footer-properties"/>
        </optional>
    </element>
</define>
```

14.4 Master Pages

In text and spreadsheet documents, the `<style:master-page>` element contains the content of headers and footers. In these applications, a sequence of pages is generated by making use of a single master page or a set of master pages.

In drawing and presentation documents, the `<style:master-page>` element is used to define master pages as common backgrounds for **drawing pages**. Each drawing page here is directly linked to one master page, which is specified by the `draw:master-page-name` attribute of the drawing pages style.

Master pages are contained in the `<office:master-styles>` element. See also section 2.8.

All document must contain at least one master page element.

```
<define name="style-master-page">
    <element name="style:master-page">
        <ref name="style-master-page-attlist"/>
        <optional>
            <ref name="style-header"/>
            <optional>
                <ref name="style-header-left"/>
            </optional>
        </optional>
        <optional>
            <ref name="style-footer"/>
            <optional>
                <ref name="style-footer-left"/>
            </optional>
        </optional>
        <optional>
            <ref name="office-forms"/>
        </optional>
        <zeroOrMore>
            <ref name="style-style"/>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="shape"/>
        </zeroOrMore>
        <optional>
            <ref name="presentation-notes"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<style:master-page>` element are:

- Page name
- Display name
- Page layout

- Page style
- Next style name

The elements that my be included in the `<style:master-page>` element are:

- Headers and Footers
- Forms
- Styles
- Shapes
- Presentation notes

Page Name

The `style:name` attribute specifies the name of a master page. Each master page is referenced using the page name. This attribute is required and the name specified must be unique.

```
<define name="style-master-page-attlist" combine="interleave">
    <attribute name="style:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `style:display-name` attribute specifies the name of the master as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="style-master-page-attlist" combine="interleave">
    <optional>
        <attribute name="style:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Page Layout

The `style:page-layout-name` attribute specifies a page layout which contains the sizes, border and orientation of the master page. See section 14.3 for details on page layouts.

```
<define name="style-master-page-attlist" combine="interleave">
    <attribute name="style:page-layout-name">
        <ref name="styleNameRef"/>
    </attribute>
</define>
```

Page Style

In graphic applications, additional drawing page attributes my be assigned to a drawing page using the `draw:style-name` attribute. This attribute is optional. The fixed family for page styles is `drawing-page`. This is used to define an optional background filling.

```
<define name="style-master-page-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="draw:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Next Style Name

For text and spreadsheet documents, the `style:next-style-name` attribute identifies the master page that is used for the next page if the current page is entirely filled. This attribute is optional. If the next style name is not specified, the current master page is used for the next page. The value of this attribute must be the name of another `style:master-page` element.

```
<define name="style-master-page-attlist" combine="interleave">
    <optional>
        <attribute name="style:next-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

14.4.1 Headers and Footers

The header and footer elements specify the content of headers and footers. They are contained within a master page element. 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.

If the `style:page-usage` attribute associated with the page layout 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. Such headers and footers usually are supported by text documents.
- A sequence of any of the following elements; `<style:region-left>`, `<style:region-center>` and `<style:region-right>`. These elements usually are supported by spreadsheet documents.
- 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.

```
<define name="style-header">
    <element name="style:header">
        <ref name="common-style-header-footer-attlist"/>
        <ref name="header-footer-content"/>
    </element>
</define>
<define name="style-footer">
    <element name="style:footer">
        <ref name="common-style-header-footer-attlist"/>
        <ref name="header-footer-content"/>
    </element>
</define>
<define name="style-header-left">
    <element name="style:header-left">
        <ref name="common-style-header-footer-attlist"/>
        <ref name="header-footer-content"/>
    </element>
</define>
<define name="style-footer-left">
    <element name="style:footer-left">
        <ref name="common-style-header-footer-attlist"/>
        <ref name="header-footer-content"/>
    </element>
</define>
<define name="header-footer-content">
    <choice>
        <group>
            <ref name="text-tracked-changes"/>
            <ref name="text-decls"/>
            <zeroOrMore>
                <choice>
                    <ref name="text-h"/>
                    <ref name="text-p"/>
                    <ref name="text-list"/>
                    <ref name="table-table"/>
                    <ref name="text-section"/>
                    <ref name="text-table-of-content"/>
                    <ref name="text-illustration-index"/>
                    <ref name="text-table-index"/>
                    <ref name="text-object-index"/>
                    <ref name="text-user-index"/>
                    <ref name="text-alphabetical-index"/>
                    <ref name="text-bibliography"/>
                    <ref name="text-index-title"/>
                    <ref name="change-marks"/>
                </choice>
            </zeroOrMore>
        </group>
        <group>
            <optional>
                <ref name="style-region-left"/>
            </optional>
            <optional>
                <ref name="style-region-center"/>
            </optional>
            <optional>
                <ref name="style-region-right"/>
            </optional>
        </group>
    </choice>
</define>
```

Display

The `style:display` attribute specifies whether the header or footer is displayed or not.

```
<define name="common-style-header-footer-attlist" combine="interleave">
    <optional>
        <attribute name="style:display" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Regions

The region elements `<style:region-left>`, `<style:region-center>` and `<style:region-right>` specify three regions of a header or footer that are displayed left aligned, centered or right aligned. Each of these regions can contain a sequence of paragraphs.

```
<define name="style-region-left">
    <element name="style:region-left">
        <ref name="region-content"/>
    </element>
</define>
<define name="style-region-center">
    <element name="style:region-center">
        <ref name="region-content"/>
    </element>
</define>
<define name="style-region-right">
    <element name="style:region-right">
        <ref name="region-content"/>
    </element>
</define>

<define name="region-content">
    <zeroOrMore>
        <ref name="text-p"/>
    </zeroOrMore>
</define>
```

14.4.2 Presentation Notes

The `<presentation:notes>` element is usually supported only by presentation applications, where each master page as well as each drawing page in a presentation can have an additional presentation notes page. The presentation notes page contains:

- A preview of the drawing page.
- Additional graphic shapes as contained in the `<presentation:notes>` element. While the `<presentation:notes>` may contain any kind of shapes, the only shape type that should be supported by presentation applications are text boxes (i.e., `<draw:text-box>` contained in a `<draw:frame>`).

```
<define name="presentation-notes">
    <element name="presentation:notes">
        <ref name="common-presentation-header-footer-attlist"/>
        <ref name="presentation-notes-attlist"/>
        <ref name="office-forms"/>
        <zeroOrMore>
            <ref name="shape"/>
        </zeroOrMore>
    </element>
</define>
```

Page Layout

The `style:page-layout-name` attribute specifies a page layout which contains the sizes, border and orientation of the notes page. See section 14.3 for details on page layouts.

```
<define name="presentation-notes-attlist" combine="interleave">
    <optional>
        <attribute name="style:page-layout-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Page Style

The attribute `draw:style-name` assigns an additional formatting attributes to a notes page by assigning a drawing page style. This attribute is optional. The fixed family for page styles is `drawing-page`.

```
<define name="presentation-notes-attlist" combine="interleave">
    <optional>
        <attribute name="draw:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Header Declaration

The `presentation:use-header-name` attribute specifies the name of the header field declaration (see section 9.11.2) that is used for all header fields (see section 9.10.1) that are displayed on the notes page. See also section 9.1.4.

Footer Declaration

The `presentation:use-footer-name` attribute specifies the name of the footer field declaration (see section 9.11.3) that is used for all footer fields (see section 9.10.2) that are displayed on the notes page. See also section 9.1.4.

Date and Time Declaration

The `presentation:use-date-time-name` attribute specifies the name of the date-time field declaration (see section 9.11.4) that is used for all date-time fields (see section 9.10.3) that are displayed on the notes page. See also section 9.1.4.

Example: Master page containing presentation notes.

```
<office:master-styles>
    ...
    <style:master-page style:name="home" style:page-layout="default">
        <style:style style:name="title" style:family="presentation">
            <style:text-properties fo:font-style="italic"/>
        </style:style>
        <style:style style:name="subtitle" style:family="presentation"
                    style:parent-style-name="title">
            <style:text-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:master-styles>
```

14.5 Table Templates

A table template is a set formatting properties, like borders, background color, and text properties that can be applied to a table when creating it. In contrast to other styles, it is not referenced by a table, but if a table is created, a set of table-cell styles is created from the table template. To change the formatting properties of a table, the cell styles and other styles themselves have to be changed. Table are contained in the `<style:master-styles>` element.

```
<define name="table-table-template">
    <element name="table:table-template">
        <ref name="table-table-template-attlist"/>
        <optional>
            <ref name="table-first-row"/>
        </optional>
        <optional>
            <ref name="table-last-row"/>
        </optional>
        <optional>
            <ref name="table-first-column"/>
        </optional>
        <optional>
            <ref name="table-last-column"/>
        </optional>
        <choice>
            <ref name="table-body"/>
            <group>
                <ref name="table-even-rows"/>
                <ref name="table-odd-rows"/>
            </group>
            <group>
                <ref name="table-even-columns"/>
                <ref name="table-odd-columns"/>
            </group>
        </choice>
    </element>
</define>
```

Style Name

The table:name attribute specifies the name of the table template.

```
<define name="table-table-template-attlist" combine="interleave">
    <attribute name="text:name">
        <ref name="string"/>
    </attribute>
</define>
```

Corner Styles

The attributes `table:first-row-start-column`, `table:first-row-end-column`, `table:last-row-start-column` and `table:last-row-end-column` specify whether the cells in the four corners of the table should get the style from the row they are in or from the column. The possible values of these attributes are row and column.

```
<define name="table-table-template-attlist" combine="interleave">
    <attribute name="text:first-row-start-column">
        <ref name="rowOrCol"/>
    </attribute>
</define>

<define name="table-table-template-attlist" combine="interleave">
    <attribute name="text:first-row-end-column">
        <ref name="rowOrCol"/>
    </attribute>
</define>

<define name="table-table-template-attlist" combine="interleave">
    <attribute name="text:last-row-start-column">
        <ref name="rowOrCol"/>
    </attribute>
</define>

<define name="table-table-template-attlist" combine="interleave">
    <attribute name="text:last-row-end-column">
        <ref name="rowOrCol"/>
    </attribute>
</define>

<define name="rowOrCol">
    <choice>
        <value>row</value>
        <value>column</value>
    </choice>
</define>
```

14.5.1 Row and Column Styles

The elements `<table:first-row>` and `<table:last-row>` specify the cell styles that shall be applied to the first and last row of a table. They have a text:`style-name` attribute that references these styles.

The optional `text:paragraph-style-name` attribute specifies the paragraph style which should be applied to the empty paragraphs created in the corresponding cells

The elements `<table:first-col>` and `<table:last-col>` do the same for the first and last table column.

For the remaining cells, the cells styles can either be specified by the `<table:body>` element, or by the `<table:even-rows>`/`<table:odd-rows>` or `<table:even-columns>`/`<table:odd-columns>` element pairs if different cell styles should be applied to even and odd rows or columns.

```
<define name="table-first-row">
    <element name="table:first-row">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-last-row">
    <element name="table:last-row">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-first-column">
    <element name="table:first-column">
```

```
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-last-column">
    <element name="table:last-column">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-body">
    <element name="table:body">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-even-rows">
    <element name="table:even-rows">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-odd-rows">
    <element name="table:odd-rows">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-even-columns">
    <element name="table:even-columns">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="table-odd-columns">
    <element name="table:odd-columns">
        <ref name="common-table-template-attlist"/>
        <empty/>
    </element>
</define>

<define name="common-table-template-attlist" combine="interleave">
    <attribute name="text:style-name">
        <ref name="styleNameRef"/>
    </attribute>
    <attribute name="text:paragraph-style-name">
        <optional>
            <ref name="styleNameRef"/>
        </optional>
    </attribute>
</define>
```

14.6 Font Face Declaration

OpenDocument font face declarations directly correspond to the `@font-face` font description of [CSS2] (see §15.3.1) and the `<font-face>` element of [SVG] (see §20.8.3), but have the following two extensions:

- OpenDocument font face declarations optionally may have an unique name. This name can be used inside styles (i.e., as attribute of `<style:text-properties>` element) as value of the `style:font-name` attribute to immediately select a font face declaration. If a font face declaration is referenced this way, the steps described in §15.5 the [CSS2] font matching algorithms for selecting a font declaration based on the font-family, font-style, font-variant, font-weight and font-size descriptors will not take place, but the referenced font face declaration is used directly.
- Some additional font descriptor attributes exist. They are described below.

With the exception mentioned above, conforming applications should implement the CSS2 font matching algorithm as described in described in §15.5 the [CSS2], but they may also implement variants of it. They are especially allowed to implement a font matching based only on the font face declarations, that is, a font matching that is not applied to every character independently but only once for each font face declaration. This is useful for editing applications, where a font matching based on characters might be too expensive.

```
<define name="style-font-face">
    <element name="style:font-face">
        <ref name="style-font-face-attlist"/>
        <optional>
            <ref name="svg-font-face-src"/>
        </optional>
        <optional>
            <ref name="svg-definition-src"/>
        </optional>
    </element>
</define>
```

14.6.1 CSS2/SVG Font Descriptors

Font face declarations support the font descriptor attributes and elements described in §20.8.3 of [SVG].

```
<define name="style-font-face-attlist" combine="interleave">
    <optional>
        <attribute name="svg:font-family">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:font-style">
            <ref name="fontStyle"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:font-variant">
            <ref name="fontVariant"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:font-weight">
            <ref name="fontWeight"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:font-stretch">
            <choice>
                <value>normal</value>
                <value>ultra-condensed</value>
                <value>extra-condensed</value>
                <value>condensed</value>
                <value>semi-condensed</value>
                <value>semi-expanded</value>
                <value>expanded</value>
                <value>extra-expanded</value>
                <value>ultra-expanded</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:font-size">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:unicode-range"/>
    </optional>
    <optional>
        <attribute name="svg:units-per-em">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:panose-1"/>
    </optional>
    <optional>
        <attribute name="svg:stemv">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:stemh">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:slope">
            <ref name="integer"/>
        </attribute>
        </optional>
    <optional>
        <attribute name="svg:cap-height">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:x-height">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:accent-height">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:ascent">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:descent">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:widths"/>
    </optional>
    <optional>
        <attribute name="svg:bbox"/>
    </optional>
    <optional>
        <attribute name="svg:ideographic">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:alphabetic">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:mathematical">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:hanging">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:v-ideographic">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:v-alphabetic">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:v-mathematical">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:v-hanging">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:underline-position">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
```

```
        <attribute name="svg:underline-thickness">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:strikethrough-position">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:strikethrough-thickness">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:overline-position">
            <ref name="integer"/>
        </attribute>
        </optional>
    <optional>
        <attribute name="svg:overline-thickness">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>

<define name="svg-font-face-src">
    <element name="svg:font-face-src">
        <oneOrMore>
            <choice>
                <ref name="svg-font-face-uri"/>
                <ref name="svg-font-face-name"/>
            </choice>
        </oneOrMore>
    </element>
</define>

<define name="svg-font-face-uri">
    <element name="svg:font-face-uri">
        <ref name="common-svg-font-face-xlink-attlist"/>
        <zeroOrMore>
            <ref name="svg-font-face-format"/>
        </zeroOrMore>
    </element>
</define>

<define name="svg-font-face-format">
    <element name="svg:font-face-format">
        <optional>
            <attribute name="svg:string"/>
        </optional>
        <empty/>
    </element>
</define>
<define name="svg-font-face-name">
    <element name="svg:font-face-name">
        <optional>
            <attribute name="svg:name"/>
        </optional>
        <empty/>
    </element>
</define>
```

```
<define name="svg-definition-src">
    <element name="svg:definition-src">
        <ref name="common-svg-font-face-xlink-attlist"/>
    <empty/>
    </element>
</define>

<define name="common-svg-font-face-xlink-attlist" combine="interleave">
    <attribute name="xlink:href">
        <ref name="anyURI"/>
    </attribute>
    <optional>
        <attribute name="xlink:type" a:defaultValue="simple">
            <value>simple</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate" a:defaultValue="onRequest">
            <value>onRequest</value>
        </attribute>
    </optional>
</define>
```

14.6.2 Name

The `style:name` attribute specifies the unique name of the font declaration. This name can be used inside styles (i.e., as an attribute of the `<style:text-properties>` element) as value of the `style:font-name` attribute to immediately select a font face declaration

```
<define name="style-font-face-attlist" combine="interleave">
    <attribute name="style:name">
        <ref name="string"/>
    </attribute>
</define>
```

14.6.3 Adornments

The `style:font-adornments` attributes specifies adornments, like bold or italic that can be used to locate a font in addition to the family name.

```
<define name="style-font-face-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-adornments">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

14.6.4 Font Family Generic

The `style:font-family-generic` attribute specifies a generic font family name. See section 15.4.15 for details.

```
<define name="style-font-face-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-family-generic">
            <ref name="fontFamilyGeneric"/>
        </attribute>
    </optional>
</define>
```

14.6.5 Font Pitch

The `style:font-pitch` attribute specifies whether a font has a fixed or variable width. See section 15.4.17 for details.

```
<define name="style-font-face-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-pitch">
            <ref name="fontPitch"/>
        </attribute>
    </optional>
</define>

```

14.6.6 Font Character Set

The `style:font-charset` attribute specifies the character set of a font. See section 15.4.18 for details.

```
<define name="style-font-face-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-charset">
            <ref name="textEncoding"/>
        </attribute>
    </optional>
</define>
```

14.7 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 urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0. The prefix `number` denotes the data styles namespace.

This section describes the OpenDocument representation of the following data styles:

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

14.7.1 Number Style

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

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:text-properties>` element and a `<style:map>` element.

```
<define name="number-number-style">
    <element name="number:number-style">
        <ref name="common-data-style-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <optional>
            <ref name="number-text"/>
        </optional>
        <optional>
            <ref name="any-number"/>
            <optional>
                <ref name="number-text"/>
            </optional>
        </optional>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>

<define name="any-number">
    <choice>
        <ref name="number-number"/>
        <ref name="number-scientific-number"/>
        <ref name="number-fraction"/>
    </choice>
</define>
```

See section 14.7.9 for information about the attributes that may be associated with the number style elements.

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

- Number
- Scientific number
- Fraction

Number

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

This element is contained in the `<number:number-style>` element. The `<number:number>` element can contain multiple `<number:embedded-text>` elements.

The `number:decimal-replacement` and `number:display-factor` attributes may be used with this element. See also section 14.7.11 for information about additional attributes that may be associate with the `<number:number>` element.

```
<define name="number-number">
    <element name="number:number">
        <ref name="number-number-attlist"/>
        <ref name="common-decimal-places-attlist"/>
        <ref name="common-number-attlist"/>
        <zeroOrMore>
            <ref name="number-embedded-text"/>
        </zeroOrMore>
    </element>
</define>
```

Decimal Replacement

If a number style specifies that decimal places are used but the number displayed is an integer, a replacement text may be displayed instead of the decimal places. The `number:decimal-replacement` attribute specifies the replacement text.

Some applications may supports replacement text only that consists of the same number of “-” characters as decimal places.

```
<define name="number-number-attlist" combine="interleave">
    <optional>
        <attribute name="number:decimal-replacement"/>
    </optional>
</define>
```

Display Factor

The `number:display-factor` attribute specifies a factor by which each number is scaled (divided) before displaying. A factor of 1000, for example, causes numbers to be displayed in thousands.

Some applications may only support display factors of 1000 to the power of a non-negative integer number, that is 1, 1000, 1000000, 1000000000, etc.

```
<define name="number-number-attlist" combine="interleave">
    <optional>
        <attribute name="number:display-factor" a:defaultValue="1">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Embedded Text

The `<number:embedded-text>` element specifies text that is displayed at one specific position within a number. This element is different to a grouping separator, which appears several times within a number.

This element is contained in the `<number:number>` element. The `<number:number>` element can contain multiple occurrences of the `<number:embedded-text>` element to describe text at different positions in the number.

```
<define name="number-embedded-text">
    <element name="number:embedded-text">
        <ref name="number-embedded-text-attlist"/>
        <text/>
    </element>
</define>
```

The `number:position` attribute specifies the position where the text appears.

Position Attribute

The position is counted from right to left, from before the decimal point if one exists, or else from the end of the number. For example, position number 1 indicates that the text is inserted before the last digit. Position number 2 indicates that the text is inserted before the second last digit, and so on.

```
<define name="number-embedded-text-attlist" combine="interleave">
    <attribute name="number:position">
        <ref name="integer"/>
    </attribute>
</define>
```

Scientific Number

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

This element is contained in the `<number:number-style>` element.

The `number:min-exponent-digits` attribute may be used with this element. See section 14.7.11 for information on additional attributes that may be associated with the `<number:scientific-number>` element.

```
<define name="number-scientific-number">
    <element name="number:scientific-number">
        <ref name="number-scientific-number-attlist"/>
        <ref name="common-decimal-places-attlist"/>
        <ref name="common-number-attlist"/>
        <empty/>
    </element>
</define>
```

Minimum Exponent Digits

The `number:min-exponent-digits` attribute specifies the minimum number of digits to use to display an exponent. This attribute is supported for the `<number:scientific-number>` element.

```
<define name="number-scientific-number-attlist" combine="interleave">
    <optional>
        <attribute name="number:min-exponent-digits">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Fraction

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

This element is contained in the `<number:number-style>` element.

The `number:min-numerator-digits` and `number:min-denominator-digits` attributes may be used with this element. See section 14.7.11 for information on the attributes that may be associated with the `<number:fraction>` elements.

```
<define name="number-fraction">
    <element name="number:fraction">
        <ref name="number-fraction-attlist"/>
        <ref name="common-number-attlist"/>
        <empty/>
    </element>
</define>
```

Minimum Numerator Digits

The `number:min-numerator-digits` attribute specifies the minimum number of digits to use to display the numerator in a fraction.

```
<define name="number-fraction-attlist" combine="interleave">
    <optional>
        <attribute name="number:min-numerator-digits">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Minimum Denominator Digits

The `number:min-denominator-digits` attribute specifies the minimum number of digits to use to display the denominator of a fraction.

```
<define name="number-fraction-attlist" combine="interleave">
    <optional>
        <attribute name="number:min-denominator-digits">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Denominator Value

The `number:denominator-value` attribute specifies an integer value that is used as denominator of a fraction. If this attribute is not present, the application may choose an arbitrary denominator value.

```
<define name="number-fraction-attlist" combine="interleave">
    <optional>
        <attribute name="number:denominator-value">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

14.7.2 Currency Style

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

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:text-properties>` element and a `<style:map>` element.

```
<define name="number-currency-style">
    <element name="number:currency-style">
        <ref name="common-data-style-attlist"/>
        <ref name="common-auto-reorder-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <optional>
            <ref name="number-text"/>
        </optional>
        <optional>
            <choice>
                <group>
                    <ref name="number-and-text"/>
                    <optional>
                        <ref name="currency-symbol-and-text"/>
                    </optional>
                </group>
                <group>
                    <ref name="currency-symbol-and-text"/>
                    <optional>
                        <ref name="number-and-text"/>
                    </optional>
                </group>
            </choice>
        </optional>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>

<define name="currency-symbol-and-text">
    <ref name="number-currency-symbol"/>
    <optional>
        <ref name="number-text"/>
    </optional>
</define>
<define name="number-and-text">
    <ref name="number-number"/>
    <optional>
        <ref name="number-text"/>
    </optional>
</define>
```

See section 14.7.9 for information about the attributes that may be associated with the number style elements.

The following elements may be contained in the `<number:currency-style>` element:

- Number, see section 14.7.1.
- Currency symbol

Currency Symbol

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

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.

```
<define name="number-currency-symbol">
    <element name="number:currency-symbol">
        <ref name="number-currency-symbol-attlist"/>
        <text/>
    </element>
</define>
```

The `number:language` and `number:country` attributes may be used to specify the language and country of the currency symbol. See section 14.7.11 for information on the other attributes that may be associated with the currency style elements.

Currency Language and Country Attributes

If the currency symbol contained in a currency style belongs to a different language or country than the currency style itself, then the `number:language` and `number:country` attributes may be used to specify the language and country of the currency symbol.

```
<define name="number-currency-symbol-attlist" combine="interleave">
    <optional>
        <attribute name="number:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="number:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

14.7.3 Percentage Style

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

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. Some applications require that at least one `<number:text>` element exist and that its text must contain a “%” character.

In addition, the `<number:percentage-style>` element can contain a `<style:text-properties>` element and a `<style:map>` element.

```
<define name="number-percentage-style">
    <element name="number:percentage-style">
        <ref name="common-data-style-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <optional>
            <ref name="number-text"/>
        </optional>
        <optional>
            <ref name="number-and-text"/>
        </optional>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>
```

See section 14.7.9 for information on the attributes that may be associated with the percentage style element.

14.7.4 Date Style

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

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:text-properties>` element and a `<style:map>` element.

```
<define name="number-date-style">
    <element name="number:date-style">
        <ref name="common-data-style-attlist"/>
        <ref name="common-auto-reorder-attlist"/>
        <ref name="common-format-source-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <!-- This DTD does not reflect the fact that some elements must not -->
        <!-- occur more than once. -->
        <optional>
            <ref name="number-text"/>
        </optional>
        <oneOrMore>
            <ref name="any-date"/>
            <optional>
                <ref name="number-text"/>
            </optional>
        </oneOrMore>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>

<define name="any-date">
    <choice>
        <ref name="number-day"/>
        <ref name="number-month"/>
        <ref name="number-year"/>
        <ref name="number-era"/>
        <ref name="number-day-of-week"/>
        <ref name="number-week-of-year"/>
        <ref name="number-quarter"/>
        <ref name="number-hours"/>
        <ref name="number-am-pm"/>
        <ref name="number-minutes"/>
        <ref name="number-seconds"/>
    </choice>
</define>
```

See section 14.7.9 for information on the attributes that may be associated 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.

If this element is used, it should be contained in the `<number:date-style>` element.

```
<define name="number-day">
    <element name="number:day">
        <ref name="number-day-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:style` attribute may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Format Attribute

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

```
<define name="number-day-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Month

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

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-month">
    <element name="number:month">
        <ref name="number-month-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:textual` and `number:style` attributes may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Textual Representation Attribute

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

```
<define name="number-month-attlist" combine="interleave">
    <optional>
        <attribute name="number:textual" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Possessive Form Attribute

The `number:possessive-form` attribute determines whether the month is displayed as is (e.g., as in "17 January 2004") or using the possessive form (e.g., as in "17th day of January"). If the value of this attribute value is `true`, the name of the month is displayed in possessive form. If the attribute value is `false`, the number of the month is displayed as is.

```
<define name="number-month-attlist" combine="interleave">
    <optional>
        <attribute name="number:possessive-form" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Format Attribute

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

```
<define name="number-month-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Year

The `<number:year>` element specifies the year in the date.

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-year">
    <element name="number:year">
        <ref name="number-year-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:style` attribute may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Format Attribute

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

```
<define name="number-year-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Era

The `<number:era>` element specifies the era in which the year is counted.

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-era">
    <element name="number:era">
        <ref name="number-era-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:style` attribute may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Format Attribute

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

```
<define name="number-era-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Day Of Week

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

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-day-of-week">
    <element name="number:day-of-week">
        <ref name="number-day-of-week-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:style` attribute may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Format Attribute

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

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

```
<define name="number-day-of-week-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Week Of Year

The `<number:week-of-year>` element specifies the week of the year in the date.

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-week-of-year">
    <element name="number:week-of-year">
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

See section 14.7.11 for information on the the attributes that may be associated with the element.

Quarter

The `<number:quarter>` element specifies the quarter of the year in the date.

If used, this element must be contained in the `<number:date-style>` element.

```
<define name="number-quarter">
    <element name="number:quarter">
        <ref name="number-quarter-attlist"/>
        <ref name="common-calendar-attlist"/>
        <empty/>
    </element>
</define>
```

The `number:style` attribute may be used with this element. See section 14.7.11 for information on the other attributes that may be associated with the element.

Format Attribute

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

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

```
<define name="number-quarter-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

14.7.5 Time Style

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

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:text-properties>` element and a `<style:map>` element.

```
<define name="number-time-style">
    <element name="number:time-style">
        <ref name="number-time-style-attlist"/>
        <ref name="common-data-style-attlist"/>
        <ref name="common-format-source-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <!-- This DTD does not reflect the fact that some elements must not -->
        <!-- occur more than once. -->
        <optional>
            <ref name="number-text"/>
        </optional>
        <oneOrMore>
            <ref name="any-time"/>
            <optional>
                <ref name="number-text"/>
            </optional>
        </oneOrMore>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>

<define name="any-time">
    <choice>
        <ref name="number-hours"/>
        <ref name="number-am-pm"/>
        <ref name="number-minutes"/>
        <ref name="number-seconds"/>
    </choice>
</define>
```

See section 14.7.9 for information on the attributes that may be associated 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

Time Value Truncation

If a time or duration is too large to be displayed using the default value range for a time component, (0 to 23 for `<number:hours>`) , the `number:truncate-on-overflow` attribute may be used to specify whether the time or duration value should be truncated or whether the value range becomes extended.

```
<define name="number-time-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:truncate-on-overflow" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Hours

The `<number:hours>` element specifies if hours are displayed as part of a date or time.

```
<define name="number-hours">
    <element name="number:hours">
        <ref name="number-hours-attlist"/>
        <empty/>
    </element>
</define>
```

Format Attribute

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

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

```
<define name="number-hours-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Minutes

The `<number:minutes>` element specifies if minutes are displayed as part of a date or time.

```
<define name="number-minutes">
    <element name="number:minutes">
        <ref name="number-minutes-attlist"/>
        <empty/>
    </element>
</define>
```

Format Attribute

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

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

```
<define name="number-minutes-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Seconds

The `<number:seconds>` element specifies if seconds are displayed as part of a date or time.

```
<define name="number-seconds">
    <element name="number:seconds">
        <ref name="number-seconds-attlist"/>
        <empty/>
    </element>
</define>
```

Format Attribute

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

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

```
<define name="number-seconds-attlist" combine="interleave">
    <optional>
        <attribute name="number:style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Decimal Places Attribute

The `number:decimal-places` attribute determines the number of decimal places to use when displaying fractions.

If this attribute is not present or if the value of the attribute is `0`, fractions are not displayed.

```
<define name="number-seconds-attlist" combine="interleave">
    <optional>
        <attribute name="number:decimal-places" a:defaultValue="0">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

AM/PM

The `<number:am-pm>` element specifies if AM/PM is included as part of the date or time.

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

```
<define name="number-am-pm">
    <element name="number:am-pm">
        <empty/>
    </element>
</define>
```

14.7.6 Boolean Style

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

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:text-properties>` element and a `<style:map>` element.

```
<define name="number-boolean-style">
    <element name="number:boolean-style">
        <ref name="common-data-style-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <optional>
            <ref name="number-text"/>
        </optional>
        <optional>
            <ref name="number-boolean"/>
            <optional>
                <ref name="number-text"/>
            </optional>
        </optional>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>
```

Boolean

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

```
<define name="number-boolean">
    <element name="number:boolean">
        <empty/>
    </element>
</define>
```

14.7.7 Text Style

The `<number:text-style>` element describes the style for displaying text.

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:text-properties>`

element and a `<style:map>` element. The `<number:text-content>` elements represent the variable text content to display, while the `<number:text>` elements contain any additional fixed text to display.

```
<define name="number-text-style">
    <element name="number:text-style">
        <ref name="common-data-style-attlist"/>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
        <optional>
            <ref name="number-text"/>
        </optional>
        <zeroOrMore>
            <ref name="number-text-content"/>
            <optional>
                <ref name="number-text"/>
            </optional>
        </zeroOrMore>
        <zeroOrMore>
            <ref name="style-map"/>
        </zeroOrMore>
    </element>
</define>
```

See section 14.7.9 for information on the attributes that may be associated with the text style elements.

Fixed Text

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

This element is contained in all data styles element.

```
<define name="number-text">
    <element name="number:text">
        <text/>
    </element>
</define>
```

Text Content

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

```
<define name="number-text-content">
    <element name="number:text-content">
        <empty/>
    </element>
</define>
```

14.7.8 Common Data Style Elements

The following common style elements may be contained within data style elements:

- Text formatting properties
- Style mappings

Formatting Properties

The `<style:text-properties>` element specifies the text formatting properties to apply to any text displayed in the data style. See section 15.4 for information on the formatting properties element.

The purpose of specifying text formatting properties within data styles is mainly to highlight certain values (for instance negative ones) by using style mappings. For this reason, data styles usually support only very few text formatting properties, for instance a text color. There may be also restrictions for the values of text formatting properties. For instance, the only value allowed for the text color might be read.

Style Mappings

The `<style:map>` element specifies an alternative data style to map to if a certain condition exists. See section 14.1.1 for information on the `<style:map>` element.

The following rules exist for using style maps element with data style elements:

- 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.

14.7.9 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
- Transliteration

Name

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

```
<define name="common-data-style-attlist" combine="interleave">
    <attribute name="style:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `style:display-name` attribute specifies the name of the style as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

The `style:display-name` attribute can be used with all data style elements.

```
<define name="style-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Language

The `number:language` attribute specifies the language of the style. The value of the attribute is a language code in conformance with [RFC3066]. The language code is used to retrieve information about any display properties that are language-dependent. The language attribute can be used with all data style elements.

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.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
</define>
```

Country

The `number:country` attribute specifies the country of the style. The value of the attribute is a country code in conformance with [RFC3066]. The country code is used to retrieve information about any display properties that are country-dependent. The language attribute can be used with all data style elements.

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.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

Title

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

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:title"/>
    </optional>
</define>
```

Volatility

Sometimes when a document is opened, not all of the styles contained in the document are actually referenced. The application may retain or discard this unused styles. This may be controlled by the `style:volatile` attribute, that is supported by all data style elements.

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.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="style:volatile">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

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

The attribute value can be `true` or `false`.

```
<define name="common-auto-reorder-attlist" combine="interleave">
    <optional>
        <attribute name="number:automatic-order" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Format Source

The `number:format-source` attribute specifies the source of the `short` and `long` display formats. It is used with the following elements:

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

The value of this attribute can be `fixed` or `language`.

If the value is `fixed`, the meaning of the values `number:style` attribute's values `short` and `long` is as described in this specification.

If the value of the `number:format-source` attribute is `language`, the meaning of short and long depends on the language and country of the date style, or, if neither of these are specified, applications should use the system settings for short and long date and time formats.

```
<define name="common-format-source-attlist">
    <optional>
        <attribute name="number:format-source" a:defaultValue="fixed">
            <choice>
                <value>fixed</value>
                <value>language</value>
            </choice>
        </attribute>
    </optional>
</define>
```

14.7.10 Transliteration

The various `number:transliteration-*` attributes specify the native number system of the style to display the number using, for example, CJK number characters. The notation is inspired by the W3C XSLT 2.0 draft, see §12.3 of [XSLT2]. However, to be able to fully distinguish between all possible native number systems additional attributes are needed in combination. For example, Korean uses 11 different systems where the digits are not always different but short and long and formal and informal forms exist.

The transliteration attributes can be used with all data style elements.

Transliteration Format

The `number:transliteration-format` attribute specifies which number characters to use. The value of the attribute is the digit "1" expressed as a native number.

If no format is specified the default ASCII representation of Arabic digits is used, other transliteration attributes present in this case are ignored.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:transliteration-format" a:defaultValue="1">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Transliteration Language

The `number:transliteration-language` attribute specifies which language the native number system belongs to. The value of the attribute is a language code in conformance with [RFC3066].

If no language/country (locale) combination is specified the locale of the data style is used.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:transliteration-language">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

Transliteration Country

The `number:transliteration-country` attribute specifies which country the native number system belongs to. The value of the attribute is a country code in conformance with [RFC3066].

If no language/country (locale) combination is specified the locale of the data style is used.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:transliteration-country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

Transliteration Style

The `number:transliteration-style` attribute specifies which style the native number system belongs to. If more than one native number system matches the transliteration-format this attribute selects one. A short style should result in a one to one mapping of Arabic digits to native number digits if possible.

```
<define name="common-data-style-attlist" combine="interleave">
    <optional>
        <attribute name="number:transliteration-style" a:defaultValue="short">
            <choice>
                <value>short</value>
                <value>medium</value>
                <value>long</value>
            </choice>
        </attribute>
    </optional>
</define>
```

14.7.11 Common Data Style Child Element 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. This attribute is supported for the following elements:

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

If this attribute is not specified, a default number of decimal places is used.

```
<define name="common-decimal-places-attlist">
    <optional>
        <attribute name="number:decimal-places">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

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. This attribute is supported for the following elements:

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

If this attribute is not specified, a default number of integer digits is used.

```
<define name="common-number-attlist" combine="interleave">
    <optional>
        <attribute name="number:min-integer-digits">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Grouping Separator

The `number:grouping` attribute specifies whether or not the integer digits of a number should be grouped using a separator character. This attribute is supported for the following elements:

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

The grouping character that is used and the number of digits that are grouped together depends on the language and country of the style.

```
<define name="common-number-attlist" combine="interleave">
    <optional>
        <attribute name="number:grouping" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Calendar System

The `number:calendar` attribute specifies the calendar system used to extract parts of a date. This attribute is supported for the following elements:

- `<number:day>`
- `<number:month>`

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

The attribute may have the values `gregorian`, `gengou`, `ROC`, `hanja_yoil`, `hanja`, `hijri`, `jewish`, `buddhist` or an arbitrary string value. If this attribute is not specified, the default calendar system is used.

```
<define name="common-calendar-attlist" combine="interleave">
    <optional>
        <attribute name="number:calendar">
            <choice>
                <value>gregorian</value>
                <value>gengou</value>
                <value>ROC</value>
                <value>hanja_yoil</value>
                <value>hanja</value>
                <value>hijri</value>
                <value>jewish</value>
                <value>buddhist</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

14.8 Text Styles

14.8.1 Text Styles

Text styles are `<style:style>` elements that have the family `text`. They can be used within all kind of applications to specify formatting properties for piece of text. They support the text properties as described in section 15.4.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>text</value>
        </attribute>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </group>
</define>
```

14.8.2 Paragraph Styles

Paragraph styles are `<style:style>` elements that have the family `paragraph`. They can be used within all kind of applications to specify formatting properties for paragraphs and headings. They support the paragraph properties described in section 15.5 as well as the text properties described in section 15.4.

```
<define name="style-style-content" combine="choice">
```

```
    <group>
        <attribute name="style:family">
            <value>paragraph</value>
        </attribute>
        <optional>
            <ref name="style-paragraph-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </group>
</define>
```

14.8.3 Section Styles

Section styles are `<style:style>` elements that have the family `section`. They can be used within text documents to specify formatting properties for a text section. They support the section properties as described in section 15.7.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>section</value>
        </attribute>
        <optional>
            <ref name="style-section-properties"/>
        </optional>
    </group>
</define>
```

14.8.4 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. Ruby styles support the formatting properties described in section 15.6.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>ruby</value>
        </attribute>
        <optional>
            <ref name="style-ruby-properties"/>
        </optional>
    </group>
</define>
```

14.9 Enhanced Text Styles

14.9.1 Line Numbering Configuration

A document can contain *none* or *one* line numbering configuration element `<text:linenumbering-configuration>` within the `<office:styles>` element. If the element is not present, a default line numbering configuration is used. The default line numbering may vary on the office application software, but every document saved by an application that supports line numbering should contain a line numbering configuration element.

```
<define name="text-linenumbering-configuration">
    <element name="text:linenumbering-configuration">
        <ref name="text-linenumbering-configuration-attlist"/>
        <optional>
            <ref name="text-linenumbering-separator"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<text:linenumbering-configuration>` element are:

- Line numbering enable
- Number format
- Text style
- Increment
- Position
- Offset
- Count empty lines
- Count line in text boxes
- Restart numbering on every page

The following element may be included in the `<text:linenumbering-separator>` element:

- Separator

Line Numbering Enable

The `text:number-lines` attribute controls whether or not lines are numbered.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:number-lines" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Number Format

See section 12.2 for detailed information on number format attributes. The attributes described in section 12.2 can also be associated with the `<text:linenumbering-configuration>` element.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <ref name="common-num-format-attlist"/>
    </optional>
</define>
```

Text Style

The `text:style-name` attribute specifies the text style for all line numbers. The value of this attribute is the name of the text style that is applied to all line numbers.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

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.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:increment">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Position

The `text:position` attribute determines whether the line numbers are printed on the left, right, inner, or outer margins.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:number-position" a:defaultValue="left">
            <choice>
                <value>left</value>
                <value>right</value>
                <value>inner</value>
                <value>outer</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Offset

The `text:offset` attribute determines the distance between the line number and the margin.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:offset">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

Count Empty Lines

The text:count-empty-lines attribute determines whether or not empty lines are included in the line count. If the value of this attribute is true, empty lines are included in the line count.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:count-empty-lines" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Count Lines in Text Boxes

The text:count-in-text-boxes attribute determines whether or not text in text boxes is included in the line count. If the value of this attribute is true, text within text boxes is included in the line count.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:count-in-text-boxes" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Restart Numbering on Every Page

The text:restart-on-page attribute determines whether or not the line count is reset to 1 at the start of 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.

```
<define name="text-linenumbering-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:restart-on-page" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Separator

The <text:linenumbering-separator> element contains the text that is displayed as a separator. A separator is text that is displayed instead of a line number for lines where no number is displayed.

This element is contained in the line numbering configuration element. If the element is not present, no separator is displayed.

The element's text:increment attribute causes the separator to appear on lines that are a multiple of the given increment. For example, if the increment is 2, only lines 2, 4, 6, and so on get a separator, provided that no number is displayed already.

```
<define name="text-linenumbering-separator">
    <element name="text:linenumbering-separator">
```

```
        <optional>
            <attribute name="text:increment">
                <ref name="nonNegativeInteger"/>
            </attribute>
        </optional>
        <text/>
    </element>
</define>
```

14.9.2 Notes Configuration Element

A document in OpenDocument format contains at most one notes configuration element for every notes class used in the document. If there is no note configuration element, a default note configuration is used.

```
<define name="text-notes-configuration">
    <element name="text:notes-configuration">
        <ref name="text-notes-configuration-content"/>
    </element>
</define>
```

The attributes that may be associated with the `<text:notes-configuration>` element are:

- Note class
- Citation text style
- Citation body text style
- Default footnote paragraph style
- Master page
- Start value
- Number format
- Numbering scheme
- Footnote position

The following element may be included in the `<text:footnotes-configuration>` element:

- Footnote continuation notice (forward and backward)

Note class

The note class attribute determines which note elements this notes configuration applies to.

```
<define name="text-notes-configuration-content" combine="interleave">
    <ref name="text-note-class"/>
</define>
```

Citation Text Style

The `text:citation-style` attribute specifies the text style to use for the footnote citation within the footnote.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:citation-style-name">
```

```
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Citation Body Text Style

The `text:citation-body-style-name` attribute specifies the text style to use for the footnote citation in the text flow.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:citation-body-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Default Note 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.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:default-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Master Page

To display the footnotes at the end of the document, the pages that contain the footnotes must be instances of the master page specified by the `text:master-page-name` attribute.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:master-page-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Start Value

The `start:value` attribute specifies the value at which the footnote numbering starts.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:start-value">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Number Format

See section 12.2 for information on the number format for footnotes.

```
<define name="text-notes-configuration-content" combine="interleave">
    <ref name="common-num-format-prefix-suffix-attlist"/>
    <optional>
        <ref name="common-num-format-attlist"/>
    </optional>
</define>
```

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.

> **Note:** [XSLT] does not have the capability to start with new footnote numbers on every page.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:start-numbering-at">
            <choice>
                <value>document</value>
                <value>chapter</value>
                <value>page</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Footnotes Position

The `text:footnotes-position` attribute specifies one of the following positions for footnotes:

- `text`: At the page where the footnote citation is located, immediately below the page's text.
- `page`: The bottom of the page where the footnote citation is located.
- `section`: The end of the section
- `document`: The end of the document.

Note: [XSL] does not have the capability to display footnotes at the end of the document. However, an [XSLT] stylesheet may generate some other flow objects to display such footnotes.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:footnotes-position">
            <choice>
                <value>text</value>
                <value>page</value>
                <value>section</value>
                <value>document</value>
            </choice>
        </attribute>
    </optional>
</define>
```

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

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <element name="text:note-continuation-notice-forward">
            <text/>
        </element>
    </optional>
</define>
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <element name="text:note-continuation-notice-backward">
            <text/>
        </element>
    </optional>
</define>
```

Example: Footnote configuration

```
<text:notes-configuration
    text:notes-type="footnote"
    text:citation-style="Footnote symbol"
    text:default-style="Footnote">
    <text:note-continuation-notice-forward>" .."
    </text:note-continuation-notice-forward>
    <text:note-continuation-notice-forward>".. "
    </text:note-continuation-notice-forward>
</text:notes-configuration>
```

14.9.3 Bibliography Configuration

The bibliography configuration element `<text:bibliography-configuration>` is contained in the document's style section. It contains information how bibliography entries are displayed in-line, and how they are displayed in the bibliography index.

```
<define name="text-bibliography-configuration">
    <element name="text:bibliography-configuration">
        <ref name="text-bibliography-configuration-attlist"/>
        <zeroOrMore>
            <ref name="text-sort-key"/>
        </zeroOrMore>
    </element>
</define>
```

Prefix and Suffix

The `text:prefix` and `text:suffix` attributes contain a string that is displayed before and after an bibliography entry's short name or number if it occurs in the document body.

```
<define name="text-bibliography-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:prefix">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:suffix">
            <ref name="string"/>
        </attribute>
    </optional>
```

```
</define>
```

Numbered Entries

The `text:numbered-entry` attribute specifies whether a number is displayed for bibliography entries instead of their short name.

Example: With prefix and suffix "[" and "]" a bibliography entry with short name "Abc123" would be displayed as "[Abc123]" in the document body if `text:numbered-entry` has the value `false`, and for instance as “[5]”, if it has the value `true`.

```
<define name="text-bibliography-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:numbered-entries" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Sorting

The `text:sort-by-position` attribute specifies whether bibliography entries are displayed in the order of their positions in the document, or by an arbitrary selection of entry fields, e.g., author name or publication date. In the later case, the collating order for entries is determined by the triplet language/country/sort-algorithm as specified in the attributes `fo:language`, `fo:country` and `text:sort-algorithm`. See also section 7.8.

```
<define name="text-bibliography-configuration-attlist" combine="interleave">
    <optional>
        <attribute name="text:sort-by-position" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:sort-algorithm">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Sort Keys

The `<text:sort-key>` element specifies a single sort key if bibliography entries are not displayed in document order. It has an attribute `text:key`, that contains the type of index entry data that should be used for sorting (see also section 7.1.4) and an attribute `text:sort-ascending` that specifies whether sorting takes pace in ascending or descending order.

```
<define name="text-sort-key">
    <element name="text:sort-key">
```

```
        <ref name="text-sort-key-attlist"/>
        <empty/>
    </element>
</define>

<define name="text-sort-key-attlist" combine="interleave">
    <attribute name="text:key">
        <choice>
            <value>address</value>
            <value>annote</value>
            <value>author</value>
            <value>bibliography-type</value>
            <value>booktitle</value>
            <value>chapter</value>
            <value>custom1</value>
            <value>custom2</value>
            <value>custom3</value>
            <value>custom4</value>
            <value>custom5</value>
            <value>edition</value>
            <value>editor</value>
            <value>howpublished</value>
            <value>identifier</value>
            <value>institution</value>
            <value>isbn</value>
            <value>issn</value>
            <value>journal</value>
            <value>month</value>
            <value>note</value>
            <value>number</value>
            <value>organizations</value>
            <value>pages</value>
            <value>publisher</value>
            <value>report-type</value>
            <value>school</value>
            <value>series</value>
            <value>title</value>
            <value>url</value>
            <value>volume</value>
            <value>year</value>
        </choice>
    </attribute>
    <optional>
        <attribute name="text:sort-ascending" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

14.10 List Style

List styles specify the formatting properties for lists. A `<text:list-style>` element contains a set of style elements for each list level, which are called **list level styles**. There are three different list level style elements, depending on whether this particular list level is to have a list label containing the list numbering, a bullet, or an image.

If a list style is applied to a list but does not contain a list level specification for the suitable level, the list level style of the next lower level is used. If no suitable list level exists, a default style is used.

```
<define name="text-list-style">
```

```
    <element name="text:list-style">
        <ref name="text-list-style-attr"/>
        <zeroOrMore>
            <ref name="text-list-style-content"/>
        </zeroOrMore>
    </element>
</define>
```

> **Note:** List styles contain different properties than paragraph or text styles. This is why they are represented by a different element.

The attributes that may be associated with the `<text:list-style>` element are:

- Name
- Display name
- Consecutive numbering

Name

The `style:name` attribute specifies the name of the list style.

```
<define name="text-list-style-attr" combine="interleave">
    <attribute name="style:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `style:display-name` attribute specifies the name of the list style as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="text-list-style-attr" combine="interleave">
    <optional>
        <attribute name="style:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

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.

```
<define name="text-list-style-attr" combine="interleave">
    <optional>
        <attribute name="text:consecutive-numbering" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

14.10.1 Common List-Level Style Attributes

The following attributes can be used on all list-level styles:

Level

The `text:level` attribute specifies the level of the number list style.

```
<define name="text-list-level-style-attr">
    <attribute name="text:level">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

14.10.2 Number Level Style

A number level style specifies a list style where the list items are preceded by numbers.

```
<define name="text-list-style-content" combine="choice">
    <element name="text:list-level-style-number">
        <ref name="text-list-level-style-attr"/>
        <ref name="text-list-level-style-number-attr"/>
        <optional>
            <ref name="style-list-level-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<text:list-level-style-number>` element are:

- Level (see section 14.10.1)
- Text style
- Number format
- Display levels
- Start value

Additional formatting properties may be contained in the `<style:list-level-properties>` and `<style:text-properties>` elements. See sections 15.12 and 15.4 for details.

Text Style

The `text:style-name` attribute specifies the name of the character style to use to format the number of the list.

```
<define name="text-list-level-style-number-attr" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Number Format

See section 12.2 for detailed information on number format attributes. The attributes described in section 12.2 can also be associated with the `<text:list-level-style-number>` element. The `style:num-format` attribute can be empty. In this case, no number is displayed.

```
<define name="text-list-level-style-number-attr" combine="interleave">
    <ref name="common-num-format-attlist"/>
    <ref name="common-num-format-prefix-suffix-attlist"/>
</define>
```

Display Levels

The `text:display-levels` attribute specifies the number of levels whose numbers are displayed at the current level.

```
<define name="text-list-level-style-number-attr" combine="interleave">
    <optional>
        <attribute name="text:display-levels" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Example: Given a third-level chapter number 1.2.3. Values of text:display-number from 1 to three would achieve the following results:

`text:display-number`	***display***
1	1
2	1.2
3	1.2.3

Start Value

The `text:start-value` attribute specifies the first number of a list item of the current level.

```
<define name="text-list-level-style-number-attr" combine="interleave">
    <optional>
        <attribute name="text:start-value" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

14.10.3 Bullet Level Style

A bullet level style element specifies a list style where the list items are preceded by bullets.

```
<define name="text-list-style-content" combine="choice">
    <element name="text:list-level-style-bullet">
        <ref name="text-list-level-style-attr"/>
        <ref name="text-list-level-style-bullet-attr"/>
        <optional>
            <ref name="style-list-level-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<text:list-level-style-bullet>` element are:

- Level (see section 14.10.1)
- Text style
- Bullet character
- Prefix and suffix
- Bullet relative size

Additional formatting properties may be contained in the `<style:list-level-properties>` and `<style:text-properties>` elements. See sections 15.12 and 15.4 for details.

Text Style

The `text:style-name` attribute specifies the name of the character style to use to format the list bullet.

```
<define name="text-list-level-style-bullet-attr" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Bullet Character

The bullet character attribute specifies the [UNICODE] character to use as the bullet in a bullet level style.

Typical bullet characters are:

UNICODE Character Code	***Typical Shape***	***UNICODE Character Name***	***Reference***
U+2022	•	BULLET	http://www.unicode.org/charts/PDF/U2000.pdf
U+25CF	●	BLACK CIRCLE	http://www.unicode.org/charts/PDF/U25A0.pdf
U+2714	✔	HEAVY CHECK MARK	http://www.unicode.org/charts/PDF/U2700.pdf
U+2717	✗	BALLOT X	
U+2794	➔	HEAV WIDE-HEADED RIGHTWARDS ARROW	
U+27A2	➢	THREE-D TOP-LIGHTED RIGHTWARDS ARROWHEAD	

These characters might not be available within some fonts.

```
<define name="text-list-level-style-bullet-attr" combine="interleave">
    <attribute name="text:bullet-char">
        <ref name="character"/>
    </attribute>
</define>
```

Prefix and Suffix

The attributes `style:num-prefix` and `style:num-suffix` specified in section 12.2 can be used to add characters before or behind the bullet character.

```
<define name="text-list-level-style-bullet-attr" combine="interleave">
    <ref name="common-num-format-prefix-suffix-attlist"/>
</define>
```

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.

```
<define name="text-list-level-style-bullet-attr" combine="interleave">
    <optional>
        <attribute name="text:bullet-relative-size">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

14.10.4 Image Level Style

An image level style element specifies a list style where the list items are preceded by images. This element can be an [XLink] and can only be contained in list style elements.

```
<define name="text-list-style-content" combine="choice">
    <element name="text:list-level-style-image">
        <ref name="text-list-level-style-attr"/>
        <ref name="text-list-level-style-image-attr"/>
        <optional>
            <ref name="style-list-level-properties"/>
        </optional>
    </element>
</define>
```

The following elements and attributes may be associated with the `<text:list-level-style-image>` element are:

- Level (see section 14.10.1)
- Image location

Additional formatting properties may be contained in the `<style:list-level-properties>` element. See section 15.12 for details.

Image Location

The image data can be stored in one of the following ways (see also section 9.3.2):

- The image data is located in an external file. Use the `xlink:href` attribute described below to specify the location of the file.
- The image data is contained in the `<text:list-level-style-image>` element. The `<text:list-level-style-image>` element must contain an `<office:binary-data>` element that contains the image data in BASE64 encoding. In this situation, the `xlink:href` attribute is not required.

```
<define name="text-list-level-style-image-attr" combine="interleave">
    <choice>
        <ref name="common-draw-data-attlist"/>
        <ref name="office-binary-data"/>
    </choice>
</define>
```

14.10.5 List Level Style Example

Example: List level style

```
<text:list-style style:name="List 1">
    <text:list-level-style-number 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:list-level-properties fo:width=".27cm" fo:height=".27cm"
            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.

     ● This is a sub sub list item.

     ● This is a sub sub list item.

3. This is the third list item.

14.11 Outline Style

The outline style is a list style that is applied to all headings within a text document where the heading's paragraph style does not define a list style to use itself.

The way in which the OpenDocument format represents outline numbering styles is very similar to the way it represents list styles. The `<text:outline-style>` element contains elements that specify the style of each outline level. It can be contained within the `<office:styles>` element only.

```
<define name="text-outline-style">
    <element name="text:outline-style">
        <oneOrMore>
            <ref name="text-outline-level-style"/>
        </oneOrMore>
    </element>
</define>
```

14.11.1 Outline Level Style

The `<text:outline-level-style>` element specifies the style for each outline level. This element is contained in `<text:outline-style>` elements only.

```
<define name="text-outline-level-style">
    <element name="text:outline-level-style">
        <ref name="text-outline-level-style-attlist"/>
        <optional>
            <ref name="style-list-level-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </element>
</define>
```

The attributes that may be associated with the `<text:outline-level-style>` element are:

- Level
- Text style
- Number format
- Display levels
- Start value

Additional formatting properties may be contained in the `<style:list-level-properties>` and `<style:text-properties>` element. See sections 15.12 and 15.4 for details.

Level

The `text:level` attribute specifies the level of the outline style.

```
<define name="text-outline-level-style-attlist" combine="interleave">
    <attribute name="text:level">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

Text Style

The `text:style-name` attribute specifies the name of the character style to use to format the number of the heading.

```
<define name="text-outline-level-style-attlist" combine="interleave">
    <optional>
        <attribute name="text:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

Number Format

See section 14.10.2 for information on the number format attributes.

```
<define name="text-outline-level-style-attlist" combine="interleave">
    <ref name="common-num-format-attlist"/>
    <ref name="common-num-format-prefix-suffix-attlist"/>
</define>
```

Display Levels

The `text:display-levels` attribute specifies the number of levels whose numbers are displayed at the current level. See also section 14.10.2.

```
<define name="text-outline-level-style-attlist" combine="interleave">
    <optional>
        <attribute name="text:display-levels" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Start Value

The `text:start-value` attribute specifies the first number of a heading of the current level.

```
<define name="text-outline-level-style-attlist" combine="interleave">
    <optional>
        <attribute name="text:start-value" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

14.12 Table Styles

14.12.1 Table Styles

Table styles are `<style:style>` elements that have the family `table`. They can be used within all kind of applications to specify formatting properties for tables. They support the table properties as described in section 15.8.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>table</value>
        </attribute>
        <optional>
            <ref name="style-table-properties"/>
        </optional>
    </group>
</define>
```

14.12.2 Table Column Styles

Table column styles are `<style:style>` elements that have the family `table-column`. They can be used within all kind of applications to specify formatting properties for table columns. They support the table column properties as described in section 15.9.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>table-column</value>
        </attribute>
        <optional>
            <ref name="style-table-column-properties"/>
        </optional>
    </group>
</define>
```

14.12.3 Table Row Styles

Table row styles are `<style:style>` elements that have the family `table-row`. They can be used within all kind of applications to specify formatting properties for table rows. They support the table properties as described in section 15.10.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>table-row</value>
        </attribute>
        <optional>
            <ref name="style-table-row-properties"/>
        </optional>
    </group>
</define>
```

14.12.4 Table Cell Styles

Table cell styles are `<style:style>` elements that have the family `table-cell`. They can be used within all kind of applications to specify formatting properties for table cells. They support the table properties as described in section 15.11 as well as the paragraph and text properties as described in sections 15.5 and 15.4.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>table-cell</value>
        </attribute>
        <optional>
            <ref name="style-table-cell-properties"/>
        </optional>
        <optional>
            <ref name="style-paragraph-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </group>
</define>
```

14.13 Graphic Styles

14.13.1 Graphic and Presentation Styles

Graphic and presentation styles are `<style:style>` elements that have either the family `graphic` or `presentation`. Graphic styles with family graphic may occur within all kinds of applications, graphic styles with family presentation may occur only within presentation documents. Both kind of styles support the graphic properties described in section 15.17. They may also contain paragraph and text properties as described in sections 15.5 and 15.4.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <choice>
                <value>graphic</value>
                <value>presentation</value>
            </choice>
        </attribute>
        <optional>
            <ref name="style-graphic-properties"/>
        </optional>
        <optional>
            <ref name="style-paragraph-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </group>
</define>

<define name="style-graphic-properties">
    <element name="style:graphic-properties">
        <ref name="style-graphic-properties-content"/>
    </element>
</define>

<define name="style-graphic-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-graphic-properties-content-strict">
    <ref name="style-graphic-properties-attlist"/>
    <ref name="style-graphic-fill-properties-attlist"/>
    <ref name="style-graphic-properties-elements"/>
</define>

<define name=" style-graphic-properties-elements">
    <empty/>
</define>
```

14.13.2 Drawing Page Style

A drawing page style is a `<style:style>` element with family `drawing-page`. Within graphical applications, drawing page styles can be used to change the background of draw page. If a background is set with the help of a drawing page style, then it overrides the background of the master page that is assigned to the draw page, but not the shapes that are on the master page.

Within presentation applications, the draw page style additionally may contain presentation properties, for example, the duration for which a page is displayed or fade effects.

The properties that can be used in a draw page style to change the background are the ones described in section 15.14.

The presentation properties that can be used in a draw page style are described in section 15.36.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>drawing-page</value>
        </attribute>
        <optional>
            <ref name="style-drawing-page-properties"/>
        </optional>
    </group>
</define>

<define name="style-drawing-page-properties">
    <element name="style:drawing-page-properties">
        <ref name="style-drawing-page-properties-content"/>
    </element>
</define>

<define name="style-drawing-page-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-drawing-page-properties-content-strict">
    <ref name="style-graphic-fill-properties-attlist"/>
    <ref name="style-drawing-page-properties-attlist"/>
    <ref name="style-drawing-page-properties-elements"/>
</define>
```

14.14 Enhanced Graphic Style Elements

The elements described in this section are enhanced graphic style. They cannot be used as automatic styles, that is, they have to be located in the `<office:styles>` section of a document. Like all other style elements, they are referenced to by a unique name. The following styles for filling graphic objects are available:

- Gradient
- SVG Gradient
- Hatch
- Image
- Opacity Gradient
- Marker
- Dash
- Presentation Page Layout

14.14.1 Gradient

The element `<draw:gradient>` defines a gradient for filling a drawing object. Gradients are not available as automatic styles.

```
<define name="draw-gradient">
```

```
    <element name="draw:gradient">
        <ref name="common-draw-gradient-attlist"/>
        <ref name="draw-gradient-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the gradient element are:

- Name
- Display name
- Gradient style
- Gradient center
- Colors
- Intensity
- Angle
- Border

Name

The attribute `draw:name` uniquely identifies a gradient inside an `<office:styles>` element.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:name">
            <ref name="styleName"/>
        </attribute>
    </optional>
</define>
```

Display Name

The `draw:display-name` attribute specifies the name of the gradient as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Gradient Style

The attribute `draw:style` specifies the style of the gradient. The gradient styles that an office application should support are `linear`, `axial`, `radial`, `ellipsoid`, `square`, and `rectangular`.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <attribute name="draw:style">
        <ref name="gradient-style"/>
    </attribute>
```

```
</define>
<define name="gradient-style">
    <choice>
        <value>linear</value>
        <value>axial</value>
        <value>radial</value>
        <value>ellipsoid</value>
        <value>square</value>
        <value>rectangular</value>
    </choice>
</define>
```

Gradient Center

If the gradient style is `radial`, `ellipsoid`, `square`, or `rectangular`, the gradient center attributes `draw:cx` and `draw:cy` specifies the center of the geometry that is used for the gradient. The values of these attributes are always percentage values.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:cx">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:cy">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Colors

The gradient interpolates between a start color and an end color, which are specified using the attributes `draw:start-color` and `draw:end-color`.

```
<define name="draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-color">
            <ref name="color"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:end-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

Intensity

The attributes `draw:start-intensity` and `draw:end-intensity` specify the intensity of the gradient's start and end color as percentage values. These attributes are optional. If the attributes are not specified, the colors are used as they are, that is at 100% intensity.

```
<define name="draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-intensity">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:end-intensity">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

Angle

The `draw:angle` attribute specifies an angle that rotates the axis at which the gradient values are interpolated. This attribute is ignored for radial style gradients.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:angle">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

Border

Depending on the style of the gradient, the `draw:border` attribute specifies a percentage value which is used to scale a border which is filled by the start or end color only.

For example, a border of 10% means that the first 10% of the gradient is colored completely in the start color and the remaining 90% are an interpolation between start and end color.

```
<define name="common-draw-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:border">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

14.14.2 SVG Gradients

In addition to the gradients specified in section 14.14.1, gradient may be defined by the SVG gradient elements `<linearGradient>` and `<radialGradient>` as specified in §13.2 of [SVG]. The following rules apply to SVG gradients if they are used in documents in OpenDocument format:

- The gradients must get a name. It is specified by the `draw:name` attribute.
- For `<linearGradient>`, only the attributes `gradientTransform`, `x1`, `y1`, `x2`, `y2` and `spreadMethod` will be evaluated.
- For `<radialGradient>`, only the attributes `gradientTransform`, `cx`, `cy`, `r`, `fx`, `fy` and `spreadMethod` will be evaluated.
- The gradient will be calculated like having a `gradientUnits` of `objectBoundingBox`, regardless what the actual value of the attribute is.
- The only child element that is evaluated is `<stop>`.
- For `<stop>`, only the attributes `offset`, `stop-color` and `stop-opacity` will be evaluated.

```
<define name="svg-linearGradient">
    <element name="svg:linearGradient">
        <ref name="common-svg-gradient-attlist"/>
        <optional>
            <attribute name="svg:x1" a:defaultValue="0%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:y1" a:defaultValue="0%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:x2" a:defaultValue="100%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:y2" a:defaultValue="100%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="svg-stop"/>
        </zeroOrMore>
    </element>
</define>

<define name="svg-radialGradient">
    <element name="svg:radialGradient">
        <ref name="common-svg-gradient-attlist"/>
        <optional>
            <attribute name="svg:cx" a:defaultValue="50%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:cy" a:defaultValue="50%">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:r" a:defaultValue="50%">
```

```
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:fx">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:fy">
                <choice>
                    <ref name="coordinate"/>
                    <ref name="percent"/>
                </choice>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="svg-stop"/>
        </zeroOrMore>
    </element>
</define>

<define name="svg-stop">
    <element name="svg:stop">
        <attribute name="svg:offset">
            <choice>
                <ref name="double"/>
                <ref name="percent"/>
            </choice>
        </attribute>
        <optional>
            <attribute name="svg:stop-color">
                <ref name="color"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="svg:stop-opacity">
                <ref name="double"/>
            </attribute>
        </optional>
    </element>
</define>

<define name="common-svg-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="svg:gradientUnits" a:defaultValue="objectBoundingBox">
            <value>objectBoundingBox</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:gradientTransform">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:spreadMethod" a:defaultValue="pad">
```

```
            <choice>
                <value>pad</value>
                <value>reflect</value>
                <value>repeat</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Name

The attribute `draw:name` uniquely identifies a gradient inside an `<office:styles>` element. Like `<draw:gradient>` elements, SVG gradients are referenced by this name using the draw:fill-gradient-name attribute within a graphic style. SVG gradients cannot be referenced by a `draw:opacity-name attribute`. The result of referencing a SVG gradient with draw:fill-gradient-name attribute and an opacity gradient with a draw:opacity-name attribute at the same time is unspecified.

```
<define name="common-svg-gradient-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `draw:display-name` attribute specifies the name of the gradient as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="common-svg-gradient-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

14.14.3 Hatch

The `<draw:hatch>` element defines a hatch for filling graphic objects. A hatch is a simple pattern of straight lines that is repeated in the fill area. Hatches are not available as automatic styles.

```
<define name="draw-hatch">
    <element name="draw:hatch">
        <ref name="draw-hatch-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the hatch element are:

- Name
- Display name
- Style
- Color

- Distance
- Angle
- Background

Name

The `draw:name` attribute uniquely identifies a hatch inside an `<office:styles>` element.

```
<define name="draw-hatch-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `draw:display-name` attribute specifies the name of the hatch style as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="draw-hatch-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Style

The `draw:style` attribute specifies the style of the hatch.

The hatch can have one of three styles: `single`, `double`, or `triple`.

```
<define name="draw-hatch-attlist" combine="interleave">
    <attribute name="draw:style">
        <choice>
            <value>single</value>
            <value>double</value>
            <value>triple</value>
        </choice>
    </attribute>
</define>
```

Color

The `draw:color` attribute specifies the color of the hatch lines.

```
<define name="draw-hatch-attlist" combine="interleave">
    <optional>
        <attribute name="draw:color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

Distance

The `draw:distance` attribute specifies the distance between two hatch lines.

```
<define name="draw-hatch-attlist" combine="interleave">
    <optional>
        <attribute name="draw:distance">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Angle

The `draw:rotation` attribute specified the rotation angle of the hatch lines.

```
<define name="draw-hatch-attlist" combine="interleave">
    <optional>
        <attribute name="draw:rotation">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

14.14.4 Fill Image

The `<draw:fill-image>` element specifies a link to a bitmap resource, for example, a .PNG file. This element follows the XLink specification. Fill image are not available as automatic styles.

```
<define name="draw-fill-image">
    <element name="draw:fill-image">
        <ref name="draw-fill-image-attlist"/>
        <attribute name="xlink:href">
            <ref name="anyURI"/>
        </attribute>
        <optional>
            <attribute name="xlink:type" a:defaultValue="simple">
                <choice>
                    <value>simple</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:show" a:defaultValue="embed">
                <choice>
                    <value>embed</value>
                </choice>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate" a:defaultValue="onLoad">
                <choice>
                    <value>onLoad</value>
                </choice>
            </attribute>
        </optional>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the fill image element are:

- Name
- Display name
- Size

Name

The `draw:name` attribute uniquely identifies a fill image inside an `<office:styles>` element.

```
<define name="draw-fill-image-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The `draw:display-name` attribute specifies the name of the fill image as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="draw-fill-image-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Size

The optional attributes `svg:width` and `svg:height` specify the size of the linked image. 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.

```
<define name="draw-fill-image-attlist" combine="interleave">
    <optional>
        <attribute name="svg:width">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

14.14.5 Opacity Gradient

The `<draw:opacity>` element specifies an opacity gradient for a graphic object. An opacity gradient works like a gradient, except that the opacity is interpolated instead of the color. Opacity gradients are not available as automatic styles.

```
<define name="draw-opacity">
    <element name="draw:opacity">
        <ref name="common-draw-gradient-attlist"/>
        <ref name="draw-opacity-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<draw:opacity>` element are:

- Name, Display name, Style, Opacity center, Angle, Border – see section 14.14.1.
- Opacity

Opacity

The opacity interpolates between a start and an end value.

The values of the attributes `draw:start` and `draw:end` are percentages where 0% is fully transparent and 100% is fully opaque.

```
<define name="draw-opacity-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:end">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

14.14.6 Marker

The element `<draw:marker>` represents a marker, which is used to draw polygons at the start and end points of strokes. Markers are not available as automatic styles.

```
<define name="draw-marker">
    <element name="draw:marker">
        <ref name="draw-marker-attlist"/>
        <ref name="common-draw-viewbox-attlist"/>
        <ref name="common-draw-path-data-attlist"/>
        <empty/>
    </element>
</define>
```

See sections 9.2.4 and 9.2.15 for information on the path data and viewbox attributes that may be associated with the `<draw:marker>` element.

Name

The `draw:name` attribute uniquely identifies a fill image inside an `<office:styles>` element.

```
<define name="draw-marker-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The draw:display-name attribute specifies the name of the marker as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="draw-marker-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

14.14.7 Stroke Dash

The dash element <draw:stroke-dash> represents a dash style that can be used to render strokes of shapes. Stroke dashes are not available as automatic styles.

```
<define name="draw-stroke-dash">
    <element name="draw:stroke-dash">
        <ref name="draw-stroke-dash-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the <draw:stroke-dash> element are:

- Name
- Display name
- Style
- Dots
- Distance

Name

The attribute draw:name uniquely identifies a dash inside an <office:styles> element.

```
<define name="draw-stroke-dash-attlist" combine="interleave">
    <attribute name="draw:name">
        <ref name="styleName"/>
    </attribute>
</define>
```

Display Name

The draw:display-name attribute specifies the name of the dash as it should appear in the user interface. In contrast to the style name itself, this name may contain arbitrary characters. If this attribute is not present, the display name equals the style name.

```
<define name="draw-stroke-dash-attlist" combine="interleave">
    <optional>
        <attribute name="draw:display-name">
            <ref name="string"/>
        </attribute>
    </optional>
```

```
</define>
```

Style

The attribute `draw:style` specifies whether the points of a dash are round or rectangular.

```
<define name="draw-stroke-dash-attlist" combine="interleave">
    <optional>
        <attribute name="draw:style">
            <choice>
                <value>rect</value>
                <value>round</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Dots

The attribute pairs `draw:dots1`, `draw:dots1-length` and `draw:dots2`, `draw:dots2-length` each define a repeating sequence of dots that are used to render a dash. Both sequences are used alternating. The `draw:dots1` and `draw:dots2` attributes specify the number of dots to draw for both sequences, and the `draw:dots1-length` and `draw:dots2-length` attributes specify the length of each dot.

```
<define name="draw-stroke-dash-attlist" combine="interleave">
    <optional>
        <attribute name="draw:dots1">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:dots1-length">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:dots2">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:dots2-length">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Distance

The `draw:distance` attribute specifies the distance between the dots of a dash.

```
<define name="draw-stroke-dash-attlist" combine="interleave">
    <optional>
        <attribute name="draw:distance">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

14.15 Presentation Page Layouts

The element `<style: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.

The `<style:presentation-page-layout>` element has an attribute `style:name`. It defines the name of the page layout. If a drawing page has been created using a presentation page layout, the name of the layout is contained in the draw page's `presentation:presentation-page-layout-name` attribute. The optional `style:display-name` attribute specifies the name of the presentation page layout as it should appear in the user interface.

```
<define name="style-presentation-page-layout">
    <element name="style:presentation-page-layout">
        <attribute name="style:name">
            <ref name="styleName"/>
        </attribute>
        <optional>
            <attribute name="style:display-name">
                <ref name="string"/>
            </attribute>
        </optional>
        <zeroOrMore>
            <ref name="presentation-placeholder"/>
        </zeroOrMore>
    </element>
</define>
```

14.15.1 Presentation Placeholder

The element `<presentation:placeholder>` specifies a placeholder for presentation objects, for example, a title or outline.

The element has the following attributes:

- `object`: Specifies the kind of object the element is a placeholder for. The value equals the one of the presentation:class attribute for presentation shapes. See section 9.6.
- `svg:x`, `svg:y`, `svg:width`, `svg:height`: position and size attributes as specified in section 9.2.15, with the exception that the attributes may take percentage values in addition to coordinates and lengths.

```
<define name="presentation-placeholder">
    <element name="presentation:placeholder">
        <attribute name="presentation:object">
            <ref name="presentation-classes"/>
        </attribute>
        <attribute name="svg:x">
            <choice>
                <ref name="coordinate"/>
                <ref name="percent"/>
            </choice>
        </attribute>
        <attribute name="svg:y">
            <choice>
                <ref name="coordinate"/>
                <ref name="percent"/>
            </choice>
        </attribute>
        <attribute name="svg:width">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
        <attribute name="svg:height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
        <empty/>
    </element>
</define>
```

14.16 Chart Styles

Chart styles are `<style:style>` elements that have the family `chart`. They can be used within chart documents to specify formatting properties for the chart, but also for certain objects within a chart. They support the chart properties described in section 15.29, but also graphic, paragraph and text properties as described in sections 15.17, 15.5 and 15.4.

```
<define name="style-style-content" combine="choice">
    <group>
        <attribute name="style:family">
            <value>chart</value>
        </attribute>
        <optional>
            <ref name="style-chart-properties"/>
        </optional>
        <optional>
            <ref name="style-graphic-properties"/>
        </optional>
        <optional>
            <ref name="style-paragraph-properties"/>
        </optional>
        <optional>
            <ref name="style-text-properties"/>
        </optional>
    </group>
</define>
```

15 Formatting Properties

A document can contain several style elements. To acquire a common set of formatting properties, all formatting properties are contained in formatting property elements which are included as a child elements of any style element. This container elements offers two important advantages, as follows:

- Formatting properties can be addressed by [CSS2] or [XSLT] 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.

The following formatting property elements do exist:

- `<style:page-layout-properties>` for page layout properties
- `<style:header-footer-properties>` for page header and footer properties
- `<style:text-properties>` for text properties
- `<style:paragraph-properties>` for paragraph properties.
- `<style:section-properties>` for text section properties.
- `<style:ruby-properties>` for ruby section properties.
- `<style:list-level-properties>` for list properties.
- `<style:table-properties>` for table properties.
- `<style:table-column-properties>` for table column properties.
- `<style:table-row-properties>` for table row properties.
- `<style:table-cell-properties>` for table cell properties.
- `<style:graphic-properties>` for drawing object properties.

15.1 Simple and Complex Formatting Properties

15.1.1 Simple Formatting Properties

Most formatting properties are simple and can be represented as attributes of the formatting property elements. Where possible, [XSL] attributes or attributes from other specifications 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 namespace.

In office application, there are very often formatting properties that cannot be specified independent of other formatting properties. If this is the case, and if some of the required properties are missing, the application assumes reasonable default values.

Example: Simple style properties

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

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

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

15.1.3 Processing Rules for Formatting Properties

In the OpenDocument schema the various `<style:*-properties>` elements may contain pre-defined formatting attributes and elements as well as custom formatting attributes and elements. The pre-defined attributes and elements have defined semantics, and are described within this chapter.

Custom formatting attributes and elements are arbitrary attributes and elements inside `<style:*-properties>` elements. Their semantics are not defined in this specification,

Conforming applications in general **should** preserve both, pre-defined and custom formatting attributes and elements when editing the document.

```
<define name="style-properties-content">
    <ref name="anyAttListOrElements"/>
</define>
```

15.2 Page Layout Formatting Properties

The properties described in this section can be contained within style page layouts (see section 14.3) They are contained in a `<style:page-layout-properties>` element.

- 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
- Table centering
- Maximum footnote height
- Footnote separator

```
<define name="style-page-layout-properties">
    <element name="style:page-layout-properties">
        <ref name="style-page-layout-properties-content"/>
    </element>
</define>

<define name="style-page-layout-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-page-layout-properties-content-strict">
    <ref name="style-page-layout-properties-attlist"/>
    <ref name="style-page-layout-properties-elements"/>
</define>
```

15.2.1 Page Size

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

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.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:page-width">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:page-height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.2.2 Page Number Format

The `style:num-format`, `style:num-prefix` and `style:num-suffix` attributes specify a default number format for page styles, which is used to display page numbers within headers and footers. See section 12.2 for detailed information on number format attributes.

The `style:num-format` attribute can be empty. In this case, no page number will be displayed by default.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <ref name="common-num-format-attlist"/>
    </optional>
    <ref name="common-num-format-prefix-suffix-attlist"/>
</define>
```

15.2.3 Paper Tray

The `style:paper-tray-name` attribute specifies the paper tray to use when printing the document. The names assigned to the printer trays depend on the printer. If the value of this attribute is `default`, the default tray specified in the printer configuration settings is used.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:paper-tray-name">
            <choice>
                <value>default</value>
                <ref name="string"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.4 Print Orientation

The `style:print-orientation` attribute specifies the orientation of the printed page. The value of this attribute can be `portrait` or `landscape`.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:print-orientation">
            <choice>
                <value>portrait</value>
                <value>landscape</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.5 Margins

The margins attributes `fo:margin`, `fo:margin-top`, `fo:margin-bottom`, `fo:margin-left` and `fo:margin-right` specify the size of the page margins. See sections 15.5.17, 15.5.20 and 15.5.21 for detailed information on these attributes. Percentage values are not supported.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
    <ref name="common-vertical-margin-attlist"/>
    <ref name="common-margin-attlist"/>
</define>
```

15.2.6 Border

The border attributes `fo:border`, `fo:border-top`, `fo:border-bottom`, `fo:border-left` and `fo:border-right` specify the border properties of the page. See section 15.5.25 for detailed information on these attributes.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-border-attlist"/>
</define>
```

15.2.7 Border Line Width

If a page contains borders, the border line width attributes `style:border-line-width`, `style:border-line-width-top`, `style:border-line-width-bottom`, `style:border-line-width-left` and `style:border-line-width-right` specify the properties of the border lines of the page. See section 15.5.26 for detailed information on these attributes.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-border-line-width-attlist"/>
</define>
```

15.2.8 Padding

The padding attributes `fo:padding`, `fo:padding-top`, `fo:padding-bottom`, `fo:padding-left` and `fo:padding-right` specify the padding properties of the page. See section 15.5.27 for detailed information on these attributes.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-padding-attlist"/>
</define>
```

15.2.9 Shadow

The shadow attribute `style:shadow` specifies the shadow of the page. See section 15.5.28 for detailed information on this attribute.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>
```

15.2.10 Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the page. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-page-layout-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.2.11 Columns

The <style:columns> element specifies if the page contains columns. See section 15.7.3 for detailed information on this element.

```
<define name="style-page-layout-properties-elements" combine="interleave">
    <ref name="style-columns"/>
</define>
```

15.2.12 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.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:register-truth-ref-style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.2.13 Print

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

The value of this attribute is a list of the following values separated by blanks:

- headers
- grid
- annotations
- objects (including graphics)
- charts
- drawings
- formulas
- zero-values

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:print">
            <list>
                <zeroOrMore>
                    <choice>
                        <value>headers</value>
                        <value>grid</value>
                        <value>annotations</value>
                        <value>objects</value>
                        <value>charts</value>
                        <value>drawings</value>
                        <value>formulas</value>
                        <value>zero-values</value>
                    </choice>
                </zeroOrMore>
            </list>
        </attribute>
    </optional>
</define>
```

15.2.14 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.

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.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:print-page-order">
            <choice>
                <value>ttb</value>
                <value>ltr</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.15 First Page Number

The `style:first-page-number` specifies the number of the first page of a text or graphical document, or for the first page of a table within a spreadsheet document.

The value of this attribute can be an integer or `continue`. If the value is `continue`, the page number is the preceding page number incremented by 1. The default first page number is 1.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:first-page-number">
            <choice>
                <ref name="positiveInteger"/>
                <value>continue</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.16 Scale

The `style:scale-to` and `style:scale-to-pages` attributes specify how the application should scale spreadsheet documents for printing.

The `style:scale-to` attribute specifies that the document is scaled to a percentage value, where 100% equals no scaling. When using this attribute, all pages are enlarged or reduced in size while printing.

The `style:scale-to-pages` attribute specifies the number of pages on which the the document should be printed. The document is then scaled to fit the defined number of pages.

If none of these attributes are present, the document is not scaled.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:scale-to">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:scale-to-pages">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.2.17 Table Centering

The `style:table-centering` attribute specifies how the application should center tables on the page. This attribute only applies to spreadsheet documents.

The value of this attribute can be `horizontal`, `vertical`, `both`, or `none`. If this attribute is not present, the table is not centered.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:table-centering">
            <choice>
                <value>horizontal</value>
                <value>vertical</value>
                <value>both</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.18 Maximum Footnote Height

The `style:footnote-max-height` attribute specifies the maximum amount of space on the page that a footnote can occupy. The value of the attribute is a length, which determines the maximum height of the footnote area.

If the value of this attribute is set to `0in`, there is no limit to the amount of space that the footnote can occupy.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:footnote-max-height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.2.19 Writing Mode

The `style:writing mode` attribute specifies the writing mode that should is used by all paragraphs that appear on the page. See section 15.5.36 for details. The value `page` is not allowed within page layouts.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <ref name="common-writing-mode-attlist"/>
</define>
```

15.2.20 Footnote Separator

The `<style:footnote-sep>` element describes the line that separates the footnote area from the body text area on a page.

The `<style:footnote-sep>` element supports the following attributes:

- `style:width` – specifies the width or thickness of the line.
- `style:rel-width` – specifies the length of the line as a percentage of the body text area.
- `style:color` – specifies the color of the line.
- `style:adjustment` – specifies how the line is aligned on the page, that is left, right, or center.
- `style:distance-before-sep` – specifies the space between the body text area and the footnote line.
- `style:distance-after-sep` – specifies the space between the footnote line and the footnote text.
- `style:line-style` – specifies the style of the line.

```
<define name="style-page-layout-properties-elements" combine="interleave">
    <ref name="style-footnote-sep"/>
</define>

<define name="style-footnote-sep">
    <optional>
        <element name="style:footnote-sep">
            <ref name="style-footnote-sep-attlist"/>
            <empty/>
        </element>
    </optional>
</define>
<define name="style-footnote-sep-attlist" combine="interleave">
    <optional>
        <attribute name="style:width">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:rel-width">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:color">
            <ref name="color"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:line-style">
            <ref name="lineStyle"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:adjustment" a:defaultValue="left">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:distance-before-sep">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:distance-after-sep">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.2.21 Layout Grid

The `style:layout-grid-mode` property enables Asian layout grids. It has the following values:

- `none`: Disables the layout grid.
- `lines`: Enables a line layout, this is, the page is divided in a fixed number of lines. The exact number of lines depends on the other grid layout properties described below. There is no space between the layout grid lines. The layout grid itself is centered on the page.
- `both`: Like `lines`, except that the lines are divided into square cells. The number of cells per line depends on the line height, where the line height is the sum of the base height and the ruby height as specified below. Within a layout cell, nor more than one Asian [UNICODE] character is displayed. Asian characters that do not fit into a single cell are displayed centered into as many cells as required. Non Asian text is centered within as many cells as required.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-mode">
            <choice>
                <value>none</value>
                <value>line</value>
                <value>both</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.2.22 Layout Grid Base Height

The `style:layout-grid-base-height` attribute specifies the height reserved in the layout grid lines for non ruby text.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-base-height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.2.23 Layout Grid Ruby Height

The `style:layout-grid-ruby-height` attribute specifies the height reserved in the layout grid lines for ruby text.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-ruby-height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.2.24 Layout Grid Lines

The `style:layout-grid-lines` attribute specifies the number of layout grid lines per page. The number of lines actually displayed may be smaller than specified if the page has not enough space to display the specified number of lines with the specified line height (i.e., the sum of the base and ruby height).

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-lines">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.2.25 Layout Grid Color

The `style:layout-grid-color` attribute specifies the color of the layout grid border lines.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.2.26 Layout Grid Ruby Below

The `style:layout-grid-ruby-below` attribute specifies whether ruby text is displayed above or below the base text.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-ruby-below">
            <ref name="boolean"/>
        </attribute>
    </optional>
```

```
</define>
```

15.2.27 Layout Grid Print

The `style:layout-grid-ruby-print` attribute specifies whether the layout grid border lines are printed.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-print">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.2.28 Layout Grid Display

The `style:layout-grid-ruby-print` attribute specifies whether the layout grid border lines are displayed.

```
<define name="style-page-layout-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:layout-grid-display">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.3 Header Footer Formatting Properties

The properties described in this section can be contained within the header and footer style elements contained in page layouts (see section 14.3) They are contained in a `<style:header-footer-properties>` element.

These attributes are:

- Fixed and minimum heights - see section 15.27
- Left and right margins - see section 15.5.17
- Bottom (for headers only) and top (for footers only) margins - see section 15.5.20.
- Borders - see section 15.5.25 and 15.5.26
- Shadows – see section 15.5.28
- Backgrounds – see section 15.5.23 and 15.5.24.
- Dynamic-Spacing

```
<define name="style-header-footer-properties">
    <element name="style:header-footer-properties">
        <ref name="style-header-footer-properties-content"/>
    </element>
</define>

<define name="style-header-footer-properties-content">
    <ref name="style-properties-content"/>
</define>

```

```
<define name="style-header-footer-properties-content-strict">
        <ref name="style-header-footer-properties-attlist"/>
        <ref name="style-header-footer-properties-elements"/>
</define>
```

15.3.1 Fixed and Minimum heights

The attributes `svg:height` and `fo:min-height` properties specify a fixed or a minimum height for the header or footer.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <optional>
        <attribute name="svg:height">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:min-height">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.3.2 Margins

The margins attributes `fo:margin`, `fo:margin-top`, `fo:margin-bottom`, `fo:margin-left` and `fo:margin-right` specify the size of the header and footer margins. See sections 15.5.17, 15.5.20 and 15.5.21 for detailed information on these attributes. Percentage values are not supported. Bottom margins are only supported for headers, top margins only for footers.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
    <ref name="common-vertical-margin-attlist"/>
    <ref name="common-margin-attlist"/>
</define>
```

15.3.3 Border

The border attributes `fo:border`, `fo:border-top`, `fo:border-bottom`, `fo:border-left` and `fo:border-right` specify the border properties of the headers and footers. See section 15.5.25 for detailed information on these attributes.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-border-attlist"/>
</define>
```

15.3.4 Border Line Width

If a page contains borders, the border line width attributes `style:border-line-width`, `style:border-line-width-top`, `style:border-line-width-bottom`, `style:border-line-width-left` and `style:border-line-width-right` specify the properties of the border lines of the headers and footers. See section 15.5.26 for detailed information on these attributes.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-border-line-width-attlist"/>
</define>
```

15.3.5 Padding

The padding attributes `fo:padding`, `fo:padding-top`, `fo:padding-bottom`, `fo:padding-left` and `fo:padding-right` specify the padding properties of the headers and footers. See section 15.5.27 for detailed information on these attributes.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-padding-attlist"/>
</define>
```

15.3.6 Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the header or footer. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-header-footer-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.3.7 Shadow

The shadow attribute `style:shadow` specifies the shadow of the headers and footers. See section 15.5.28 for detailed information on this attribute.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>
```

15.3.8 Dynamic Spacing

The `style:dynamic-spacing` property specifies whether or not the header or footer grows into the space between the page body and the header or footer before the height of the page body becomes smaller. If the value of this attribute is `true`, the header or footers first grows into the space between the header and footer and the page body.

```
<define name="style-header-footer-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:dynamic-spacing">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4 Text Formatting Properties

The properties described in this section can be contained within text styles (see section 14.8.1), but also within other styles, like paragraph styles (see section 14.8.2) or cell styles (see section 14.12.4) They are contained in a `<style:text-properties>` element.

```
<define name="style-text-properties">
    <element name="style:text-properties">
        <ref name="style-text-properties-content"/>
    </element>
</define>

<define name="style-text-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-text-properties-content-strict">
    <ref name="style-text-properties-attlist"/>
    <ref name="style-text-properties-elements"/>
</define>

<define name="style-text-properties-elements">
    <empty/>
</define>
```

15.4.1 Font Variant

Use the `fo:font-variant` property to switch the option to display text as small capitalized letters on or off. See §7.8.8 of [XSL] for details.

For some implementations, the `fo:font-variant` and `fo:text-transform` properties are mutually exclusive. If both properties are used simultaneously, the result is undefined except that the `fo:text-transform` value is `none` and the `fo:font-variant` value is `normal`.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:font-variant">
            <ref name="fontVariant"/>
        </attribute>
    </optional>
</define>

<define name="fontVariant">
    <choice>
        <value>normal</value>
        <value>small-caps</value>
    </choice>
</define>
```

15.4.2 Text Transformations

Use the `fo:text-transform` property to describe text transformations to uppercase, lowercase, and capitalization. See §7.16.6 of [XSL] for details.

For some implementations, the `fo:font-variant` and `fo:text-transform` properties are mutually exclusive. If both properties are attached used simultaneously, the result is undefined except that the `fo:text-transform` value is `none` and the `fo:font-variant` value is `normal`.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:text-transform">
            <choice>
                <value>none</value>
                <value>lowercase</value>
                <value>uppercase</value>
                <value>capitalize</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.3 Color

Use the `fo:color` property to specify the foreground color of text. See §7.17.1 of [XSL] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
            <attribute name="fo:color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.4.4 Window Font Color

Use the `style:use-window-font-color` property to specify whether or not the window foreground color should be as used as the foreground color for a light background color and white for a dark background color.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:use-window-font-color">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4.5 Text Outline

Use the `style:text-outline` property to specify whether to display an outline of text or the text itself. This attribute can have a value of `true` or `false`.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-outline">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4.6 Line-Through Type

Use the `style:text-line-through-type` property to specify whether text is lined through, and if so, whether a single or double line will be used. See section 15.4.28 for details.

```
<define name="style-text-properties-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="style:text-line-through-type">
            <ref name="lineType"/>
        </attribute>
    </optional>
</define>
```

15.4.7 Line-Through Style

Use the `style:text-line-through-style` property to specify if and how text is lined through. This property is similar to the [CSS3Text] `text-line-style` property, except that it has the additional value `long-dash` and that it does not have the value `double`. Instead of this, the attribute `style:text:line-through-type` can be used to turn each line style into a double line. See §9.2 of [CSS3Text] for details. See also section 15.4.29.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-style">
            <ref name="lineStyle"/>
        </attribute>
    </optional>
</define>
```

15.4.8 Line-Through Width

Use the `style:text-line-through-width` property to specifies the width of a line-through line. This property is very similar to the [CSS3Text] `text-line-through-width` property, except that it has an additional value `bold`. `bold` specifies a line width that is calculated from the font sizes like an `auto` width, but is wider than an `auto` width. See §9.3 of [CSS3Text] for details. See also section 15.4.30.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-width">
            <ref name="lineWidth"/>
        </attribute>
    </optional>
</define>
```

15.4.9 Line-Through Color

Use the `style:text-line-through-color` property to specify the color that is used to line-through text. 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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-color">
            <choice>
                <value>font-color</value>
                <ref name="color"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.10 Line-Through Text

The `style:text-line-through-text` attribute is evaluated only if the value of `style:text-line-through-style` attribute is different than `none`. If the attribute value is not empty, the attribute value string is used for line-through instead of the line that has been specified, provided that the application supports line-through with text. If the application does not support line-through with text, the attribute is ignored, this means, `style:text-line-through-style` will be evaluated only. If the application supports line-through with single characters only, and the text-line-through-text has more than one character, the first character of the line-through-text should be used only. If the applications supports line-through with with certain characters only (like "x" or "/"), the application should use one of these characters if the text-line-through-text specifies characters that are not supported. In other words: line-through with text has a higher priority than line-through with lines, even if the line-through text that is specified has to be adapted to be usable by the application.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-text">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.4.11 Line-Through Text Style

The `style:text-line-through-text-style` specifies a text style that is applied to the text-line-through characters. It is not applied to line-through lines. If the attribute appears in an automatic style, it may reference either an automatic text style or a common style. If the attribute appears in a common style, it may reference a common style only.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-text-style">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.4.12 Text Position

Use the `style:text-position` 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.

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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="style:text-position">
            <list>
                <choice>
                    <ref name="percent"/>
                    <value>super</value>
                    <value>sub</value>
                </choice>
                <optional>
                    <ref name="percent"/>
                </optional>
            </list>
        </attribute>
    </optional>
</define>
```

15.4.13 Font Name

Use the `style:font-name`, `style:font-name-asian` and `style:font-name-complex` properties to assign a font to the text.

The values of these attributes form the name of a font that is declared by a `<style:font-face>` element within the `<office:font-face-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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-name">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-name-asian">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-name-complex">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.4.14 Font Family

Use the `fo:font-family`, `style:font-family-asian` and `style:font-family-complex` properties to specify the font family for the text.

These attributes may be used instead of the font name attributes to specify the properties of a font individually. However, it is advisable to use the `style:font-name` attributes instead. See section 15.4.13 for information about when Asian and complex variants of the attribute are evaluated. See also §7.8.2 of [XSL].

```
<define name="style-text-properties-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="fo:font-family">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-family-asian">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-family-complex">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.4.15 Font Family Generic

Use the `style:font-family-generic`, `style:font-family-generic-asian` and `style:font-family-generic-complex` properties to specify a generic font family name.

These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element.

Although it is recommended to use the font name attributes (see section 15.4.13), these properties may be used instead of them to specify the properties of a font.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-family-generic">
            <ref name="fontFamilyGeneric"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-family-generic-asian">
            <ref name="fontFamilyGeneric"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-family-generic-complex">
            <ref name="fontFamilyGeneric"/>
        </attribute>
    </optional>
</define>

<define name="fontFamilyGeneric">
    <choice>
        <value>roman</value>
        <value>swiss</value>
        <value>modern</value>
        <value>decorative</value>
        <value>script</value>
        <value>system</value>
    </choice>
</define>
```

15.4.16 Font Style

Use the `style:font-style-name`, `style:font-style-name-asian` and `style:font-style-name-complex` properties to specify a font style name.

These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element.

Although it is recommended to use the font name attributes (see section 15.4.13), these properties may be used instead of them to specify the properties of a font.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-style-name">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-style-name-asian">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-style-name-complex">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.4.17 Font Pitch

Use the `style:font-pitch`, `style:font-pitch` and `style:font-pitch-complex` properties to specify whether a font has a fixed or variable width.

These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element.

Although it is recommended to use the font name attributes (see section 15.4.13), these properties may be used instead of them to specify the properties of a font.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-pitch">
            <ref name="fontPitch"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-pitch-asian">
            <ref name="fontPitch"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-pitch-complex">
            <ref name="fontPitch"/>
        </attribute>
    </optional>
</define>

<define name="fontPitch">
    <choice>
        <value>fixed</value>
        <value>variable</value>
    </choice>
</define>
```

15.4.18 Font Character Set

Use the `style:font-charset`, `style:font-charset-asian` and `style:font-charset-complex` properties to specify the character set of a font.

The value of these attributes can be `x-symbol` or the character encoding in the notation described in the §4.3.3 of [XML1.0]. 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.

Although it is recommended to use the font name attributes (see section 15.4.13), these properties may be used instead of them to specify the properties of a font.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-charset">
            <ref name="textEncoding"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-charset-asian">
            <ref name="textEncoding"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-charset-complex">
            <ref name="textEncoding"/>
        </attribute>
    </optional>
</define>

<define name="textEncoding">
    <data type="string">
        <param name="pattern">[A-Za-z][A-Za-z0-9._\-]*</param>
    </data>
</define>
```

15.4.19 Font Size

Use the `fo:font-size`, `style:font-size-asian` and `style:font-size-complex` properties to specify the size of font.

The value of these property is either an absolute length or a percentage as described in §8.8.4 of [XSL]. In contrast to XSL, percentage values can be used within common styles only and relates 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 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:font-size">
            <choice>
                <ref name="positiveLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-size-asian">
            <choice>
                <ref name="positiveLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-size-complex">
            <choice>
                <ref name="positiveLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.20 Relative Font Size

Use the `style:font-size-rel`, `style:font-size-rel-asian` and `style:font-size-rel-complex` properties to specify a relative font size change.

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 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-size-rel">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-size-rel-asian">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-size-rel-complex">
            <ref name="length"/>
        </attribute>
    </optional>
```

```
</define>
```

15.4.21 Script Type

The `style:script-type` property may be used to specify which script dependent attributes (like `fo:font-family`, `style:font-family-asian`, `style:font-family-complex`) are currently active for some text. The attribute should be evaluated by applications that do not support script types to select the correct script dependent properties. Application that support script types may also evaluate the attribute and overwrite the script type they would evaluate for a certain character, but they don't have to.

The usage of this property simplifies for instance transformations from and to [CSS2]/[XSL] and other formats that don't have script-dependent attributes, and also can be used to assign script-types to weak [UNICODE] characters, where application may choose different script types.

The values of this property are `latin`, `asian`, `complex` and `ignore`. The value `ignore` can be used only within default styles. If it is set, all script-dependent attributes are applied to all script types. This would mean for example that a `fo:font-family` would be applied to all script types as well as a `style:font-family-asian` or `style:font-family-complex`. This simplifies saving documents with application that do not support a script type.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:script-type">
            <choice>
                <value>latin</value>
                <value>asian</value>
                <value>complex</value>
                <value>ignore</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.22 Letter Spacing

Use the `fo:letter-spacing` property to specify the amount of space between letters. The value of this property can be `normal` or it can specify a length. See §7.16.2 of [XSL] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:letter-spacing">
            <choice>
                <ref name="length"/>
                <value>normal</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.23 Language

Use the `fo:language`, `fo:language-asian` and `fo:language-complex` properties to specify the language of the text. See §7.9.2 of [XSL] for details.

Some applications ignore these properties if they are not specified together with the corresponding `fo:country` property.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

`fo:language`, `fo:language-asian` and `fo:language-complex`

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:language">
            <ref name="languageCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:language-asian">
            <ref name="languageCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:language-complex">
            <ref name="languageCode"/>
        </attribute>
    </optional>
</define>
```

15.4.24 Country

Use the `fo:country`, `style:country-asian` and `style:country-complex` properties to specify the country of the text. See §7.9.1 of [XSL] for details.

Some application ignore these properties if they are not specified together with the corresponding `fo:language` property.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:country">
            <ref name="countryCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:country-asian">
            <ref name="countryCode"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:country-complex">
            <ref name="countryCode"/>
        </attribute>
    </optional>
</define>
```

15.4.25 Font Style

Use the `fo:font-style`, `style:font-style-asian` and `style:font-style-complex` properties to specify whether to use normal or italic font face. See §7.8.7 of [XSL] for details.

See section 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="fo:font-style">
            <ref name="fontStyle"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-style-asian">
            <ref name="fontStyle"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-style-complex">
            <ref name="fontStyle"/>
        </attribute>
    </optional>
</define>

<define name="fontStyle">
    <choice>
        <value>normal</value>
        <value>italic</value>
        <value>oblique</value>
    </choice>
</define>
```

15.4.26 Font Relief

Use the `style:font-relief` property to specify whether the font should be embossed, engraved, or neither.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-relief">
            <choice>
            <value>none</value>
                <value>embossed</value>
                <value>engraved</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.27 Text Shadow

Use the `fo:text-shadow` property to specify the text shadow style to use. See §7.16.5 of [XSL] for details.

Some applications may only supports a limited number of shadow effects, for instance a default text shadow style only.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:text-shadow">
            <ref name="shadowType"/>
        </attribute>
    </optional>
</define>

<define name="shadowType">
    <choice>
```

```
        <value>none</value>
        <!-- The following string must match an XSL shadow decl -->
        <ref name="string"/>
    </choice>
</define>
```

15.4.28 Underlining Type

Use the `style:text-underline-type` property to specify whether text is underlined, and if so, whether a single or double line will be used for underlining.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-underline-type">
            <ref name="lineType"/>
        </attribute>
    </optional>
</define>

<define name="lineType">
    <choice>
        <value>none</value>
        <value>single</value>
        <value>double</value>
    </choice>
</define>
```

15.4.29 Underlining Style

Use the `style:text-underline-style` property to specify if and how text is underlined. The value of this property is the underlining style for the text, for example, `single`, `dotted`, `dash`. This property is similar to the [CSS3Text] `text-underline-style` property, except that has the additional value `long-dash` and that it does not have the value `double`. Instead of this, the attribute `style:text:underline-type` can be used to turn each line style into a double line. See §9.2 of [CSS3Text] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-underline-style">
            <ref name="lineStyle"/>
        </attribute>
    </optional>
</define>

<define name="lineStyle">
    <choice>
        <value>none</value>
        <value>solid</value>
        <value>dotted</value>
        <value>dash</value>
        <value>long-dash</value>
        <value>dot-dash</value>
        <value>dot-dot-dash</value>
        <value>wave</value>
    </choice>
</define>
```

15.4.30 Underling Width

Use the `style:text-underline-width` property specifies the width of an underline. This property is very similar to the [CSS3Text] `text-underline-width` property, except that it has an additional value `bold`. `bold` specifies a line width that is calculated from the font sizes like an `auto` width, but is wider than an `auto` width. See §9.3 of [CSS3Text] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-underline-width">
            <ref name="lineWidth"/>
        </attribute>
    </optional>
</define>

<define name="lineWidth">
    <choice>
        <value>auto</value>
        <value>normal</value>
        <value>bold</value>
        <value>thin</value>
        <value>dash</value>
        <value>medium</value>
        <value>thick</value>
        <ref name="positiveInteger"/>
        <ref name="percent"/>
        <ref name="positiveLength"/>
    </choice>
</define>
```

15.4.31 Underline Color

Use the `style:text-underline-color` property to specify the color that is used to underline text. 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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-underline-color">
            <choice>
                <value>font-color</value>
                <ref name="color"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.32 Font Weight

Use the `fo:font-weight`, `style:font-weight-asian` and `style:font-weight-complex` properties to specify the weight of the font. See §7.8.9 of [XSL] for details.

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 15.4.13 for information about when the Asian and complex variants of the attribute are evaluated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="fo:font-weight">
            <ref name="fontWeight"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-weight-asian">
            <ref name="fontWeight"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:font-weight-complex">
            <ref name="fontWeight"/>
        </attribute>
    </optional>
</define>

<define name="fontWeight">
    <choice>
        <value>normal</value>
        <value>bold</value>
        <value>100</value>
        <value>200</value>
        <value>300</value>
        <value>400</value>
        <value>500</value>
        <value>600</value>
        <value>700</value>
        <value>800</value>
        <value>900</value>
    </choice>
</define>
```

15.4.33 Text Underline Word Mode

Use the `style:text-underline-mode` property to specify whether underlining is applied to words only or to portions of text. If underlining is applied to text portions, the spaces between words and the words are underlined. This property is very similar to the `text-underline-mode` property of [CSS3Text]. See § 9.5 of [CSS3Text] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-underline-mode">
            <ref name="lineMode"/>
        </attribute>
    </optional>
</define>

<define name="lineMode">
    <choice>
        <value>continuous</value>
        <value>skip-white-space</value>
    </choice>
</define>
```

15.4.34 Text Line-Through Word Mode

Use the `style:text-line-through-mode` property to specify whether lining through is applied to words only or to portions of text. If lining through is applied to text portions, the spaces between words and the words are line-through. This property is very similar to the `text-line-through-mode` property of [CSS3Text]. See § 9.5 of [CSS3Text] for details.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-line-through-mode">
            <ref name="lineMode"/>
        </attribute>
    </optional>
</define>
```

15.4.35 Letter Kerning

Use the `style:letter-kerning` property to enable or disable kerning between characters.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:letter-kerning">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4.36 Text Blinking

Use the `style:text-blinking` property to specify whether or not text blinks.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-blinking">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4.37 Text Background Color

Use the `fo:background-color` property to specify the background color to apply to characters. See §7.7.2 of [XSL] for details.

The value of this property can be `transparent` or a color. See also section 15.5.23.

```
<define name="style-text-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
```

15.4.38 Text Combine

Use the `style:text-combine` property to combine characters so that they are displayed within two lines.

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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="style:text-combine">
            <choice>
                <value>none</value>
                <value>letters</value>
                <value>lines</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.39 Text Combine Start and End Characters

Use the two properties `style:text-combine-start-char` and `style:text-combine-end-char` to 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`.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-combine-start-char">
            <ref name="character"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:text-combine-end-char">
            <ref name="character"/>
        </attribute>
    </optional>
</define>
```

15.4.40 Text Emphasis

Use the `style:text-emphasize` property to emphasize text in Asian documents.

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.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-emphasize">
            <choice>
                <value>none</value>
                <list>
                    <choice>
                        <value>none</value>
                        <value>accent</value>
                        <value>dot</value>
                        <value>circle</value>
                        <value>disc</value>
                    </choice>
                    <choice>
                        <value>above</value>
                        <value>below</value>
                    </choice>
                </list>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.41 Text Scale

Use the `style:text-scale` property to decrease or increase the width of the text by scaling the font width.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-scale">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.4.42 Text Rotation Angle

The `style:text-rotation-angle` property specifies an angle to which text is rotated. The value of this attribute can be `0`, `90`, or `270`. For any angle greater than 359 the remainder of a division by 360 is used. Any angle other than 0, 90 or 270 is rounded to the nearest possible value.

If this attribute is specified for more than one character, all text containing these characters is rotated.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-rotation-angle">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

15.4.43 Text Rotation Scale

If text is rotated, the `style:text-rotation-scale` property specifies whether the width of the text should be scaled to fit into the current line height or the width of the text should remain fixed, therefore changing the current line height.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-rotation-scale">
            <choice>
                <value>fixed</value>
                <value>line-height</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.4.44 Hyphenation

Use the `fo:hyphenate` property to enable or disable automatic hyphenation. See §7.9.4 of [XSL] for details.

Some application might not support setting the properties fo:hyphenate, fo:hyphenation-keep, fo:hyphenation-remain-char-count, fo:hyphenation-push-char-count and fo:hyphenation-ladder-count independent of each other within a style. A reasonable default for fo:hyphenate in this case is false.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:hyphenate">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.4.45 Hyphenation Remain Char Count

Use the fo:hyphenation-remain-char-count property to specify the number of characters that must be present before a hyphenation character. See §7.9.7 of [XSL] for details.

Some application might not support setting the properties fo:hyphenate, fo:hyphenation-keep, fo:hyphenation-remain-char-count, fo:hyphenation-push-char-count and fo:hyphenation-ladder-count independent of each other within a style. A reasonable default for fo:hyphenation-remain-char-count in this case is 0.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:hyphenation-remain-char-count">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.4.46 Hyphenation Push Char Count

Use the fo:hyphenation-push-char-count property to specify the minimum number of characters that are moved to the next line. See §7.9.6 of [XSL] for details.

Some application might not support setting the properties fo:hyphenate, fo:hyphenation-keep, fo:hyphenation-remain-char-count, fo:hyphenation-push-char-count and fo:hyphenation-ladder-count independent of each other within a style. A reasonable default for fo:hyphenation-push-char-count in this case is 0.

```
<define name="style-text-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:hyphenation-push-char-count">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.4.47 Hidden or Conditional Text

The text:display property allows text to be hidden. This can be made dependent on a condition as well. This attributes and its values are the same as for text:display attribute on text sections (see also section 4.4). The values of this attribute may be any of:

- true – the text will be displayed normally. This is the default.
- none – the text will be hidden.

- `condition` – a condition determines whether the text will be displayed or hidden. In this case, a `text:condition` attribute must be present specifying the condition.

```
<define name="style-text-properties-attlist" combine="interleave">
    <choice>
        <attribute name="text:display">
            <value>true</value>
        </attribute>
        <attribute name="text:display">
            <value>none</value>
        </attribute>
        <group>
            <attribute name="text:display">
                <value>condition</value>
            </attribute>
            <attribute name="text:condition">
                <value>none</value>
            </attribute>
        </group>
        <empty/>
    </choice>
</define>
```

15.5 Paragraph Formatting Properties

The properties described in this section can be contained within paragraph styles (see section 14.8.2), but also within other styles, like cell styles (see section 14.12.4) They are contained in a `<style:paragraph-properties>` element.

```
<define name="style-paragraph-properties">
    <element name="style:paragraph-properties">
        <ref name="style-paragraph-properties-content"/>
    </element>
</define>

<define name="style-paragraph-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-paragraph-properties-content-strict">
    <ref name="style-paragraph-properties-attlist"/>
    <ref name="style-paragraph-properties-elements"/>
</define>
```

15.5.1 Fixed Line Height

Use the `fo:line-height` 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. The value of this property can be a length, a percentage, or a value of `normal`. See §7.15.4 of [XSL] for details.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:line-height">
            <choice>
                <value>normal</value>
                <ref name="nonNegativeLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.2 Minimum Line Height

Use the `style:line-height-at-least` property to specify a minimum line height. The value of this property is a length. There is no `normal` value for the property.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:line-height-at-least">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.5.3 Line Distance

Use the `style:line-spacing` property to specify a fixed distance between two lines. There is no `normal` value for this property.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:line-spacing">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.5.4 Font-Independent Line Spacing

The `style:font-independent-line-spacing` property specifies if font independent line spacing is used. If the attribute's value is `true`, then the line height is calculated only from the font height as specified by the font size attributes `fo:font-size`, `style:font-size-asian` and `style:font-size-complex`. If the value is `false`, the font metric of the actual font is taken into account.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-independent-line-spacing">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.5 Text Align

Use the `fo:text-align` property to specify how to align text in paragraphs.

The value of this property can be `start`, `end`, `left`, `right`, `center`, or `justify`. See §7.15.9 of [XSL] for details. The values `inside` and `outside` are not supported.

If there are no values specified for the `fo:text-align-last` and `style:justify-single-word` properties within the same item set element, the values of these properties are set to `start` and `false` respectively.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
```

```
    <ref name="common-text-align"/>
</define>

<define name="common-text-align">
    <optional>
        <attribute name="fo:text-align">
            <choice>
                <value>start</value>
                <value>end</value>
                <value>left</value>
                <value>right</value>
                <value>center</value>
                <value>justify</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.6 Text Align of Last Line

Use the `fo:text-align-last` property to specify how to align the last line of a justified paragraph. See §7.15.9 of [XSL] for details. The only values of this property that are supported are `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 `start` and `false` respectively.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:text-align-last">
            <choice>
                <value>start</value>
                <value>center</value>
                <value>justify</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.7 Justify Single Word

If the last line in a paragraph is justified, use the `style:justify-single-word` property to specify whether or not a single word should be justified.

If there are no values specified for the `fo:text-align` and `fo:text-align-last` properties, the values of these properties are set to `start`. This means that specifying a `style:justify-single-word` property without specifying a `fo:text-align` and `fo:text-align-last` property has no effect.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:justify-single-word">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.8 Keep Together

Use the `fo:keep-together` property to control whether the lines of a paragraph should be kept together on the same page or column (if the value is `always`), or whether breaks are allowed within the paragraph (if the value is `auto`). See §7.19.3 of [XSL] for details.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:keep-together">
            <choice>
                <value>auto</value>
                <value>always</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.9 Widows

Use the `fo:widows` property to specify the minimum number of lines allowed at the top of a page to avoid paragraph **widows**. See §7.19.7 of [XSL] for details.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:widows">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.5.10 Orphans

Use the `fo:orphans` property to specify the minimum number of lines required at the bottom of a page to avoid paragraph **orphans**. See See §7.19.6 of [XSL] for details.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:orphans">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.5.11 Tab Stops

Use the tab stop element `<style:tab-stops>` to specify tab stop definitions.

Every tab stop position is represented by a single `<style:tab-stop>` element that is contained in the `<style:tab-stops>` element.

```
<define name="style-paragraph-properties-elements" combine="interleave">
    <ref name="style-tab-stops"/>
</define>

<define name="style-tab-stops">
    <optional>
        <element name="style:tab-stops">
            <zeroOrMore>
                <ref name="style-tab-stop"/>
            </zeroOrMore>
        </element>
    </optional>
</define>

<define name="style-tab-stop">
    <element name="style:tab-stop">
        <ref name="style-tab-stop-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes that may be associated with the `<style:tab-stop>` elements are:

- Tab position
- Tab type
- Delimiter character
- Leader type
- Leader style
- Leader width
- Leader color
- Leader text
- Leader text style

Tab Position

The `style:position` attribute specifies the position of a tab stop.

This attribute is associated with the `<style:tab-stop>` element and its value is a length.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <attribute name="style:position">
        <ref name="nonNegativeLength"/>
    </attribute>
</define>
```

Tab Type

The `style:type` attribute specifies the type of tab stop.

This attribute is associated with the `<style:tab-stop>` element and its value can be `left`, `center`, `right` or `char`.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <choice>
        <optional>
            <attribute name="style:type" a:defaultValue="left">
                <choice>
                    <value>left</value>
                    <value>center</value>
                    <value>right</value>
                </choice>
            </attribute>
        </optional>
        <group>
            <attribute name="style:type">
                <value>char</value>
            </attribute>
            <ref name="style-tab-stop-char-attlist"/>
        </group>
    </choice>
</define>
```

Delimiter Character

The `style:char` attribute specifies the delimiter character for tab stops of type `char`.

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.

```
<define name="style-tab-stop-char-attlist" combine="interleave">
    <attribute name="style:char">
        <ref name="character"/>
    </attribute>
</define>
```

Leader Type

Use the `style:leader-type` attribute to specify whether a leader line should be drawn, and if so, whether a single or double line will be used. See also section 15.4.28.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-type">
            <ref name="lineType"/>
        </attribute>
    </optional>
</define>
```

Leader Style

Use the `style:leader-style` property to specify if and how a leader line is drawn. The line styles that can be used are described in section 15.4.29.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-style">
            <ref name="lineStyle"/>
        </attribute>
    </optional>
</define>
```

Leader Width

Use the `style:leader-width` property to specifies the width of a leader line. See section 15.4.30 for the values of this attribute.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-width">
```

```
            <ref name="lineWidth"/>
        </attribute>
    </optional>
</define>
```

Leader Color

Use the `style:leader-color` property to specify the color that is for the leader line. 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 the leader line.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-color">
            <choice>
                <value>font-color</value>
                <ref name="color"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Leader Text

The `style:leader-text` attribute specifies the leader text to use for tab stops. If the attribute value is not empty, the attribute value string is used as leader instead of the line that has been specified, provided that the application supports textual leaders. If the application does not support textual, the attribute is ignored, this means, `style:leader-style` will be evaluated only. If the application supports textual consisting of a single characters only, and the leader text has more than one character, the first character of the leader text should be used only. If the applications supports textual leaders with with certain characters only (like "." or "_"), the application should use one of these characters if the leader-text specifies characters that are not supported. In other words: textual leaders have a higher priority than line leaders, even if the leader text that is specified has to be adapted to be usable by the application.

This attribute is associated with the `<style:tab-stop>` element and its value must be a single [UNICODE] character.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-text" a:defaultValue=" ">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Leader Text Style

The `style:leader-text-style` specifies a text style that is applied to a textual leader. It is not applied to leader lines. If the attribute appears in an automatic style, it may reference either an automatic text style or a common style. If the attribute appears in a common style, it may reference a common style only.

```
<define name="style-tab-stop-attlist" combine="interleave">
    <optional>
        <attribute name="style:leader-text-style">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
```

```
</define>
```

15.5.12 Tab Stop Distance

The attribute `style:tab-stop-distance` specifies the distance between default tab stops. A default tab stop is repeated automatically after the specified distance. Default tab stops usually are only evaluated if they are specified within a default style (see section 14.2).

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:tab-stop-distance">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.5.13 Hyphenation Keep

Use the `fo:hyphenation-keep` property to enable or disable the hyphenation of the last word on a page. See §7.15.1 of [XSL] for details.

Some application might not support setting the properties `fo:hyphenate`, `fo:hyphenation-keep`, `fo:hyphenation-remain-char-count`, `fo:hyphenation-push-char-count` and `fo:hyphenation-ladder-count` independent of each other within a style. A reasonable default for `fo:hyphenation-keep` in this case is `auto`.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:hyphenation-keep">
            <choice>
                <value>auto</value>
                <value>page</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.14 Maximum Hyphens

Use the `fo:hyphenation-ladder-count` property to specify the maximum number of successive lines that can contain a hyphenated word. See §7.15.2 of [XSL] for details.

Some application might not support setting the properties `fo:hyphenate`, `fo:hyphenation-keep`, `fo:hyphenation-remain-char-count`, `fo:hyphenation-push-char-count` and `fo:hyphenation-ladder-count` independent of each other within a style. A reasonable default for `fo:hyphenation-push-char-count` in this case is `no-limit`.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:hyphenation-ladder-count">
            <choice>
                <value>no-limit</value>
                <ref name="positiveInteger"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.15 Drop Caps

Use the `<style:drop-cap>` element to specify if the first character or more of a paragraph is displayed in a larger font. This element can be contained in a `<style:paragraph-properties>` element.

```
<define name="style-paragraph-properties-elements" combine="interleave">
    <ref name="style-drop-cap"/>
</define>

<define name="style-drop-cap">
    <optional>
        <element name="style:drop-cap">
            <ref name="style-drop-cap-attlist"/>
            <empty/>
        </element>
    </optional>
</define>
```

The attributes that may be associated 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.

The value of this attribute can be a number or `word`, which indicates that the first word should be dropped.

```
<define name="style-drop-cap-attlist" combine="interleave">
    <optional>
        <attribute name="style:length" a:defaultValue="1">
            <choice>
                <value>word</value>
                <ref name="positiveInteger"/>
            </choice>
        </attribute>
    </optional>
</define>
```

Lines

The `style:lines` attribute specifies the number of lines that the dropped characters should encircle. If the value of this attribute is `1` or `0`, drop caps is disabled.

```
<define name="style-drop-cap-attlist" combine="interleave">
    <optional>
        <attribute name="style:lines" a:defaultValue="1">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

Distance

The `style:distance` attribute specifies the distance between the last dropped character and the first of the remaining characters of each line. The value of this attribute is a length.

```
<define name="style-drop-cap-attlist" combine="interleave">
    <optional>
        <attribute name="style:distance" a:defaultValue="0cm">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

Text Style

The `style:style-name` attribute specifies the text style to apply to the dropped characters.

```
<define name="style-drop-cap-attlist" combine="interleave">
    <optional>
        <attribute name="style:style-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.5.16 Register True

The `style:register-true` property specifies whether the lines on both sides of a printed page match when a document is printed using two-sided printing, 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. See also section 15.2.12.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
        <optional>
            <attribute name="style:register-true">
                <ref name="boolean"/>
            </attribute>
        </optional>
    </define>
```

15.5.17 Left and Right Margins

Use the `fo:margin-left` and `fo:margin-right` properties to specify the left and right margins for a paragraph. See §7.10.3 and §7.10.4 of [XSL] for details. The attributes' values are lengths. If the attribute is contained in a common style, the attributes' values may be also percentages. They here relate to the corresponding margin of the parent style.

For some applications. these two properties must be used simultaneously and also together with the `fo:text-indent` property. If any of the properties is missing, its value is assumed to be 0cm.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
</define>

<define name="common-horizontal-margin-attlist">
    <optional>
        <attribute name="fo:margin-left">
            <choice>
```

```
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:margin-right">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.18 Text Indent

Use the `fo:text-indent` property to specify a positive or negative indent for the first line of a paragraph. See §7.15.11 of [XSL] for details. The attribute's value is a length. If the attribute is contained in a common style, the attribute's value may be also a percentage. It here relates to the corresponding margin of the parent style.

For some applications. the `fo:text-indent` property must be used 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.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:text-indent">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.19 Automatic Text Indent

Use the `style:auto-text-indent` property to specify that the first line of a paragraph is indented by a value that is based on the current font size.

For some applications. the `style:auto-text-indent` property must be used 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 has a value of `true` and is used together with a `fo:text-indent` property, then the `fo:text-indent` property is ignored.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:auto-text-indent">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.20 Top and Bottom Margins

Use the `fo:margin-top` and `fo:margin-bottom` properties to specify the top and bottom margins for paragraphs. See §7.10.1 and §7.10.2 of [XSL] for details. The attributes' values are lengths. If the attribute is contained in a common style, the attributes' values may be also percentages. They here relate to the corresponding margin of the parent style.

For some applications. these two properties must be used simultaneously. If any of the properties is missing, its value is assumed to be 0cm.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-vertical-margin-attlist"/>
</define>

<define name="common-vertical-margin-attlist">
    <optional>
        <attribute name="fo:margin-top">
            <choice>
                <ref name="nonNegativeLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:margin-bottom">
            <choice>
                <ref name="nonNegativeLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.21 Margins

Use the `fo:margin` property to specify the top, bottom, left and right margins for paragraphs simultaneously. See §7.29.4 of [XSL] and sections 15.5.17 and 15.5.20 for details.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-margin-attlist"/>
</define>

<define name="common-margin-attlist">
    <optional>
        <attribute name="fo:margin">
            <choice>
                <ref name="nonNegativeLength"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.22 Break Before and Break After

Use the `fo:break-before` and `fo:break-after` properties to insert a page or column break before or after a paragraph. See §7.19.1 and §7.19.2 of [XSL] for details. The values `odd-page` and `even-page` are not supported.

These two properties are mutually exclusive. If they are used simultaneously, the result is undefined.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-break-attlist"/>
</define>

<define name="common-break-attlist">
    <optional>
        <attribute name="fo:break-before">
            <choice>
                <value>auto</value>
                <value>column</value>
                <value>page</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:break-after">
            <choice>
                <value>auto</value>
                <value>column</value>
                <value>page</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.23 Paragraph Background Color

Use the `fo:background-color` property to specify the background color of a paragraph. See §7.7.2 of [XSL] for details.

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 simultaneously.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>

<define name="common-background-color-attlist">
    <optional>
        <attribute name="fo:background-color">
            <choice>
                <value>transparent</value>
                <ref name="color"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.24 Paragraph Background Image

Use the `<style:background-image>` element to specify a background image for a paragraph.

The background image can be stored in one of the following ways (see also section 9.3.2):

- The image data is stored in an external file. Use the [XLink] attributes to specify the location of the image.

- The image data is contained in an `<office:binary-data>` sub-element in BASE64 encoding.

If the `<style:background-image>` element is empty and if there is no color specified by an `fo:background-color` element in the same properties element, the background color is set to transparent.

```
<define name="style-paragraph-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>

<define name="style-background-image">
    <optional>
        <element name="style:background-image">
            <ref name="style-background-image-attlist"/>
            <choice>
                <ref name="common-draw-data-attlist"/>
                <ref name="office-binary-data"/>
                <empty/>
            </choice>
        </element>
    </optional>
</define>
```

The attributes that may be associated with the `<style:background-image>` element are:

- Repetition
- Position
- Filter
- Opacity

Repetition

The `style:repeat` attribute specifies whether a background image is repeated or stretched in a paragraph.

This attribute is attached to the `<style:background-image>` element and its value can be `no-repeat`, `repeat`, or `stretch`.

```
<define name="style-background-image-attlist" combine="interleave">
    <optional>
        <attribute name="style:repeat" a:defaultValue="repeat">
            <choice>
                <value>no-repeat</value>
                <value>repeat</value>
                <value>stretch</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Position

The `style:position` attribute specifies where to position a background image in a paragraph.

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. If one position is specified, the other position defaults to `center`.

```
<define name="style-background-image-attlist" combine="interleave">
    <optional>
        <attribute name="style:position" a:defaultValue="center">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>top</value>
                <value>bottom</value>
                <list>
                    <ref name="horiBackPos"/>
                    <ref name="vertBackPos"/>
                </list>
                <list>
                    <ref name="vertBackPos"/>
                    <ref name="horiBackPos"/>
                </list>
            </choice>
        </attribute>
    </optional>
</define>

<define name="horiBackPos">
    <choice>
        <value>left</value>
        <value>center</value>
        <value>right</value>
    </choice>
</define>
<define name="vertBackPos">
    <choice>
        <value>top</value>
        <value>center</value>
        <value>bottom</value>
    </choice>
</define>
```

Filter

The `style:filter-name` attribute specifies the application specific filter name that is used to load the image into the document.

This attribute is attached to the `<style:background-image>` element.

```
<define name="style-background-image-attlist" combine="interleave">
    <optional>
        <attribute name="style:filter-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Opacity

The `draw:opacity` attribute specifies the opacity of the background image. The value is a percentage, where 0% is fully transparent and 100% is fully opaque.

```
<define name="style-background-image-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="draw:opacity">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.5.25 Border

Use the border properties `fo:border`, `fo:border-top`, `fo:border-bottom`, `fo:border-left` and `fo:border-right` to specify the border properties for paragraphs. See §7.29.3 - §7.29.7 of [XSL] for details.

The `fo:border` property applies to all four sides of a paragraph while the other properties apply to one side only.

For some applications, 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 a properties element. In the latter case, if one or more of the properties is missing their values are assumed to be `none`.

There may be also restriction regarding the border styles and widths that are supported. In addition to this, some applications may add a default padding for sides that have a border.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-border-attlist"/>
</define>

<define name="common-border-attlist">
    <optional>
        <attribute name="fo:border">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:border-top">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:border-bottom">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:border-left">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:border-right">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.5.26 Border Line Width

If the line style for a border is `double`, use the border line properties `style:border-line-width`, `style:border-line-width-top`, `style:border-line-width-bottom`, style:border-line-width-left and style:border-line-width-right to individually specify the width of the inner and outer lines and the distance between them.

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

The result of specifying a border line width without specifying a border width style of double for the same border is undefined.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-border-line-width-attlist"/>
</define>

<define name="common-border-line-width-attlist">
    <optional>
        <attribute name="style:border-line-width">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:border-line-width-top">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:border-line-width-bottom">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:border-line-width-left">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:border-line-width-right">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
</define>

<define name="borderWidths">
    <list>
        <ref name="positiveLength"/>
        <ref name="positiveLength"/>
        <ref name="positiveLength"/>
    </list>
</define>
```

15.5.27 Padding

Use the padding properties fo:padding, fo:padding-top, fo:padding-bottom, fo:padding-left and fo:padding-right to specify the spacing around a paragraph. See §7.29.15 and §7.7.35- §7.7.38 of [XSL] for details.

For some application, the value of these properties can be a non-zero value only if there is a border at the same side and the border is specified within the same properties element. If a properties element contains a padding specification for one but not all four sides, some applications may also assign a zero or a default padding to these sides depending on whether or not there is a border at that side. There might be also other restriction regarding the combination of borders and paddings.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-padding-attlist"/>
</define>

<define name="common-padding-attlist">
    <optional>
        <attribute name="fo:padding">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:padding-top">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:padding-bottom">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:padding-left">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:padding-right">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.5.28 Shadow

Use the `style:shadow` property to specify a shadow effect for the paragraph.

The valid values for this attribute are the same as the values for the `fo:text-shadow` property. See section 15.4.27 for information.

Some applications may only supports a limited number of shadow effects, for instance only one effect where the the horizontal and vertical positions have the same value.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>

<define name="common-shadow-attlist">
    <optional>
        <attribute name="style:shadow">
            <ref name="shadowType"/>
        </attribute>
    </optional>
</define>
```

15.5.29 Keep with Next

Use the `fo:keep-with-next` property to specify whether or not to keep the current paragraph and the next paragraph together on a page or in a column after a break is inserted. See §7.9.14 of [XSL] for details. The only supported values are `auto` and `always`.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-keep-with-next-attlist"/>
</define>

<define name="common-keep-with-next-attlist">
    <optional>
        <attribute name="fo:keep-with-next">
            <choice>
                <value>auto</value>
                <value>always</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.30 Line Numbering

The `text:number-lines` attribute controls whether or not lines are numbered.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:number-lines" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.31 Line Number Start Value

The `text:line-number` property specifies a new start value for line numbering. The attribute is only recognized if there is also a `text:number-lines` attribute with a value of `true` in the same properties element.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:line-number">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.5.32 Text Autospace

Use the `style:text-autospace` property to specify whether to add space between Asian, western, and complex text.

The possible values are `none` and `ideograph-alpha`.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-autospace">
            <choice>
                <value>none</value>
                <value>ideograph-alpha</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.33 Punctuation Wrap

Use the `style:punctuation-wrap` property to determine whether or not a punctuation mark, if one is present, can be hanging, that is, whether it can placed in the margin area at the end of a full line of text. This is a common setting in East Asian typography.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:punctuation-wrap">
            <choice>
                <value>simple</value>
                <value>hanging</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.34 Line Break

Use the `style:line-break` property to select the set of line breaking rules to use for text. If the value is `strict`, line breaks are forbidden between certain user and application configurable characters. If the value is `normal`, line breaks may occur between arbitrary characters.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:line-break">
            <choice>
                <value>normal</value>
                <value>strict</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.35 Vertical Alignment

The `style:vertical-align` property specifies the vertical position of a character. By default characters are aligned according to their baseline, which is the default for most European languages. This is also the alignment used in this specification. Alternatively, characters may be vertically aligned as follows:

- `bottom` — To the bottom of the line.
- `top` —To the top of the line.
- `middle` —To the center of the line.
- `auto` — Automatically, which sets the vertical alignment to suit the text rotation. Text that is rotated 0 or 90 degrees is aligned to the baseline, while text that is rotated 270 degrees is aligned to the center of the line.

The following graphic illustrates the effect of the vertical alignment property when it is set to baseline, top, bottom, and middle respectively.

mgk mgk
mgk mgk
mgk mgk
mgk mgl

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:vertical-align" a:defaultValue="auto">
            <choice>
                <value>top</value>
                <value>middle</value>
                <value>bottom</value>
                <value>auto</value>
                <value>baseline</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.36 Writing Mode

The `style:writing mode` attribute specifies the writing mode of a paragraph. The attribute is similar to the `writing-mode` attribute specified in §7.27.7 of [XSL], except hat it has the additional value `page`. This value specifies that the writing mode is inherited from the page that contains the paragraph.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-writing-mode-attlist"/>
</define>

<define name="common-writing-mode-attlist">
    <optional>
        <attribute name="style:writing-mode">
            <choice>
                <value>lr-tb</value>
                <value>rl-tb</value>
                <value>tb-rl</value>
                <value>tb-lr</value>
                <value>lr</value>
                <value>rl</value>
                <value>tb</value>
                <value>page</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.37 Automatic Writing Mode

If the `style:writing-mode-automatic` attribute is given for a paragraph and if its value is `true`, then an application is allowed to recalculate the writing mode of the paragraph based on its content whenever the content changes. The actual value for the writing-mode should be contained in `style:writing-mode` attribute, so that applications that do not support an automatic writing mode calculation or use a different algorithm always know the actual value.

By specifying a `fo:text-align='start'` attribute additionally, the text alignment can be adapted to the writing mode simultaneously.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="style:writing-mode-automatic">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.38 Snap To Layout

The `style:snap-to layout-grid` attribute specifies whether the paragraph should consider the layout grid settings of the page. See section 15.2.21.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:snap-to-layout-grid">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.5.39 Page Number

If a paragraph style specifies a master page that should be applied beginning from the start of the paragraph, the `style:page-number` attribute specifies the page number that should be used for new page.

The attribute value can be an integer value or the value `auto`. An integer value specifies the page number of the new page directly. The value `auto` specifies that the page gets the page number of the previous page, incremented by one.

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <ref name="common-page-number-attlist"/>
</define>

<define name="common-page-number-attlist">
    <optional>
        <attribute name="style:page-number">
            <choice>
                <ref name="positiveInteger"/>
                <value>auto</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.5.40 Background Transparency

```
<define name="style-paragraph-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:background-transparency">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.6 Ruby Text Formatting Properties

The properties described in this section can be used within ruby styles (see section 14.8.4 for details). They are contained in a `<style:ruby-properties>` element.

```
<define name="style-ruby-properties">
    <element name="style:ruby-properties">
        <ref name="style-ruby-properties-content"/>
    </element>
</define>

<define name="style-ruby-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-ruby-properties-content-strict">
    <ref name="style-ruby-properties-attlist"/>
    <ref name="style-ruby-properties-elements"/>
</define>

<define name="style-ruby-properties-elements">
    <empty/>
</define>
```

15.6.1 Ruby Position

This property specifies the position of the ruby text relative to the ruby base.

```
<define name="style-ruby-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:ruby-position">
            <choice>
                <value>above</value>
                <value>below</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.6.2 Ruby Alignment

This property specifies the alignment of the ruby text relative to the ruby base.

```
<define name="style-ruby-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:ruby-align">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>distribute-letter</value>
                <value>distribute-space</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.7 Section Formatting Properties

The properties described in this section can be used within section styles (see section 14.8.3 for details). They are contained in a `<style:section-properties>` element.

```
<define name="style-section-properties">
    <element name="style:section-properties">
        <ref name="style-section-properties-content"/>
    </element>
</define>

<define name="style-section-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-section-properties-content-strict">
    <ref name="style-section-properties-attlist"/>
    <ref name="style-section-properties-elements"/>
</define>
```

15.7.1 Section Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the section. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-section-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-section-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.7.2 Margins

The margins attributes `fo:margin-left` and `fo:margin-right` specify the size of the section margins. See sections 15.5.17 for detailed information on these attributes. Percentage values are not supported.

```
<define name="style-section-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
</define>
```

15.7.3 Columns

The `<style:columns>` element contains `<style:column>` elements that specify each column individually (see section 15.7.4). 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 15.7.5 for information on this element.

```
<define name="style-section-properties-elements" combine="interleave">
    <ref name="style-columns"/>
</define>

<define name="style-columns">
    <optional>
        <element name="style:columns">
            <ref name="style-columns-attlist"/>
            <optional>
                <ref name="style-column-sep"/>
            </optional>
            <zeroOrMore>
                <ref name="style-column"/>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

The attributes that may be associated 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.

```
<define name="style-columns-attlist" combine="interleave">
    <attribute name="fo:column-count">
        <ref name="positiveInteger"/>
    </attribute>
</define>
```

> **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, then the gap between columns may be specified by the `fo:column-gap` attribute. If there are individual column elements, this attribute is ignored.

```
<define name="style-columns-attlist" combine="interleave">
    <optional>
        <attribute name="fo:column-gap">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

> **Note:** This attribute has the same name as an [XSL] property but it is attached to a different element.

15.7.4 Column Specification

The `<style:column>` element can be contained in a `<style:columns>` element, to specify details of an individual column. 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.

```
<define name="style-column">
    <element name="style:column">
        <ref name="style-column-attlist"/>
    </element>
</define>
```

> **Note:** In [XSL], it is not possible to specify columns individually.

The attributes that may be associated 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. The column widths are specified as number values 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.

The column width is not specified in a percentage length, but rather in terms of relative weights, that is, a number followed by a '*' character. The total space available for the entire table is distributed among its columns according to its relative widths. For example, if three columns are assigned the relative widths 1, 2 and 3, then the first column will take up 1/6 of the available width, the second will take up 1/3, and the last column will take up 1/2 of the available space. To achieve these figures, all given relative widths must be summed up (six in the example), and then each column will get as much space as the proportion of its own relative width to the sum of all relative widths indicates (3/6 = 1/2 for the last column in the example).

```
<define name="style-column-attlist" combine="interleave">
    <attribute name="style:rel-width">
        <ref name="relativeLength"/>
    </attribute>
</define>
```

Column Left, Right, Upper, and Lower Space

For each column, its left, right, upper, and lower space may be specified. 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.

```
<define name="style-column-attlist" combine="interleave">
    <optional>
        <attribute name="fo:start-indent" a:defaultValue="0cm">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
<define name="style-column-attlist" combine="interleave">
    <optional>
        <attribute name="fo:end-indent" a:defaultValue="0cm">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
<define name="style-column-attlist" combine="interleave">
    <optional>
        <attribute name="fo:space-before" a:defaultValue="0cm">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
<define name="style-column-attlist" combine="interleave">
    <optional>
        <attribute name="fo:space-after" a:defaultValue="0cm">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.7.5 Column Separator

The `<style:column-sep>` element specifies the separator line to use between columns. This element can be contained in a `<style:columns>` element to specify the type of separator line to use between columns.

```
<define name="style-column-sep">
    <element name="style:column-sep">
        <ref name="style-column-sep-attlist"/>
    </element>
</define>
```

Note: [XSL] does not support column separators.

The attributes that may be associated 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.

```
<define name="style-column-sep-attlist" combine="interleave">
    <optional>
        <attribute name="style:style" a:defaultValue="solid">
            <choice>
                <value>none</value>
                <value>solid</value>
                <value>dotted</value>
                <value>dashed</value>
                <value>dot-dashed</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Line Width

Use the `style:width` attribute to specify the width of the column separator line.

```
<define name="style-column-sep-attlist" combine="interleave">
    <attribute name="style:width">
        <ref name="length"/>
    </attribute>
</define>
```

Line Height

Use the `style:height` to specify the height of the column separator line. The value of this attribute is a percentage that relates to the height of the columned area.

```
<define name="style-column-sep-attlist" combine="interleave">
```

```
    <optional>
        <attribute name="style:height" a:defaultValue="100%">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

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. The value of this attribute can be either `top`, `middle`, or `bottom`.

```
<define name="style-column-sep-attlist" combine="interleave">
    <optional>
        <attribute name="style:vertical-align" a:defaultValue="top">
            <choice>
                <value>top</value>
                <value>middle</value>
                <value>bottom</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Line Color

Use the `style:color` attribute to specify the color of the column separator line.

```
<define name="style-column-sep-attlist" combine="interleave">
    <optional>
        <attribute name="style:color" a:defaultValue="#000000">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.7.6 Protect

Sections marked with the `style:protect` attribute should not be changed. The user interface should prevent the user from manually making any changes. The `style:protect` attribute should be 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.

```
<define name="style-section-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:protect" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.7.7 Don't Balance Text Columns

The text:dont-balance-text-columns attribute specifies whether the text column content should be evenly distributed over all text columns or not.

```
<define name="style-section-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="text:dont-balance-text-columns">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.7.8 Writing Mode

The `style:writing-mode` attribute specifies the writing mode that should be used for the section. See section 15.5.36 for details.

```
<define name="style-section-properties-attlist" combine="interleave">
    <ref name="common-writing-mode-attlist"/>
</define>
```

15.7.9 Notes Configuration

A section style may contain have its own notes configurations (see section 14.9.2). If this is the case, notes of the corresponding notes type are displayed at the end of the columns of the section or the section itself instead of the end of the page's columns or the end of the document.

```
<define name="style-section-properties-elements" combine="interleave">
    <zeroOrMore>
        <ref name="text-notes-configuration"/>
    </zeroOrMore>
</define>
```

15.8 Table Formatting Properties

The properties described in this section can be contained within table styles (see section 14.12.1) They are contained in a `<style:table-properties>` element.

```
<define name="style-table-properties">
    <element name="style:table-properties">
        <ref name="style-table-properties-content"/>
    </element>
</define>

<define name="style-table-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-table-properties-content-strict">
    <ref name="style-table-properties-attlist"/>
    <ref name="style-table-properties-elements"/>
</define>
```

15.8.1 Table Width

Every table must have a fixed width. This width is specified by the `style:width` attribute.

The width of a table may be also specified relative to the width of the area that the table is in. In this case, the width is specified as a percentage using the `style:rel-width` attribute. User agents that support specifying the relative width of a table can specify widths in this way, but it is not essential.

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.

```
<define name="style-table-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:width">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:rel-width">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.8.2 Table Alignment

A table alignment property `table:align` specifies the horizontal alignment of a table.

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.

User agents that do not support the `margins` value, may treat this value as `left`.

```
<define name="style-table-properties-attlist" combine="interleave">
    <optional>
        <attribute name="table:align">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>margins</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.8.3 Table Left and Right Margin

The `fo:margin-left` and `fo:margin-right` properties specify the distance of the table from the left and right margins. See section 15.5.17 for a full explanation of left and right margin properties. 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.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
</define>
```

15.8.4 Table Top and Bottom Margin

The `fo:margin-top` and `fo:margin-bottom` properties specify the distance of the table from the top and bottom. See section 15.5.20 for a full explanation of top and bottom margin properties.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-vertical-margin-attlist"/>
</define>
```

15.8.5 Table Margins

The `fo:margin` property specifies the distance of the table from the left, right, top and bottom. See section 15.5.21 for a full explanation of this property.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-margin-attlist"/>
</define>
```

15.8.6 Page Number

If the table style specifies a master page that should be applied beginning from the start of the table, the `style:page-number` attribute specifies the page number that should be used for the first page of the table. See also section 15.5.39.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-page-number-attlist"/>
</define>
```

15.8.7 Break Before and Break After

The `fo:break-before` and `fo:break-after` properties insert a page or column break before or after a table. See section 15.5.22 for a full explanation of these properties.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-break-attlist"/>
</define>
```

15.8.8 Table Background and Background Image

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the table. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-table-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.8.9 Table Shadow

The `style:shadow` property specifies that a shadow visual effect appears on a table. See section 15.5.28 for a full explanation of this property.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>
```

15.8.10 Keep with Next

The `fo:keep-with-next` property specifies that a table stays with the paragraph that follows it. See section 15.5.29 for a full explanation of this property.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-keep-with-next-attlist"/>
</define>
```

15.8.11 May Break Between Rows

The `style:may-break-between-rows` property specifies that a page break may occur inside a table.

```
<define name="style-table-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:may-break-between-rows">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.8.12 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 OpenDocument, 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.

```
<define name="style-table-properties-attlist" combine="interleave">
    <optional>
        <attribute name="table:border-model">
            <choice>
                <value>collapsing</value>
                <value>separating</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.8.13 Writing Mode

The `style:writing-mode` attribute specifies the writing mode that should is used for the table. See section 15.5.36 for details.

```
<define name="style-table-properties-attlist" combine="interleave">
    <ref name="common-writing-mode-attlist"/>
</define>
```

15.8.14 Display

The `table:display` attribute specifies whether or not a table is displayed.

```
<define name="style-table-properties-attlist" combine="interleave">
    <optional>
        <attribute name="table:display">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.9 Column Formatting Properties

The properties described in this section can be contained within table column styles (see section 14.12.2) They are contained in a `<style:table-column-properties>` element.

```
<define name="style-table-column-properties">
    <element name="style:table-column-properties">
        <ref name="style-table-column-properties-content"/>
    </element>
</define>

<define name="style-table-column-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-table-column-properties-content-strict">
    <ref name="style-table-column-properties-attlist"/>
    <ref name="style-table-column-properties-elements"/>
</define>

<define name="style-table-column-properties-elements">
    <empty/>
</define>
```

15.9.1 Column Width

Every table column must have a fixed width. This width is specified by the `style:column-width` attribute.

The width of a column may be also specified relative to the other column widths. Applications that support specifying the relative width of a column may specify widths in this way, but it is not essential.

A relative width is specified by the `style:rel-column-width` property that takes a number value, followed by a '*' character. If r_c is the relative with of the column, r_s the sum of all relative columns widths, and w_s the absolute width that is available for these columns, then the absolute with w_c of the column is $w_c=r_cw_s/r_s$.

```
<define name="style-table-column-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:column-width">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:rel-column-width">
            <ref name="relativeLength"/>
        </attribute>
    </optional>
</define>
```

15.9.2 Optimal Table Column Width

The `style:use-optimal-column-width` attribute specifies that the column width should be recalculated automatically if some content in the column changes.

```
<define name="style-table-column-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:use-optimal-column-width">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.9.3 Break Before and Break After

The `fo:break-before` and `fo:break-after` properties insert a page or column break before or after a table column. See section 15.5.22 for a full explanation of these properties.

```
<define name="style-table-column-properties-attlist" combine="interleave">
    <ref name="common-break-attlist"/>
</define>
```

15.10 Table Row Formatting Properties

The properties described in this section can be contained within table row styles (see section 14.12.3) They are contained in a `<style:table-row-properties>` element.

```
<define name="style-table-row-properties">
    <element name="style:table-row-properties">
        <ref name="style-table-row-properties-content"/>
    </element>
</define>

<define name="style-table-row-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-table-row-properties-content-strict">
    <ref name="style-table-row-properties-attlist"/>
    <ref name="style-table-row-properties-elements"/>
</define>
```

15.10.1 Row Height

The `style:row-height` and `style:min-row-height` properties specifies the height of a table row. By default, the row height is the height of the tallest item in the row.

The `style:row-height` property specifies a fixed row height, while the `style:min-row-height` property specifies a fixed height.

```
<define name="style-table-row-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:row-height">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:min-row-height">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.10.2 Optimal Table Row Height

The `style:use-optimal-row-height` attribute specifies that the row height should be recalculated automatically if some content in the row changes.

```
<define name="style-table-row-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:use-optimal-row-height">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.10.3 Row Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the table. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-table-row-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-table-row-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.10.4 Break Before and Break After

The `fo:break-before` and `fo:break-after` properties insert a page or row break before or after a table row. See section 15.5.22 for a full explanation of these properties.

```
<define name="style-table-row-properties-attlist" combine="interleave">
    <ref name="common-break-attlist"/>
</define>
```

15.10.5 Keep Together

Use the `fo:keep-together` property to control whether the contents of a table cell should be kept together on the same page or column (if the value is `always`), or whether breaks are allowed within the cell (if the value is `auto`). See §7.19.3 of [XSL] for details.

```
<define name="style-table-row-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="fo:keep-together">
            <choice>
                <value>auto</value>
                <value>always</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11 Table Cell Formatting Properties

The properties described in this section can be contained within table cell styles (see section 14.12.4) They are contained in a `<style:table-column-properties>` element.

```
<define name="style-table-cell-properties">
    <element name="style:table-cell-properties">
        <ref name="style-table-cell-properties-content"/>
    </element>
</define>

<define name="style-table-cell-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-table-cell-properties-content-strict">
    <ref name="style-table-cell-properties-attlist"/>
    <ref name="style-table-cell-properties-elements"/>
</define>
```

15.11.1 Vertical Alignment

The vertical alignment property `style:vertical-align` is used to specify the vertical alignment of text in a table cell.

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.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:vertical-align">
            <choice>
                <value>top</value>
                <value>middle</value>
                <value>bottom</value>
                <value>automatic</value>
            </choice>
            </attribute>
    </optional>
</define>
```

15.11.2 Text Align Source

The `style:text-align-source` property specifies the source of the text-align property. If the value of this attribute is `fix`, the value of the `fo:text-align` property is used. If the value is `value-type`, the text alignment depends on the value-type of the cell.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:text-align-source">
            <choice>
                <value>fix</value>
                <value>value-type</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.3 Direction

The `style:direction` 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.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-style-direction-attlist"/>
</define>

<define name="common-style-direction-attlist">
    <optional>
        <attribute name="style:direction">
            <choice>
                <value>ltr</value>
                <value>ttb</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.4 Vertical Glyph Orientation

The `style:glyph-orientation-vertical` property specifies the vertical glyph orientation. The property specifies an angle or automatic mode. The only possible angle is 0, which disables this feature.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:glyph-orientation-vertical">
            <choice>
                <value>auto</value>
                <value>0</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.5 Cell Shadow

The `style:shadow` property specifies that a shadow visual effect appears on a table cell. See section 15.5.28 for a full explanation of this property.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>
```

15.11.6 Cell Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the table cell. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-table-cell-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.11.7 Cell Border

The border attributes `fo:border`, `fo:border-top`, `fo:border-bottom`, `fo:border-left` and `fo:border-right` specify the border properties of the table cell. See section 15.5.25 for detailed information on these attributes.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-border-attlist"/>
</define>
```

15.11.8 Diagonal Lines

Spreadsheet cells can also have diagonal lines, which follow the same specification as borders.

`style:diagonal-tl-br` defines the style of "border" to use for the topleft-bottomright diagonal (see section 15.5.25 for detailed information). In case of a double line, `style:diagonal-bl-tr-widths` allows to specify the width of the inner and outer lines and the distance between them (see section 15.5.26 for detailed information).

`style:diagonal-bl-tr` and `style:diagonal-tl-br-widths` define the same properties for the bottomleft-topright diagonal.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:diagonal-tl-br">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:diagonal-tl-br-widths">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:diagonal-bl-tr">
            <ref name="string"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="style:diagonal-bl-tr-widths">
            <ref name="borderWidths"/>
        </attribute>
    </optional>
</define>
```

15.11.9 Border Line Width

The border line width attributes `style:border-line-width`, `style:border-line-width-top`, `style:border-line-width-bottom`, `style:border-line-width-left` and `style:border-line-width-right` specify the properties of the border lines of the page. See section 15.5.26 for detailed information on these attributes.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-border-line-width-attlist"/>
</define>
```

15.11.10 Padding

The padding attributes `fo:padding`, `fo:padding-top`, `fo:padding-bottom`, `fo:padding-left` and `fo:padding-right` specify the padding properties of the table cell. See section 15.5.27 for detailed information on these attributes.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-padding-attlist"/>
</define>
```

15.11.11 Wrap Option

The `fo:wrap-option` property specifies whether text wraps within a table cell. See §7.5.13 of [XSL] for details. If wrapping is disabled, the application determines whether the clipped text is visible or hidden. If the text is hidden applications may support a scrolling mechanism to access the text. This is similar to setting a `fo:overflow` property to a value of `auto`. See also §7.20.2 of [XSL].

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:wrap-option">
            <choice>
                <value>no-wrap</value>
                <value>wrap</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.12 Rotation Angle

The `style:rotation-angle` property specifies the rotation angle of the cell content in degrees.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <ref name="common-rotation-angle-attlist"/>
</define>

<define name="common-rotation-angle-attlist">
    <optional>
        <attribute name="style:rotation-angle">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.11.13 Rotation Align

The `style:rotation-align` property specifies how the edge of the text in a cell is aligned after a rotation. There are four alignment options: "`none`", "`bottom`", "`top`", or "`center`".

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.		

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:rotation-align">
            <choice>
                <value>none</value>
                <value>bottom</value>
                <value>top</value>
                <value>center</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.14 Cell Protect

The `style:cell-protect` property specifies how a cell is protected.

This attribute is only evaluated if the current table is protected (see section 8.1.1). The value of the attribute can be "`none`", "`hidden-and-protected`", or a space-separated list containing the values "`protected`" or "`formula-hidden`".

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:cell-protect">
            <choice>
                <value>none</value>
                <value>hidden-and-protected</value>
                <list>
                    <oneOrMore>
                        <choice>
                            <value>protected</value>
                            <value>formula-hidden</value>
                        </choice>
                    </oneOrMore>
                </list>
            </choice>
        </attribute>
    </optional>
</define>
```

15.11.15 Print Content

The `style:print-content` property specifies whether or not the cell content is printed.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:print-content">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.11.16 Decimal places

The `style:decimal-places` attribute specifies the maximum number of decimal places that are displayed if numbers are formatted by a data style that has no setting for number of decimal places itself. See also section 14.7.9.

This property is usually only evaluated if it is contained in a default style (see section 14.2).

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:decimal-places">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.11.17 Repeat Content

The `style:repeat-content` property specifies whether the content of a cell is displayed as many times as there is space left in the cell's writing direction. Only full instances of the text are displayed. The property has no effect for cell content that contains a line break. This property is for instance used to "fill" a table cell with "-" or "x" characters so that no other data can be entered.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:repeat-content">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.11.18 Shrink To Fit

The `style:shrink-to-fit` property specifies whether the content of a cell, if necessary, gets shrunk to fit into the cell. Shrinking does mean that the cell's font size is decreased, so that the complete text fits into the cell. The property has no effect on cells where the cell content fits already into the cell.

```
<define name="style-table-cell-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:shrink-to-fit">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.12 List-Level Style Properties

The properties described in this section can be contained within the various list style level elements (see section 14.10). They are contained in a `<style:list-level-properties>` element.

```
<define name="style-list-level-properties">
    <element name="style:list-level-properties">
        <ref name="style-list-level-properties-content"/>
    </element>
</define>

<define name="style-list-level-properties-content">
    <ref name="style-properties-content"/>
</define>

<define name="style-list-level-properties-content-strict">
    <ref name="style-list-level-properties-attlist"/>
    <ref name="style-list-level-properties-elements"/>
</define>

<define name="style-list-level-properties-elements">
    <empty/>
</define>
```

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. See also section 15.5.5,

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <ref name="common-text-align"/>
</define>
```

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.

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.

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:space-before">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

Minimum Label Width

The `text:min-label-width` attribute specifies the minimum width of a number.

This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.

The label can be aligned horizontally with the width using an `fo:text-align` property. See the Label Alignment attribute below for more information.

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:min-label-width">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

Minimum Label Distance

The `text:min-label-distance` attribute specifies the minimum distance between the number and the text of the list item.

This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:min-label-distance">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

Font Name

The `style:font-name` attribute species the name of a font that is used to display a bullet character. See also section 15.4.13.

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:font-name">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

Image Size

The size of the image is specified by the following attributes:

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:width">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:height">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
</define>
```

Vertical Alignment

The vertical alignment of the image is specified by the `style:vertical-pos` and style:vertical-rel properties. See sections 15.27.11 and 15.27.12 for details.

```
<define name="style-list-level-properties-attlist" combine="interleave">
    <ref name="common-vertical-rel-attlist"/>
    <ref name="common-vertical-pos-attlist"/>
</define>
```

15.13 Stroke Properties

The following **stroke properties** are used to define drawing object line characteristics. They are available for drawing objects contained in all kinds of applications.

- Style
- Dash
- Width
- Color
- Start marker
- End marker
- Start marker width
- End marker width
- Start marker center
- End marker center
- Opacity
- Joint

The properties described in this section can be contained within style elements `<style:style>` whose family is either `graphic` or `presentation`. They are contained in a `<style:graphic-properties>` element.

15.13.1 Stroke Style

The attribute `draw:stroke` specifies the style of the stroke on the current object. The value `none` means that no stroke is drawn, and the value `solid` means that a solid stroke is drawn. If the value is `dash`, the stroke referenced by the `draw:stroke-dash` property is drawn.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:stroke">
            <choice>
                <value>none</value>
                <value>dash</value>
                <value>solid</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.13.2 Dash

The attribute `draw:stroke-dash` specifies the dash style that is used for the stroke. See section 14.14.7 for dash styles.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:stroke-dash">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.13.3 Multiple Dashes

The attribute `draw:stroke-dash-names` specifies a list of dash styles that are used for the stroke in addition to the dash specified by the `draw:stroke-dash` attribute. See section 15.13.2 for the `draw:stroke-dash` attribute and section 14.14.7 for dash styles.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:stroke-dash-names">
            <ref name="styleNameRefs"/>
        </attribute>
    </optional>
</define>
```

15.13.4 Width

The attribute `svg:stroke-width` specifies the width of the stroke on the current object.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="svg:stroke-width">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.13.5 Color

The attribute `svg:stroke-color` specifies the color of the stroke on the current object.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="svg:stroke-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.13.6 Start Marker

The attribute `draw:marker-start` specifies a line start marker, which is a path that can be connected to the start of a stroke. See section 14.14.6 for markers.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-start">
```

```
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.13.7 End Marker

The attribute `draw:marker-end` specifies a stroke end marker, which is a path that can be connected to the end of a stroke. See section 14.14.6 for markers.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-end">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.13.8 Start Marker Width

The attribute `draw:marker-start-width` specifies the width of the marker at the start of the stroke.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-start-width">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.13.9 End Marker Width

The attribute `draw:marker-end-width` specifies the width of the marker at the end of the stroke.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-end-width">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.13.10 Start Marker Center

The attribute `draw:marker-start-center` specifies whether or not a start marker is centered at the start of a stroke.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-start-center">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.13.11 End Marker Center

The attribute `draw:marker-end-center` specifies whether or not an end marker is centered at the end of a stroke.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:marker-end-center">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.13.12 Opacity

The attribute `svg:stroke-opacity` specifies the opacity of a stroke. The value of this attribute can be a number between 0 (fully transparent) and 1 (fully opaque) or a percentage.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="svg:stroke-opacity">
            <choice>
                <data type="double">
                    <param name="minInclusive">0</param>
                    <param name="maxInclusive">1</param>
                </data>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.13.13 Line Join

The attribute `draw:stroke-linejoin` specifies the shape at the corners of paths or other vector shapes, when they are stroked. The values are the same as for [SVG]'s `stroke-linejoin` attribute, except that the attribute in addition to the values supported by SVG may have the value `middle`, which means that the mean value between the joints is used.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:stroke-linejoin">
            <choice>
                <value>miter</value>
                <value>round</value>
                <value>bevel</value>
                <value>middle</value>
                <value>none</value>
                <value>inherit</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14 Fill Properties

The following **fill properties** are used to define drawing object fill characteristics. They are available for drawing objects contained in all kinds of applications.

- Style
- Color
- Gradient
- Gradient step count
- Hatch
- Solid hatch
- Bitmap
- Opacity
- Fill rule

15.14.1 Fill Style

The attribute `draw:fill` specifies the fill style for a graphic object. Graphic objects that are not closed, such as a path without a `closepath` at the end, will not be filled. The fill operation does not automatically close all open subpaths by connecting the last point of the subpath with the first point of the subpath before painting the fill. The attribute has the following values:

- `none`: the drawing object is not filled.
- `solid`: the drawing object is filled with color specified by the `draw:fill-color` attribute.
- `bitmap`: the drawing object is filled with the bitmap specified by the `draw:fill-image-name` attribute.
- `gradient`: the drawing object is filled with the gradient specified by the `draw:fill-gradient-name` attribute.
- `hatch`: the drawing object is filled with the hatch specified by the `draw:fill-hatch-name` attribute.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill">
            <choice>
                <value>none</value>
                <value>solid</value>
                <value>bitmap</value>
                <value>gradient</value>
                <value>hatch</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14.2 Color

The attribute `draw:fill-color` specifies the color of the fill for a graphic object. It is used only if the `draw:fill` attribute has the value `solid`.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.14.3 Secondary Fill Color

The `draw:secondary-fill-color` attribute specifies the secondary fill color. It may be used as fill color for the extrusion.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:secondary-fill-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.14.4 Gradient

The attribute `draw:fill-gradient-name` specifies a gradient style that is used for filling graphic objects. It is used only if the `draw:fill` attribute has the value `gradient`. See section 14.14.1 and 14.14.2 for gradients.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-gradient-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.14.5 Gradient Step Count

If a gradient is used for filling, the attribute `draw:gradient-step-count` can be used to set the gradient step count of the color interpolation to be a fixed value. By default, the step count is automatically calculated based on the size and resolution of the filled area.

A step count less than 3 is not valid as there would be no interpolation possible. Values above 256 may not be supported or may result in performance issues.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:gradient-step-count">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.14.6 Hatch

The attribute `draw:fill-hatch-name` specifies a hatch style that is used for filling. It is used only if the `draw:fill` attribute has the value `hatch`. See section 14.14.3 for hatches.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-hatch-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.14.7 Solid Hatch

The attribute `draw:fill-hatch-solid` specifies whether the background of a hatch filling is solid or transparent.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-hatch-solid">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.14.8 Fill Image

The attribute `draw:fill-image-name` specifies a fill image that is used for filling. It is used only if the `draw:fill` attribute has the value `bitmap`. See section 14.14.4 for fill images.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-image-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.14.9 Fill Image Rendering Style

If an image is used for filling, the bitmap image can either be rendered in the given size, stretched to the filled area, or tiled over the area. The attribute `style:repeat` specifies how the bitmap image should be treated.

The value of the attribute can be `no-repeat`, `repeat`, or `stretch`.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:repeat">
            <choice>
                <value>no-repeat</value>
                <value>repeat</value>
                <value>stretch</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14.10 Fill Image Size

If an image is used for filling, the optional attributes `draw:fill-image-width` and `draw:fill-image-height` can be used to override the logical size of the source image data. If the value of the `style:repeat` attribute is `stretch`, these attributes are ignored.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-image-width">
```

```
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:fill-image-height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14.11 Fill Image Tile Reference Point

If an image is used for filling, the attributes `draw:fill-image-ref-point`, `draw:fill-image-ref-point-x` and `draw:fill-image-ref-point-y` specify the reference position of the image. The `draw:fill-image-ref-point` attribute specifies the position as an alignment of the image within the filling area, while the draw:`fill-image-ref-point-x` and `draw:fill-image-ref-point-y` attributes specify an horizontal and vertical movement as percentage values, where the percentage value relates to the image width and height. If an alignment and a movement is specified at the same time, the image first is aligned and afterwards moved.

These attributes are only interpreted if the value of the current `style:repeat` attribute is `repeat`.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fill-image-ref-point-x">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:fill-image-ref-point-y">
            <ref name="percent"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:fill-image-ref-point">
            <choice>
                <value>top-left</value>
                <value>top</value>
                <value>top-right</value>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>bottom-left</value>
                <value>bottom</value>
                <value>bottom-right</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14.12 Fill Image Tile Translation

If an image is used for filling, the attribute `draw:tile-repeat-offset` defines the translation of each tile in relation to the previous tile. 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`.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:tile-repeat-offset"/>
    </optional>
</define>
```

Example: Tile translation

```
<style:graphic-properties draw:tile-repeat-offset="50% horizontal"/>
```

15.14.13 None and Linear Opacity

The fill area of a graphic object can either have a full, a linear, or gradient opacity. Full and linear opacity is selected using the `draw:opacity` attribute, while gradient opacity is selected using the `draw:opacity-name` attribute.

The `draw:opacity` attribute disables any transparency effect or sets a linear opacity for the fill area of a graphic object.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:opacity">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.14.14 Gradient Opacity

The `draw:opacity-name` attribute specifies an opacity gradient that defines the opacity for the fill area of a graphic object. When applying an opacity gradient, the opacity is interpolated as defined in the referenced opacity gradient style. This fill style is rendered independently from other fill styles like gradient, image, and hatch. See section 14.14.5 for opacity gradients.

The value of this attribute overrides the `draw:opacity` attribute.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:opacity-name">
            <ref name="styleNameRef"/>
        </attribute>
    </optional>
</define>
```

15.14.15 Fill Rule

The `svg:fill-rule` specifies the algorithm which is to be used to determine what parts of the canvas are included inside the shape. See §11.3 of [SVG] for more details.

```
<define name="style-graphic-fill-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="svg:fill-rule">
            <choice>
                <value>nonzero</value>
                <value>evenodd</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.14.16 Symbol color

The `draw:symbol-color` attribute defines the color to be used to draw symbols contained on the drawing object. This could be for instance arrows displayed within a control.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:symbol-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.15 Text Animation Properties

Drawing objects that contain text and text boxes can have optional text animation properties. These properties always animate the complete text of a drawing object or text frame. The following attributes define the text animation:

- Animation
- Animation direction
- Animation start inside
- Animation stop inside
- Animation repeat
- Animation delay
- Animation steps

These properties are available for drawing objects contained in all kinds of applications.

15.15.1 Animation

The attribute `text:animation` specifies the type of animation that is used for the text.

The value of this attribute can be one of the following:

- `none`, disables the text animation.
- `scroll`, scrolls the text from one side to another.
- `alternate`, scrolls the text from one side to another and back.
- `slide`, scrolls the text from one side to the original text position and stops there.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="text:animation">
            <choice>
                <value>none</value>
                <value>scroll</value>
                <value>alternate</value>
                <value>slide</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.15.2 Animation Direction

The attribute `text:animation-direction` specifies the scroll direction of animated text.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-direction">
            <choice>
                <value>left</value>
                <value>right</value>
                <value>up</value>
                <value>down</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.15.3 Animation Start Inside

If this attribute `text:animation-start-inside` is `true`, the text starts its animation inside the shape. If its `false`, the text starts its animation just outside the shapes bounding rectangle.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-start-inside">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.15.4 Animation Stop Inside

If this attribute `text:animation-stop-inside` is `true`, the text stops when it is inside the the shape. If its `false`, the text stops its animation just outside the shapes bounding rectangle.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-stop-inside">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.15.5 Animation Repeat

The attribute `text:animation-repeat` specifies the number of times the animation is repeated. If the value of the attribute is `0`, the animation is repeated indefinitely.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-repeat">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.15.6 Animation Delay

The attribute `text:animation-delay` specifies a delay before the animation is started. The value of this attribute must conform to the time period format described in §3.2.6 of [xmlschema-2].

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-delay">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

15.15.7 Animation Steps

The attribute `text:animation-steps` specifies the distance by which text is moved within each scrolling step.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="text:animation-steps">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.16 Text and Text Alignment Properties

Drawing objects that contain text and text boxes can have optional properties that specify how the text is aligned within the drawing object. These properties are available for drawing objects contained in all kinds of applications.

15.16.1 Auto Grow Width and Height

The attributes `draw:auto-grow-width` and `draw:auto-grow-height` specify whether or not to automatically increase the width and height of the drawing object if text is added to the drawing object. These attributes usually are evaluated only for text boxes.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:auto-grow-width">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:auto-grow-height">
            <ref name="boolean"/>
        </attribute>
    </optional>
```

```
</define>
```

15.16.2 Fit To Size

The attribute `draw:fit-to-size` specifies whether or not to stretch the text content of a drawing object to fill the entire object. If the value of the attribute is `true`, the text content is stretched.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fit-to-size">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.16.3 Fit To Contour

The attribute `draw:fit-to-contour` specifies whether or not to stretch the text content of a drawing object to fill the contour of the object. If the value of the attribute is `true`, the text content is stretched.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:fit-to-contour">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.16.4 Text Area Vertical Align

The attribute `draw:textarea-vertical-align` specifies the vertical alignment of the text area inside a shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:textarea-vertical-align">
            <choice>
                <value>top</value>
                <value>middle</value>
                <value>bottom</value>
                <value>justify</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.16.5 Text Area Horizontal Align

The attribute `draw:textarea-horizontal-align` specifies the horizontal alignment of the text area inside a shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:textarea-horizontal-align">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>justify</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.16.6 Word Wrap

The `fo:wrap-option` attribute specifies if text is word wrapped in a shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:wrap-option">
            <choice>
                <value>no-wrap</value>
                <value>wrap</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.16.7 List Styles

The `<text:list-style>` element as described in section 14.10 specifies a list style that is applied to the paragraphs contained in a text box. Although the list style has a name, it is not displayed in the user interface, even if the graphic style that contains it is a common style.

Including a list style element into a graphic style has the same semantics as adding a `style:list-style-name` attribute (see section 14.1) to the style that references a list style that is declared outside a graphic style. The inclusion of a list style element is required in cases where a common graphic style should be associated with an automatic list style.

List styles contained in a graphic style can be referenced by other graphic styles using the `style:list-style-name` attribute.

```
<define name="style-graphic-properties-elements" combine="interleave">
    <optional>
        <ref name="text-list-style"/>
    </optional>
</define>
```

15.17 Color Properties

Drawing objects that display a bitmap graphic can have optional properties that adjust the colors of the bitmap. These properties are available for drawing objects contained in all kinds of applications.

15.17.1 Color Mode

The attribute `draw:color-mode` affects the output of colors from a source bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:color-mode">
            <choice>
```

```
                <value>greyscale</value>
                <value>mono</value>
                <value>watermark</value>
                <value>standard</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.17.2 Color Inversion

The attribute `draw:color-inversion` specifies whether or not the colors in the graphic shape should be inverted.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:color-inversion">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.17.3 Adjust Luminance

The attribute `draw:luminance` specifies a signed percentage value that affects the output luminance of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:luminance">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.17.4 Adjust Contrast

The attribute `draw:contrast` specifies a signed percentage value that affects the output contrast of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:contrast">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.17.5 Adjust Gamma

The attribute `draw:gamma` specifies a value that affects the output gamma of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:gamma">
            <ref name="percent"/>
        </attribute>
    </optional>
```

```
</define>
```

15.17.6 Adjust Red

The attribute `draw:`red specifies a signed percentage value that affects the output of the red color space of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:red">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.17.7 Adjust Green

The attribute `draw:green` specifies a signed percentage value that affects the output of the green color space of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:green">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.17.8 Adjust Blue

The attribute `draw:blue` specifies a signed percentage value that affects the output of the blue color space of a bitmap or raster graphic.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:blue">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.17.9 Adjust Opacity

The attribute `draw:image-opacity` adjusts the opacity of an image. The value can be between 0% and 100%. See also section 15.14.13.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:image-opacity">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.18 Shadow Properties

Most drawing objects can have a shadow. The following attributes specify how the shadow is rendered. These properties are available for drawing objects contained in all kinds of applications.

15.18.1 Shadow

The attribute `draw:shadow` enables or disables the visibility of a shadow.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:shadow">
            <choice>
                <value>visible</value>
                <value>hidden</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.18.2 Offset

The attributes `draw:shadow-offset-x` and `draw:shadow-offset-y` are used to render a shadow. A copy of the shape is rendered in the single shadow color (specified by `draw: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

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:shadow-offset-x">
            <ref name="length"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:shadow-offset-y">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.18.3 Color

The attribute `draw:shadow-color` specifies the color in which the shadow is rendered.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:shadow-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.18.4 Opacity

The attribute `draw:shadow-opacity` specifies the opacity in which the shadow is rendered. The value of this attribute is a percentage value.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:shadow-opacity">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.19 Connector Properties

The properties described in this section are specific to connector drawing objects. These properties are available for connector drawing objects contained in all kinds of applications.

15.19.1 Start Line Spacing

For standard connectors, the attributes `draw:start-line-spacing-horizontal` and `draw:start-line-spacing-vertical` increment the length of the escape line from the start shape for standard connectors. For lines connectors, these attributes specify the absolute length of the escape line from the start shape. For other connector types, they are ignored.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-line-spacing-horizontal">
            <ref name="distance"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:start-line-spacing-vertical">
            <ref name="distance"/>
        </attribute>
    </optional>
</define>
```

15.19.2 End Line Spacing

For standard connectors, the attributes `draw:end-line-spacing-horizontal` and `draw:end-line-spacing-vertical` increment the length of the escape line from the end shape. For lines connectors, they specify the absolute length of the escape line from the end shape. For other connector types, they are ignored.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:end-line-spacing-horizontal">
            <ref name="distance"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:end-line-spacing-vertical">
            <ref name="distance"/>
        </attribute>
    </optional>
</define>
```

15.20 Measure Properties

The properties described in this section are specific to measure drawing objects. These properties are available for measure drawing objects contained in all kinds of applications.

15.20.1 Line Distance

The attribute `draw:line-distance` specifies the distance from the reference points to the measure line.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="draw:line-distance">
            <ref name="distance"/>
        </attribute>
    </optional>
</define>
```

15.20.2 Guide Overhang

The guides are the two lines from the reference points to the measure line. The attribute `draw:guide-overhang` specifies the length that the guides are drawn after they cross the measure line.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:guide-overhang">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.20.3 Guide Distance

The attribute `draw:guide-distance` specifies the distance between the reference points and the start point of the guide lines. This distance does not take the attributes `draw:start-guide` and `draw:end-guide` into account, that is, the distance specified in `draw:guide-distance` equals the distance that is actually drawn only if `draw:start-guide` and `draw:end-guide` both are 0.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:guide-distance">
            <ref name="distance"/>
        </attribute>
    </optional>
</define>
```

15.20.4 Start Guide

The `draw:start-guide` attribute specifies a length that is added to the length of the guide from the first reference point to the measure line. The guide is extended by this length at the end that points towards the reference points.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:start-guide">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.20.5 End Guide

The `draw:end-guide` attribute specifies a length that is added to the length of the guide from the second reference point to the measure line. The guide is extended by this length at the end that points towards the reference points.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="draw:end-guide">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.20.6 Placing

The attribute `draw:placing` specifies whether the measure line is rendered below or above the edge defined by the two reference points. The value of this attribute can be `below` or `above`.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:placing">
            <choice>
                <value>below</value>
                <value>above</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.20.7 Parallel

The `draw:parallel` attributes specifies whether the measure text is displayed parallel to the measure line or perpendicular.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:parallel">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.20.8 Text Alignment

The attributes `draw:measure-align` and `draw:measure-vertical-align` determine the horizontal and vertical alignment of the measure text relative to the measure line. If value of these attributes is `automatic`, the application chooses the best position.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:measure-align">
            <choice>
                <value>automatic</value>
                <value>left-outside</value>
                <value>inside</value>
                <value>right-outside</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:measure-vertical-align">
            <choice>
                <value>automatic</value>
                <value>above</value>
                <value>below</value>
                <value>center</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.20.9 Unit

The attribute `draw:unit` specifies the unit used in the textual presentation of a measure shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:unit">
            <choice>
                <value>automatic</value>
                <value>mm</value>
                <value>cm</value>
                <value>m</value>
                <value>km</value>
                <value>pt</value>
                <value>pc</value>
                <value>inch</value>
                <value>ft</value>
                <value>mi</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.20.10 Show Unit

The attribute `draw:show-unit` toggles the display of the unit in the textual presentation of a measure shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:show-unit">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.20.11 Decimal Places

The attribute `draw:decimal-places` specifies the number of decimal places that are used for the measure text.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:decimal-places">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.21 Caption Properties

The following attributes can be used in the styles for caption shapes. These properties are available for caption objects contained in all kinds of applications.

- Type
- Angle type
- Angle
- Gap
- Escape direction
- Escape
- Line length
- Fit line length

15.21.1 Type

The attribute `draw:caption-type` specifies the geometry of the line of a caption.

- `straight-line:` a straight perpendicular line is drawn to the caption point.
- `angled-line:` a straight line is drawn to the caption point.
- `angled-connector-line:` a straight perpendicular line, followed by a straight line is drawn to the caption point.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-type">
            <choice>
                <value>straight-line</value>
                <value>angled-line</value>
                <value>angled-connector-line</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.21.2 Angle Type

The attribute `draw:caption-angle-type` specifies if the escape angle of the line of a caption is fixed or free. If this is set to `free` the application can choose the best possible angle.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-angle-type">
            <choice>
                <value>fixed</value>
                <value>free</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.21.3 Angle

The attribute `draw:caption-angle` specifies the escape angle of the line of a caption. It is evaluated only if `draw:caption-angle-type` has the value `fixed`.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-angle">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.21.4 Gap

The attribute `draw:caption-gap` specifies the distance between the text area of the caption and the start of the line.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-gap">
            <ref name="distance"/>
        </attribute>
    </optional>
</define>
```

15.21.5 Escape Direction

The attribute `draw:caption-escape-direction` specifies the escape direction for the line of a caption. If this is set to `auto` the application can choose the best direction.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-escape-direction">
            <choice>
                <value>horizontal</value>
                <value>vertical</value>
                <value>auto</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.21.6 Escape

The attribute `draw:caption-escape` specifies the escape point of the caption line measured from the top left corner of the text area. The value can be an absolute length or a percentage.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-escape">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.21.7 Line Length

The attribute `draw:caption-line-length` specifies the length of the first caption line (i.e., the one that starts at the caption's text area). The attribute is only evaluated if `draw:caption-fit-line-length` has the value `false`.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-line-length">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.21.8 Fit Line Length

If the attribute `draw:caption-fit-line-length` is `true`, the application determines the best possible length for the caption line.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:caption-fit-line-length">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.22 3D Geometry Properties

The 3D geometry properties described in this section are applicable to 3D drawing objects. These properties are available for 3D drawing objects contained in all kinds of applications.

15.22.1 Horizontal Segments

If the geometry of a 3D object is generated during run-time, the `dr3d:horizontal-segments` attribute is used to specify the number of horizontal segments that are used to generate the geometry. Typical applications support values between 2 and 256.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:horizontal-segments">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.22.2 Vertical Segments

If the geometry of a 3D object is generated during run-time, the `dr3d:vertical-segments` attribute is used to specify the number of vertical segments that are used to generate the geometry. Typical applications support values between 2 and 256.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:vertical-segments">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
```

```
</define>
```

15.22.3 Edge Rounding

If the geometry of a 3D object is generated during run-time, the `dr3d:edge-rounding` attribute is used to specify the size of an area at the edges of the geometry that is used for rounding the edges.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:edge-rounding">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.22.4 Edge Rounding Mode

The attribute `dr3d:edge-rounding-mode` specifies how to generate rounded edges.

The value of this attribute can be `correct` or `attractive`. If the value is `correct`, the mathematically correct method is used. If the value is `attractive`, a method which preserves the visual appearance of the text is used.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:edge-rounding-mode">
            <choice>
                <value>correct</value>
                <value>attractive</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.22.5 Back Scale

The attribute `dr3d:back-scale` specifies the proportion of the background geometry for lathe and extrude objects.

For example, with a back scale of 50%, the background plane of an extrude object is half the size of the foreground plane.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:back-scale">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.22.6 Depth

The `dr3d:depth` attribute specifies the extrusion depth for extrude objects.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:depth">
            <ref name="length"/>
        </attribute>
    </optional>
</define>
```

15.22.7 Backface Culling

The `dr3d:backface-culling` attribute enables or disables backface culling.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:backface-culling">
            <choice>
                <value>enabled</value>
                <value>disabled</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.22.8 End Angle

The attribute `dr3d:end-angle` specifies the rotation angle for 3D lathe objects. If it is the default (360°), the lathe object is closed and completely rotated. With smaller values it is possible to define opened lathe objects (segments). The then visible sides are closed and take into account the `dr3d:back-scale` and `dr3d:edge-rounding` attributes. With bigger values it is possible to create lathe objects with more than one rotation. This will only have a visible effect when e.g., `dr3d:back-scale` is used.

For example, with a end angle of 270°, the lathe object will be opened by 90°.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:end-angle">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.22.9 Close Front

The `dr3d:close-front` property specifies whether a front plane shall be generated. E.g., if an ellipse is extruded, and this attribute is set, the ellipse will have an open front. The attribute can be used with extrudes and lathe objects.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:close-front">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.22.10 Close Back

The `dr3d:close-back property` describes if a back plane shall be generated. E.g., if an ellipse is extruded, and this attribute is set, the ellipse will have an open back. The attribute can be used with extrudes and lathe objects.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:close-back">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.23 3D Lighting Properties

The 3D lightning properties described in this section are applicable to 3D drawing objects. These properties are available for 3D drawing objects contained in all kinds of applications.

15.23.1 Mode

The attribute `dr3d:lighting-mode` determines the lighting algorithm used to render the corresponding 3D object.

The value of this attribute can be `standard` or `double-sided`. If the value is `double-sided`, the reverse sides of the objects are also lighted.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:lighting-mode">
            <choice>
                <value>standard</value>
                <value>double-sided</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.23.2 Normals Kind

The attribute `dr3d:normals-kind` specifies how the normal settings for the generated lighting.

- `object:` does not produce standard normals, but leaves the object-specific ones untouched.
- `flat:` forces one normal per flat part
- `sphere:` forces normals to behave as the object would be a sphere.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:normals-kind">
            <choice>
                <value>object</value>
                <value>flat</value>
                <value>sphere</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.23.3 Normals Direction

The `dr3d:normals-direction` attribute is used to inverse the generated normal lighting settings.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:normals-direction">
            <choice>
                <value>normal</value>
                <value>inverse</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.24 3D Texture Properties

The 3D texture properties described in this section are applicable to 3D drawing objects. These properties are available for 3D drawing objects contained in all kinds of applications.

15.24.1 Generation Mode

The attributes `dr3d:texture-generation-mode-x and dr3d:texture-generation-mode-y` specify how the texture coordinates are generated.

- `object:` This value specifies that the standard object projection method is used
- `parallel:` This value specifies a flat parallel projection in the specified degree of freedom (X or Y).
- `sphere:` This value forces projection to wrapping in X and/or Y direction

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:texture-generation-mode-x">
            <choice>
                <value>object</value>
                <value>parallel</value>
                <value>sphere</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:texture-generation-mode-y">
            <choice>
                <value>object</value>
                <value>parallel</value>
                <value>sphere</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.24.2 Kind

The attribute `dr3d:texture-kind` is used to select whether the texture changes the luminance, intensity, or color of the shape.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:texture-kind">
            <choice>
                <value>luminance</value>
                <value>intensity</value>
                <value>color</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.24.3 Filter

The attribute `dr3d:texture-filter` is used to enable or disable texture filtering.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:texture-filter">
            <choice>
                <value>enabled</value>
                <value>disabled</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.24.4 Mode

The attribute `dr3d:normals-direction` is used to specify how the texture is modulated.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:texture-mode">
            <choice>
                <value>replace</value>
                <value>modulate</value>
                <value>blend</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.25 3D Material Properties

The 3D texture properties described in this section are applicable to 3D drawing objects. These properties are available for 3D drawing objects contained in all kinds of applications.

15.25.1 Colors

The attributes `dr3d:ambient-color, dr3d:emissive-color, dr3d:specular-color and dr3d:diffuse-color` specify the four colors that define a material.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:ambient-color">
            <ref name="color"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:emissive-color">
            <ref name="color"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:specular-color">
            <ref name="color"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="dr3d:diffuse-color">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.25.2 Shininess

The attribute `dr3d:shininess` specifies the shine of the used material.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:shininess">
            <ref name="percent"/>
        </attribute>
    </optional>
</define>
```

15.26 3D Shadow Properties

The 3D shadow properties described in this section are applicable to 3D drawing objects. These properties are available for 3D drawing objects contained in all kinds of applications.

15.26.1 Shadow

The attribute `dr3d:shadow` enables or disables a three-dimensional shadow for a three-dimensional object.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="dr3d:shadow">
            <choice>
                <value>visible</value>
                <value>hidden</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27 Frame Formatting Properties

The properties described in this section apply to draw frames (see section 9.3). They can be used within graphic styles (see section 14.13.1) and they are contained in a `<style:graphic-properties>` element.

15.27.1 Frame Widths

There are three types of frame widths; fixed widths, minimum widths and relative widths. Fixed widths are specified using the `svg:width` attribute, minimum widths are specified using the `fo:min-width` attribute and relative widths are specified using the `style:rel-width` attribute.

The meaning of these attributes is the same as described in section 9.3, except that the attributes specify the default width for new created frames only. The `style:rel-width` attribute will be evaluated only for graphic styles that are applied to text boxes.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-draw-rel-size-attlist"/>
    <optional>
        <attribute name="fo:min-width">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.2 Frame Heights

There are three types of frame heights; fixed heights, minimum heights and relative heights. Fixed heights are specified using the `svg:height` attribute, minimum heights are specified using the `fo:min-height` attribute and relative heights are specified using the `style:rel-height` attribute. The meaning of these attributes is the same as described in section 9.3, except that the attributes specify the default height for new created frames only. The `style:rel-height` attribute will be evaluated only for graphic styles that are applied to text boxes. See also section 15.27.1.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:min-height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.3 Maximum Width and Height

Text boxes can increase in size automatically when content is added. The `fo:max-width` and `fo:max-height` attributes specify a maximum width and height for the frame. When the maximum values are reached, the frame stops increasing in size. The attributes' value can be either a length or a percentage. If the anchor for the text box is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the text box is in a text box, the percentage value relates to the surrounding text box. In other cases, the percentage values relate to the height of the page or window.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:max-height">
            <choice>
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="fo:max-width">
            <choice>
```

```
                <ref name="length"/>
                <ref name="percent"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.4 Left and Right Margins

The `fo:margin-left` and `fo:margin-right` properties determine the left and right margins to set around a frame. See sections 15.5.17 for detailed information on these attributes. Percentage values are not supported.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-horizontal-margin-attlist"/>
</define>
```

15.27.5 Top and Bottom Margins

The `fo:margin-top` and `fo:margin-bottom` properties determine the top and bottom margins to set around a frame. See sections 15.5.20 for detailed information on these attributes. Percentage values are not supported.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-vertical-margin-attlist"/>
</define>
```

15.27.6 Margins

The `fo:margin` property specifies the the margin for all four edges of a frame. See section 15.5.21 for a full explanation of this property.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-margin-attlist"/>
</define>
```

15.27.7 Print Content

The `style:print-content` property specifies whether or not the content of a frame is printed.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:print-content">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.27.8 Protect

The `style:protect` property specifies whether the content, size, or position of a frame is protected. 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`.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:protect">
            <choice>
```

```
                <value>none</value>
                <list>
                    <oneOrMore>
                        <choice>
                            <value>content</value>
                            <value>position</value>
                            <value>size</value>
                        </choice>
                    </oneOrMore>
                </list>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.9 Horizontal Position

Within text documents, the `style:horizontal-pos` property specifies the horizontal alignment of the frame in relation to the specific area.

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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:horizontal-pos">
            <choice>
                <value>left</value>
                <value>center</value>
                <value>right</value>
                <value>from-left</value>
                <value>inside</value>
                <value>outside</value>
                <value>from-inside</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:x">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

The following tables display the possible values of the attributes `style:horizontal-pos` and `style:horizontal-rel`. 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			
paragraph			X	X	
paragraph-content			X	X	
paragraph-start-margin			X	X	
paragraph-end-margin			X	X	
char				X	

15.27.10 Horizontal Relation

The `style:horizontal-rel` property specifies the area to which the horizontal position of a frame relates. See section 15.27.9 for information on the `style:horizontal-pos` property.

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`.

Some values can be used with only certain frame anchor types.

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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
```

```
        <attribute name="style:horizontal-rel">
            <choice>
                    <value>page</value>
                <value>page-content</value>
                <value>page-start-margin</value>
                <value>page-end-margin</value>
                <value>frame</value>
                <value>frame-content</value>
                <value>frame-start-margin</value>
                <value>frame-end-margin</value>
                <value>paragraph</value>
                <value>paragraph-content</value>
                <value>paragraph-start-margin</value>
                <value>paragraph-end-margin</value>
                <value>char</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.11 Vertical Position

The `style:vertical-pos` property specifies the vertical alignment of the frame in relation to a specific area.

The value of this property can be one of the following: `from-top`, `top`, `middle`, `below` or `bottom`. The area that the position relates to is specified by the `style:vertical-rel` property. `top`, `middle` and `bottom` specify the the given corners of the frame and the reference area get aligned. `below` specifies that the top corner of the frame is positioned below the reference area.

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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-vertical-pos-attlist"/>
</define>

<define name="common-vertical-pos-attlist">
    <optional>
        <attribute name="style:vertical-pos">
            <choice>
                <value>top</value>
                <value>middle</value>
                <value>bottom</value>
                <value>from-top</value>
                <value>below</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="svg:y">
            <ref name="coordinate"/>
        </attribute>
    </optional>
</define>
```

The following tables display the possible values of the attributes `style:vertical-pos` and `style:vertical-rel`. 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: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
text					X

15.27.12 Vertical Relation

The `style:vertical-rel` property specifies the area to which the vertical position of a frame relates. See section 15.27.11 for information on the `style:vertical-pos` property.

The value of this property can be one of the following: `page`, `page-content`, `frame`, `frame-content`, `paragraph`, `paragraph-content`, `line`, `baseline`, `text` or `char`.

Some values can be used with only certain frame anchor types.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-vertical-rel-attlist"/>
</define>

<define name="common-vertical-rel-attlist">
    <optional>
        <attribute name="style:vertical-rel">
            <choice>
                <value>page</value>
                <value>page-content</value>
                <value>frame</value>
                <value>frame-content</value>
                <value>paragraph</value>
                <value>paragraph-content</value>
                <value>char</value>
                <value>line</value>
                <value>baseline</value>
                <value>text</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.13 Frame Anchor

The text:anchor-type and text:anchor-page-number specify the default anchor for new frames and drawing objects. See section 9.2.16 for details.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-text-anchor-attlist"/>
</define>
```

15.27.14 Border

The border attributes `fo:border`, `fo:border-top`, `fo:border-bottom`, `fo:border-left` and `fo:border-right` specify the border properties of the frame. See section 15.5.25 for detailed information on these attributes.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-border-attlist"/>
</define>
```

15.27.15 Border Line Width

If a frame has borders, the border line width attributes `style:border-line-width`, `style:border-line-width-top`, `style:border-line-width-bottom`, `style:border-line-width-left` and `style:border-line-width-right` specify the properties of the border lines of the frame. See section 15.5.26 for detailed information on these attributes.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-border-line-width-attlist"/>
</define>
```

15.27.16 Padding

The padding attributes `fo:padding`, `fo:padding-top`, `fo:padding-bottom`, `fo:padding-left` and `fo:padding-right` specify the padding properties of the frame. See section 15.5.27 for detailed information on these attributes.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-padding-attlist"/>
</define>
```

15.27.17 Shadow

The shadow attribute `style:shadow` specifies the shadow of the frame. See section 15.5.28 for detailed information on this attribute.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-shadow-attlist"/>
</define>
```

15.27.18 Background

The background attribute `fo:background-color` and the background element `<style:background-image>` specify the background properties of the frame. See sections 15.5.23 and 15.5.24 for detailed information on this attribute and element.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-background-color-attlist"/>
</define>
<define name="style-graphic-properties-elements" combine="interleave">
    <ref name="style-background-image"/>
</define>
```

15.27.19 Columns

The `<style:columns>` element specifies if a text box contains columns. See section 15.7.3 for detailed information on this element.

```
<define name="style-graphic-properties-elements" combine="interleave">
    <ref name="style-columns"/>
</define>
```

15.27.20 Editable

Within text documents, a text box can be editable even if the document in which it is contained is a read-only document. The `style:editable` property specifies if a text box can be edited.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:editable">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.27.21 Wrapping

Within text documents, the `style:wrap` property specifies how text around a frame or graphic object 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. The possible values are:

- `none`: no text wraps around the drawing shape.
- `left`: Text may wrap around the left side of the drawing shape.
- `right`: Text may wrap around the left side of the drawing shape.

- `parallel`: Text may wrap around both sides of the drawing shape.
- `dynamic`: Text may wrap around both sides of the drawing shape, provided that there is sufficient space left.
- `biggest`: Text may wraps around the object border where the difference to the left or right page or column border is largest.
- `run-through`: Text runs through the drawing object.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:wrap">
            <choice>
                <value>none</value>
                <value>left</value>
                <value>right</value>
                <value>parallel</value>
                <value>dynamic</value>
                <value>run-through</value>
                <value>biggest</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.22 Dynamic Wrap Threshold

The `style:wrap-dynamic-threshold` attribute is evaluated only if the `style:wrap` attribute has a value of `dynamic`. It specifies the minimum distance between the page or column border and the object for which wrapping will be enabled.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:wrap-dynamic-threshold">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.27.23 Paragraph-only Wrapping

If the anchor position of a frame or drawing shape is a paragraph or a character, and the wrap mode specified by the `style:wrap` property is `left`, `right`, `parallel`, or `dynamic`, the number of paragraphs that wrap around the frame can be specified using a `style:number-wrapped-paragraphs` attribute.

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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:number-wrapped-paragraphs">
            <choice>
                <value>no-limit</value>
                <ref name="positiveInteger"/>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.24 Contour Wrapping

Within text documents, the `style:wrap-contour` attribute specifies for some frame types 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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:wrap-contour">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.27.25 Contour Wrapping Mode

The `style:wrap-contour-mode` attribute is used to further specify how the text should wrap around the contour.

This attribute is recognized only by frames/drawing shapes 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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:wrap-contour-mode">
            <choice>
                <value>full</value>
                <value>outside</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.26 Run Through

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

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.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:run-through">
            <choice>
                <value>foreground</value>
                <value>background</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.27 Flow with Text

The `style:flow-with-text` attribute specifies the behavior of drawing shapes that are positioned at a certain distance below an anchor and do not fit on the page where the anchor is. If the value of the property is `true`, such drawing objects follow the text flow, that is, they a displayed on the next page. If the attribute value is `false`, such drawing objects are displayed outside the page's text area.

Example: A graphic is to be positioned 10cm below its anchor. It is followed by only 8cm of text before the next page break. With `style:flow-with-text='false'` the graphics would then be positioned 2cm below the text area (somewhere in the footer); with `style:flow-with-text='true'` it would positioned 2cm into the text flow of the following page.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:flow-with-text">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.27.28 Overflow behavior

For text boxes contained within text document, the `style:overflow-behavior` property specifies the behavior of text boxes where the containing text does not fit into the text box. If the attribute's value is `clip`, the text that does not fit into the text box is not displayed. If the attribute value is `auto-create-new-frame`, a new frame will be created on the next page, with the same position and dimensions of the original frame.

If the `style:overflow-behavior` property's value is `auto-create-new-frame` and the text box has a minimum width or height specified, then the text box will grow until the page bounds are reached before a new frame is created.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:overflow-behavior">
            <choice>
                <value>clip</value>
                <value>auto-create-new-frame</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.27.29 Mirroring

The `style:mirror` property 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.

The value of this attribute can be `none`, `vertical`, `horizontal`, `horizontal-on-odd`, or `horizontal-on-even`. The value `vertical` and the various horizontal values can be specified together, separating them by a white space.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="style:mirror">
            <choice>
                <value>none</value>
                <value>vertical</value>
                <ref name="horizontal-mirror"/>
                <list>
                    <value>vertical</value>
                    <ref name="horizontal-mirror"/>
                </list>
                <list>
                    <ref name="horizontal-mirror"/>
                    <value>vertical</value>
                </list>
            </choice>
        </attribute>
    </optional>
</define>

<define name="horizontal-mirror">
    <choice>
        <value>horizontal</value>
        <value>horizontal-on-odd</value>
        <value>horizontal-on-even</value>
    </choice>
</define>
```

15.27.30 Clipping

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

- A rectangular section of an image, or
- the entire image.

See §7.20.1 of [XSL] for details.

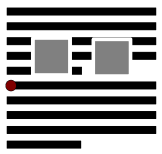
```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="fo:clip">
            <!-- The attribute value must match the one XSL's clip -->
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.27.31 Wrap Influence on Position

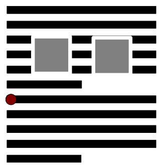
This attribute details how the wrapping mode (see the `style:wrap` attribute) influences the positioning of a frame. It is intended as a hint to the layout algorithm to help decide on the placement of frames in certain cases where several correct placements could be used. All three options describe different, correct interpretations of the layout constraints already in the format. The new hint would allow to disambiguate between these situations.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:wrap-influence-on-position"
                    a:defaultValue="iterative">
            <choice>
                <value>iterative</value>
                <value>once-concurrent</value>
                <value>once-successive</value>
            </choice>
        </attribute>
    </optional>
</define>
```

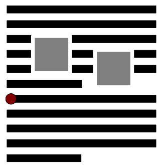
The situation in which this attribute makes a difference is when the anchor, position and wrapping mode of a frame are such that they influence each other. For example, consider a paragraph of text with two images positioned somewhat above the anchor. Without wrapping, the images overly the text and can simply be placed at the given offset from the anchor.



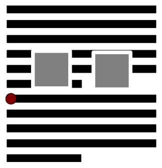
If wrap-around is enabled, the text hidden behind the images now needs to flow around the images, making the first paragraph use more space than previously. This moves the anchor position further down. If one does the placement only once and concurrently for all objects, this is the final result. This corresponds to the object `once-concurrently`.



If one proceeds as above, but does the process one image at a time, one arrives at the positions given to the right. This corresponds to the option `once-successive`.



If one places the images iteratively, until a position is found which corresponds to the given offset from the anchor, one can often achieve a placement that fully satisfy all the given layout properties (at a certain price in implementation cost). This corresponds to the option `iterative`.



15.27.32 Writing Mode

The `style:writing-mode` attribute specifies the writing mode for a text-box. See section 15.5.36 for details.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <ref name="common-writing-mode-attlist"/>
</define>
```

15.28 Floating Frame Formatting Properties

The attributes described in this section can be assigned to a graphic style that is assigned to floating frames.

15.28.1 Display Scrollbar

The `draw:display-scrollbar` attribute specifies whether or not vertical and horizontal scrollbars are displayed. This attribute can be assigned to automatic styles only.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:frame-display-scrollbar">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.28.2 Display Border

The draw:display-border attribute specifies whether or not a border is displayed on the floating frame. This attribute can be assigned to automatic styles only.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:frame-display-border">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.28.3 Margins

The draw:margin-horizontal and draw:margin-vertical attributes specify the horizontal and vertical margins between the border and the content of the floating frame. If these attributes are not specified, the default margins are used. These attributes can be assigned to automatic styles only. The value of these attributes must be a length in pixels.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:frame-margin-horizontal">
            <ref name="nonNegativePixelLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:frame-margin-vertical">
            <ref name="nonNegativePixelLength"/>
        </attribute>
    </optional>
</define>

<define name="nonNegativePixelLength">
    <data type="string">
        <param name="pattern">([0-9]+(\.[0-9]*)?|\.[0-9]+)(px)</param>
    </data>
</define>
```

15.28.4 Object Formatting Properties

The attributes described in this section can be assigned to a graphic style that is assigned to objects.

15.28.5 Visible Area

The visible area of an object is the rectangular area of the object that is currently visible. The attributes draw:visible-area-left, draw:visible-area-top, draw:visible-area-width and draw:visible-area-height specify a default visible area that the object has the option to use.

When the entire object is visible, the values of the draw:visible-area-left and draw:visible-area-top attributes are 0 and the draw:visible-area-width and draw:visible-area-height attributes specify the size of the object. These attributes can be assigned to automatic styles only.

Not all objects support these attributes. Some objects, may store and load their own visible area.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:visible-area-left">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:visible-area-top">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:visible-area-width">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="draw:visible-area-height">
            <ref name="positiveLength"/>
        </attribute>
    </optional>
</define>
```

15.28.6 Draw Aspect

For embedded OLE objects, the draw:ole-draw-aspect attribute specifies the draw aspect that is used to display embedded objects (see [OLE]). The draw aspect controls whether the object is displayed as a normal sub document, or whether the object is for instance displayed as an icon only. Within the [OLE] API, the draw aspect is an unsigned integer value that the host application passes to the object when it requests its presentation.

The `draw:ole-draw-aspect` attribute takes a non negative integer value and has only a meaning for objects that are embedded using the [OLE] API. In this case, its value specifies a default value for method calls that require a draw aspect. The interpretation of this integer value is left to the OLE object's discretion and not part of this specification.

```
<define name="style-graphic-properties-attlist" combine="interleave">
    <optional>
        <attribute name="draw:ole-draw-aspect">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.29 Chart Formatting Properties

The properties described in this section can be applied to all charts. They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

```
<define name="style-chart-properties">
    <element name="style:chart-properties">
        <ref name="style-chart-properties-content"/>
    </element>
</define>

<define name="style-chart-properties-content">
    <ref name="style-properties-content"/>
</define>
```

```
<define name="style-chart-properties-content-strict">
    <ref name="style-chart-properties-attlist"/>
    <ref name="style-chart-properties-elements"/>
</define>

<define name="style-chart-properties-elements">
    <empty/>
</define>
```

15.29.1 Scale Text

The `chart:scale-text property` is used to specify that all text objects in the chart should be scaled whenever the size of the chart changes. To enable scaling, set the value of this property to `true`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:scale-text" a:defaultValue="true">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.30 Chart Subtype Properties

The properties described in this section can be used to customize the basic chart type set in the `<chart:chart>` element. They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

15.30.1 Three-dimensional Charts

The `chart:three-dimensional` property specifies whether chart is displayed as a 3D scene.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:three-dimensional">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.30.2 Chart Depth

The `chart:deep` property is only relevant with the `chart:three-dimensional` property. It specifies that the data series are displayed back-to-back rather than side by side.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:deep">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.30.3 Chart Symbol

For some chart types, the data points can be denoted by symbols. The `chart:symbol-type` attribute determines whether a symbol is used, and whether it is a pre-defined symbol type, an image, or whether the application is free to automatically choose a type out of the set of pre-defined symbol types, e.g., choose one symbol per series in round-robin fashion.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <choice>
        <attribute name="chart:symbol-type">
            <value>none</value>
        </attribute>
        <attribute name="chart:symbol-type">
            <value>automatic</value>
        </attribute>
        <group>
            <attribute name="chart:symbol-type">
                <value>named-symbol</value>
            </attribute>
            <attribute name="chart:symbol-name">
                <choice>
                    <value>square</value>
                    <value>diamond</value>
                    <value>arrow-down</value>
                    <value>arrow-up</value>
                    <value>arrow-right</value>
                    <value>arrow-left</value>
                    <value>bow-tie</value>
                    <value>hourglass</value>
                    <value>circle</value>
                    <value>star</value>
                    <value>x</value>
                    <value>plus</value>
                    <value>asterisk</value>
                    <value>horizontal-bar</value>
                    <value>vertical-bar</value>
                </choice>
            </attribute>
        </group>
        <group>
            <attribute name="chart:symbol-type">
                <value>image</value>
            </attribute>
            <element name="chart:symbol-image">
                <attribute name="xlink:href">
                    <ref name="anyURI"/>
                </attribute>
            </element>
        </group>
        <empty/>
    </choice>
</define>
```

15.30.4 Chart Symbol Size

The width and height of each symbol can be set using the attribute `chart:symbol-width` and `chart:symbol-length`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:symbol-width">
```

```
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:symbol-height">
            <ref name="nonNegativeLength"/>
        </attribute>
    </optional>
</define>
```

15.30.5 Bar Chart Properties

The `chart:vertical` and `chart:connect-bars` properties are for bar charts only. `chart:vertical` determines whether the bars will be oriented horizontally or vertically. If `chart:connect-bars` is set to true, the data points (the top of the bars) are additionally connected by lines.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:vertical" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:connect-bars" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

With bar charts, the properties `chart:gap-width` and `chart:overlap` can be used to specify the relative size and distance of bars. The `chart:gap-width` attribute contains the relative width of the gap between bars for neighboring categories. The `chart:overlap` attributes determines how much bars within the same category overlap. Both are integral percentages.

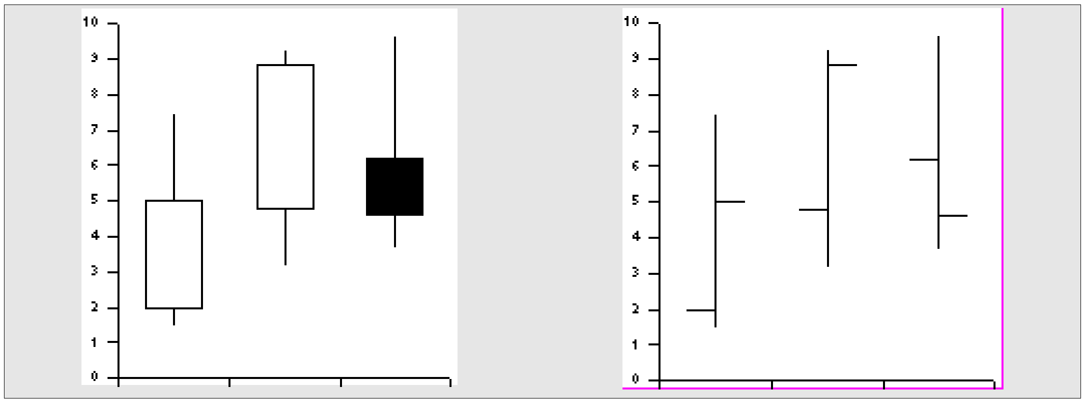
```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:gap-width">
            <ref name="integer"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:overlap">
            <ref name="integer"/>
        </attribute>
    </optional>
</define>
```

15.30.6 Stock Chart Properties

These attributes are only effective for stock charts.

Stock charts display a span from minimum to maximum values as a straight line. Opening and closing courses can be displayed either as left and right tick-lines, respectively, or as colored bars, with their color depending on whether the opening value is larger than the closing value. The `chart:japanese-candle-stick` attribute distinguish between those two representations.

Example: A stock chart in *Japanese-candle-stick* fashion (left), and as default (right).



```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:japanese-candle-stick"
                    a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.30.7 Line Chart Properties

For line chart-types, the attribute `chart:interpolation` can be set to one of the following values:

- `none` -Straight lines – don't use spline interpolation
- `cubic-spline` - Cubic Splines (`chart:spline-resolution` determines the number of interpolated points between two data points)
- `b-spline` - B-Splines (`chart:spline-order` determines the order of the polygons used for calculation. The `chart:spline-resolution` is also taken into account.)

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:interpolation" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>cubic-spline</value>
                <value>b-spline</value>
            </choice>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:spline-order" a:defaultValue="2">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:spline-resolution" a:defaultValue="20">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.30.8 Pie Chart Properties

The `chart:pie-offset` attribute is only interpreted by pie charts. It determines the offset the tip of a 'pie' in a pie chart (or circle chart) has from the center of the circle.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:pie-offset" a:defaultValue="0">
            <ref name="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

15.30.9 Lines

The `chart:lines` property determines whether connecting lines between data points are shown. The line interpolation is determined by the `chart:splines` property.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:lines" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.30.10 Solid Charts Bars

The `chart:solid-type` attribute determines how the bars in three-dimensional bar charts should be rendered.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:solid-type" a:defaultValue="cuboid">
            <choice>
                <value>cuboid</value>
                <value>cylinder</value>
                <value>cone</value>
                <value>pyramid</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.30.11 Stacked Chart Bars

The attribute `chart:stacked` attribute causes bars in bar charts to be stacked on top of each other, instead of next to each other. If `chart:percentage` is set to true, the stacked bars will all be scaled to the full height of the plot area, so that the bar segments represent the percentage of their respective data point in the total bar stack.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:stacked" a:defaultValue="false">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:percentage" a:defaultValue="false">
```

```
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.31 Chart Axes Properties

The properties described in this section can be applied to chart axis elements (see section 10.8). They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

15.31.1 Linked Data Formats

The `chart:link-data-style-to-source` attribute can only be used in chart documents that reside in a document that provides the data for the chart. If the value of the attribute is true, the number format used for rendering the axis is the format that the container document suggests based on the selected cell range. For example, if a cell range contains currencies all formatted in €, then this format will also be used at this axis.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:link-data-style-to-source">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.31.2 Visibility

To determine whether or not an axis object is visible, use the `chart:axis-visible` style property. This way, a chart with scaling information can be provided without displaying the axis object.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:visible">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.31.3 Scaling

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`.

The optional `chart:axis-logarithmic` attribute can be used to cause logarithmic scaling on an axis. By default, proportional scaling is used.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:logarithmic">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

The following set of optional attributes further details the scaling of an axis. The properties have the following uses:

`chart:minimum`, `chart:maximum` – set minimal and maximal scaling values of an axis

`chart:origin` – determine the origin of the chart axis

`chart:interval-major`, `chart:interval-minor-divisor` – set major and minor interval for ticks or markings on the axis. The `chart:interval-major` defines the interval value. The minor interval is determined by dividing the `chart:interval-major` value by the `chart:interval-minor-divisor`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:maximum">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:minimum">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:origin">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:interval-major">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:interval-minor-divisor">
            <ref name="positiveInteger"/>
        </attribute>
    </optional>
</define>
```

15.31.4 Tick Marks

The tick mark properties are used to specify the existence of tick marks 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.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:tick-marks-major-inner">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:tick-marks-major-outer">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:tick-marks-minor-inner">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:tick-marks-minor-outer">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.31.5 Labels

The following set of properties describes how axis labels are being represented. `chart:display-label` determines whether labels will be displayed at all. If `chart:text-overlap` is set true, labels may overlap. `text:line-break` determines whether label lines may be broken into multiple lines.

The `chart:label-arrangement` property allows labels to be arranged either `side-by-side` (i.e., all labels start on one line), or staggered (i.e., labels are distributed to two lines, with every other label starting on the same line). In case of staggered labels, one can choose between even or odd staggering, i.e., one can choose whether even or odd labels are aligned on the line that would be used for `side-by-side` arrangement.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:display-label">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:text-overlap">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:line-break">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:label-arrangement"
                   a:defaultValue="side-by-side">
            <choice>
                <value>side-by-side</value>
                <value>stagger-even</value>
                <value>stagger-odd</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.32 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>`. They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

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, set the property in the `<chart:data-point>` element.

15.32.1 Stacked Text

The property `style:direction` determines whether or not text is displayed vertically without rotating the letters. It can be applied to several text objects.

The value of this property can be `ltr` if text goes from left to right or `ttb` if the text is stacked, that is goes from top to bottom. It can be applied to several text objects. See section 15.11.3 for details.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <ref name="common-style-direction-attlist"/>
</define>
```

15.32.2 Rotation Angle

The `style:rotation-angle` property specifies the value of a rotation angle in degrees. See section 15.11.12 for information on using this property.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <ref name="common-rotation-angle-attlist"/>
</define>
```

15.32.3 Data Labels

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

The `chart:data-label-number` attribute represents the value of the data label.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:data-label-number">
            <choice>
                <value>none</value>
                <value>value</value>
                <value>percentage</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Label

The `chart:data-label-text` attribute determines whether or not to display the label of the corresponding series.

The value of this attribute can be `true` or `false`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:data-label-text">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

Legend Symbol

The `chart:data-label-symbol` attribute determines whether or not to display the legend symbol. The value of this attribute can be `true` or `false`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:data-label-symbol">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.33 Statistical Properties

Statistical properties can be applied to data series or to an entire chart. In the latter case, the properties apply to all series in the chart. They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

15.33.1 Mean Value

The `chart:mean-value` attribute determines whether or not to display a line that represents the statistical mean value of all data points of a series. The value of this attribute can be `true` or `false`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:mean-value">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.33.2 Error Category

The `chart:error-category` attribute is used to determine which function is used to display error indicators at data points. The following functions are available:

- Variance of the values of a series assuming an equal distribution.
- Standard-deviation of the values of a series assuming an equal distribution.
- Use a fixed percentage of each value

- Use a fixed percentage of the biggest value – this is called error-margin.
- Use fixed absolute values for both directions: positive and negative

If this attribute is set to any value other than `none`, error indicators are shown. To determine in which direction the indicators are pointing see the attributes `chart:error-upper-indicator` and `chart:error-lower-indicator`.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:error-category" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>variance</value>
                <value>standard-deviation</value>
                <value>percentage</value>
                <value>error-margin</value>
                <value>constant</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Error Percentage

The `chart:error-percentage` attribute determines the percentage that is used to display error indicators for each data point of a series.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:error-percentage">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Error Margin

The `chart:error-margin` attribute determines the percentage that is used to display error indicators for the biggest value in a series.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:error-margin">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Constant Error Lower and Upper Limit

If the error category is set to `constant`, the `chart:error-lower-limit` and `chart:error-upper-limit` attributes determine the absolute values in a positive and negative direction that are used to display the error indicators.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:error-lower-limit">
            <ref name="double"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:error-upper-limit">
            <ref name="double"/>
        </attribute>
    </optional>
</define>
```

Error Indicators

The `chart:error-lower-indicator` and `chart:error-upper-indicator` attributes determine in which direction indicators should be drawn.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:error-upper-indicator">
            <ref name="boolean"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="chart:error-lower-indicator">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.34 Plot Area Properties

The properties described in this section can be applied to chart plot area elements (see section 10.5). They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

15.34.1 Series Source

The `chart:series-source` attribute determines whether the data table contains the data series in column-wise or row-wise fashion.

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:series-source" a:defaultValue="columns">
            <choice>
                <value>columns</value>
                <value>rows</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.35 Regression Curve Properties

The properties described in this section can be applied to chart regression curves elements (see section 10.14). They can be used within chart styles (see section 14.16) and are contained in a `<style:chart-properties>` element.

15.35.1 Regression Type

Use the `chart:regression-type` attribute to display a regression for a series. A regression can be used to approximate the data points in a series by a mathematical function. The following models for approximation are available:

- Linear regression – approximate the values of the series using the model: $y = A \cdot x + B$.
- Logarithmic regression – approximate the values of the series using the model: $y = A \cdot \log(x) + B$.
- Exponential regression – approximate the values of the series using the model: $y = A \cdot e^{B \cdot x}$.
- Regression with a power function – approximate the values of the series using the model: $y = A \cdot x^B$.

This property is only relevant in scatter charts, because regression needs both *x* and *y* values for calculation

```
<define name="style-chart-properties-attlist" combine="interleave">
    <optional>
        <attribute name="chart:regression-type" a:defaultValue="none">
            <choice>
                <value>none</value>
                <value>linear</value>
                <value>logarithmic</value>
                <value>exponential</value>
                <value>power</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36 Presentation Page Attributes

The properties described in this section can be contained within style elements `<style:style>` whose family is `drawing-page`. They are contained in a `<style:style-drawpage-properties>` element.

The following presentation properties do exist:

- Transition Type
- Transition Style
- Transition Speed
- Page Duration
- Page Visibility
- Sound
- Background Size
- Background Objects Visible
- Background Visible
- Display Header
- Display Footer

- Display Page Number
- Display Date and Time

15.36.1 Transition Type

The mode of transition, for example manual, can be set using the attribute `presentation:transition-type`.

- `manual`: slide transition and shape effects must be started separately by the user.
- `automatic`: slide transition and shape effects start automatically.
- `semi-automatic`: slide transition starts automatically, shape effects must be started by the user.

```
<define name="style-drawing-page-properties-attlist"
        combine="interleave">
    <optional>
        <attribute name="presentation:transition-type">
            <choice>
                <value>manual</value>
                <value>automatic</value>
                <value>semi-automatic</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36.2 Transition Style

The attribute `presentation:transition-style` specifies the way that each presentation page replaces the previous presentation page, for example left-to-right replacement, or fading.

- `none`: no effect is used.
- `fade-*`: the pages fades from a visible or hidden state to a hidden or visible state in the specified direction.
- `move-*`: the page moves in the specified direction to its final position.
- `uncover-*`: the page get uncovered in the specified direction.
- `*-stripes`: the page is uncovered by drawing horizontal or vertical stripes that change their size during this effect.
- `clockwise`: the page is uncovered by the hand of a watch, moving clockwise.
- `counterclockwise`: the page is uncovered by the hand of a watch, moving counterclockwise.
- `open-*`: the page is uncovered by drawing it line by line, either horizontally or vertically, starting at the center of the page.
- `close-*`: the page is uncovered by drawing it line by line, either horizontally or vertically, starting at the edge of the page.
- `wavyline-*`: the page is uncovered by drawing small blocks in a snake like fashion.

- `spiralin-*`: the page is uncovered by drawing blocks in a spiral fashion, starting from the edge of the page.
- `spiralout-*`: the page is uncovered by drawing blocks in a spiral fashion, starting from the center of the page.
- `roll-*`: the pages moves in the specified direction to its final position, pushing the old page out.
- `stretch-*`: the page is uncovered by changing its size during this effect.
- `*-lines`: the page is uncovered by drawing it line by line, either horizontally or vertically in a random fashion.
- `dissolve`: the page is faded in by drawing small blocks in a random fashion.
- `random`: an effect is chosen at random to uncover the page.
- `*-checkerboard`: the page is uncovered by drawing checkerboard like blocks that increase in size horizontally or vertically.
- `interlocking-horizontal-*`: the new page appears in 4 horizontal stripes (i.e., the height is divided in 4, a bit like in the horizontal-stripes effect) but those stripes come from left, right, left, and right, and cross each other in the middle of the screen.
- `interlocking-vertical-*`: similar effect with vertical stripes crossing each other.
- `fly-away`: the page first reduces itself to a smaller size (while remaining centered in the screen), and then "flies away" (turns around a bit and moves to the bottom-right corner of the screen). The next slide appears under it meanwhile.
- `open`: Combination of open-horizontal and open-vertical, i.e., a sort of *plus* sign opening.
- `close`: Combination of close-horizontal and close-vertical, i.e., a sort of *plus* sign closing.
- `melt`: Small vertical stripes move down at random speed, which gives the effect of the current page "melting down".

```
<define name="style-drawing-page-properties-attlist"
         combine="interleave">
    <optional>
        <attribute name="presentation:transition-style">
            <choice>
                <value>none</value>
                <value>fade-from-left</value>
                <value>fade-from-top</value>
                <value>fade-from-right</value>
                <value>fade-from-bottom</value>
                <value>fade-from-upperleft</value>
                <value>fade-from-upperright</value>
                <value>fade-from-lowerleft</value>
                <value>fade-from-lowerright</value>
                <value>move-from-left</value>
                <value>move-from-top</value>
                <value>move-from-right</value>
                <value>move-from-bottom</value>
                <value>move-from-upperleft</value>
                <value>move-from-upperright</value>
                <value>move-from-lowerleft</value>
                <value>move-from-lowerright</value>
                <value>uncover-to-left</value>
                <value>uncover-to-top</value>
                <value>uncover-to-right</value>
                <value>uncover-to-bottom</value>
                <value>uncover-to-upperleft</value>
                <value>uncover-to-upperright</value>
                <value>uncover-to-lowerleft</value>
                <value>uncover-to-lowerright</value>
                <value>fade-to-center</value>
                <value>fade-from-center</value>
                <value>vertical-stripes</value>
                <value>horizontal-stripes</value>
                <value>clockwise</value>
                <value>counterclockwise</value>
                <value>open-vertical</value>
                <value>open-horizontal</value>
                <value>close-vertical</value>
                <value>close-horizontal</value>
                <value>wavyline-from-left</value>
                <value>wavyline-from-top</value>
                <value>wavyline-from-right</value>
                <value>wavyline-from-bottom</value>
                <value>spiralin-left</value>
                <value>spiralin-right</value>
                <value>spiralout-left</value>
                <value>spiralout-right</value>
                <value>roll-from-top</value>
                <value>roll-from-left</value>
                <value>roll-from-right</value>
                <value>roll-from-bottom</value>
                <value>stretch-from-left</value>
                <value>stretch-from-top</value>
                <value>stretch-from-right</value>
                <value>stretch-from-bottom</value>

                <value>vertical-lines</value>
                <value>horizontal-lines</value>
                <value>dissolve</value>
                <value>random</value>
                <value>vertical-checkerboard</value>
                <value>horizontal-checkerboard</value>
                <value>interlocking-horizontal-left</value>
                <value>interlocking-horizontal-right</value>
                <value>interlocking-vertical-top</value>
                <value>interlocking-vertical-bottom</value>
                <value>fly-away</value>
                <value>open</value>
                <value>close</value>
                <value>melt</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36.3 Transition Speed

The attribute `presentation:transition-speed` controls the speed at which a presentation page is removed from display, and replaced by a new presentation page. See also section 9.7.2.

```
<define name="style-drawing-page-properties-attlist"
        combine="interleave">
    <optional>
        <attribute name="presentation:transition-speed">
```

```
            <ref name="presentationSpeeds"/>
        </attribute>
    </optional>
</define>
```

15.36.4 Transition Type or Family

The [SMIL20] `smil:type` attribute is used to specify the transition type or family. See §12.4.1 of [SMIL20] for details. See §12.8 of [SMIL20] for a list of supported types.

If this attribute is present, the attributes `presentation:transition-type` and `presentation:transition-style` attributes should be ignored.

```
<define name="style-drawing-page-properties-attlist " combine="interleave">
    <optional>
        <attribute name="smil:type">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.36.5 Transition Subtype

The [SMIL20] `smil:subtype` attribute is used to specify the transition subtype. See §12.4.1 of [SMIL20] for details. See §12.8 of [SMIL20] for a list of supported subtypes.

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="smil:subtype">
            <ref name="string"/>
        </attribute>
    </optional>
</define>
```

15.36.6 Transition Direction

The [SMIL20] `smil:direction` attribute is used to specify the transition direction. See §12.4.1 of [SMIL20] for details.

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="smil:direction" a:defaultValue="forward">
            <choice>
                <value>forward</value>
                <value>reverse</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36.7 Fade Color

The [SMIL20] `smil:fadeColor` attribute is used to specify the transition fade color for transitions that make use of a start or end color. See §12.4.1 of [SMIL20] for details.

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="smil:fadeColor">
            <ref name="color"/>
        </attribute>
    </optional>
</define>
```

15.36.8 Page Duration

The attribute `presentation:page-duration` controls the amount of time that the presentation page is displayed. The value of this attribute must conform to the time period format described in §3.2.6 of [xmlschema-2].

```
<define name="style-drawing-page-properties-attlist"
        combine="interleave">
    <optional>
        <attribute name="presentation:duration">
            <ref name="duration"/>
        </attribute>
    </optional>
</define>
```

15.36.9 Page Visibility

A drawing page can be marked as hidden during a presentation by using the attribute `presentation:visibility`. A page marked with this attribute is only shown while editing the document but not during the presentation.

```
<define name="style-drawing-page-properties-attlist"
         combine="interleave">
    <optional>
        <attribute name="presentation:visibility">
            <choice>
                <value>visible</value>
                <value>hidden</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36.10 Sound

Sound effects can be added to your presentation pages using the element `presentation:sound`. It must be included in the `<style:presentation-properties>` element.

```
<define name="style-drawing-page-properties-elements"
         combine="interleave">
    <optional>
        <ref name="presentation-sound"/>
    </optional>
</define>
```

15.36.11 Background Size

The attribute `draw:background-size` specifies whether the background of a page is rendered on the full page or only inside the borders of the page.

```
<define name="style-drawing-page-properties-attlist"
         combine="interleave">
    <optional>
```

```
        <attribute name="draw:background-size">
            <choice>
                <value>full</value>
                <value>border</value>
            </choice>
        </attribute>
    </optional>
</define>
```

15.36.12 Background Objects Visible

The attribute `presentation:background-objects-visible` specifies whether or not to hide objects on the background of the master page when displaying the presentation page.

```
<define name="style-drawing-page-properties-attlist"
         combine="interleave">
    <optional>
        <attribute name="presentation:background-objects-visible">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.36.13 Background Visible

The attribute `presentation:background-visible` specifies whether or not to hide the background of the master page when displaying the presentation page.

```
<define name="style-drawing-page-properties-attlist"
         combine="interleave">
    <optional>
        <attribute name="presentation:background-visible">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.36.14 Display Header

The `presentation:display-header` attribute sets the visibility of presentation shapes from the master page with the presentation class `header` (see section 9.6.1).

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:display-header">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.36.15 Display Footer

The `presentation:display-footer` attribute sets the visibility of presentation shapes from the master page with the presentation class `footer` (see section 9.6.1).

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:display-footer">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.36.16 Display Page Number

The `presentation:display-page-number` attribute sets the visibility of presentation shapes from the master page with the presentation class `page-number` (see section 9.6.1).

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:display-page-number">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.36.17 Display Date And Time

The `presentation:display-date-time` attribute sets the visibility of presentation shapes from the master page with the presentation class date-`time` (see section 9.6.1).

```
<define name="style-drawing-page-properties-attlist" combine="interleave">
    <optional>
        <attribute name="presentation:display-date-time">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

16 Data Types and Schema Definitions

16.1 Data Types

The following data types are used within this specification:

- W3C Schema data types as defined in [xmlschema-2] (referenced by `<ref>` elements named the same as the corresponding data types)
  - string
  - date
  - time
  - dateTime
  - duration
  - integer
  - nonNegativeInteger
  - positiveInteger
  - double
  - anyURI
  - base64Binary
  - ID
  - IDREF
  - IDREFS

Relax-NG definitions for the W3C schema data types:

```
<define name="string">
    <data type="string"/>
</define>
<define name="date">
    <data type="date"/>
</define>
<define name="time">
    <data type="time"/>
</define>
<define name="dateTime">
    <data type="dateTime"/>
</define>
<define name="duration">
    <data type="duration"/>
</define>
<define name="integer">
    <data type="integer"/>
</define>
<define name="nonNegativeInteger">
    <data type="nonNegativeInteger"/>
```

```
</define>
<define name="positiveInteger">
    <data type="positiveInteger"/>
</define>
<define name="double">
    <data type="double"/>
</define>
<define name="anyURI">
    <data type="anyURI"/>
</define>
<define name="base64Binary">
    <data type="base64Binary"/>
</define>
<define name="ID">
    <data type="ID"/>
</define>
<define name="IDREF">
    <data type="IDREF"/>
</define>
<define name="IDREFS">
    <data type="IDREFS"/>
</define>
```

- custom data types (usually specializations of W3C Schema data types)
  - boolean

    A Boolean value may have either of the values `true` or `false`.
  - dateOrDateTime

    A dateOrDateTime value is essentially an [xmlschema-2] date and time value with an optional time component. In other words, it may contain either a date, or a date and time value.
  - timeOrDateTime

    A timeOrDateTime value is essentially an [xmlschema-2] date and time value with an optional date component. In other words, it may contain either a time, or a date and time value.
  - language

    A language is the same as an [xmlschema-2] language data type.
  - countryCode

    A countryCode is a country code in conformance with [RFC3066], as specified in [XSL].
  - languageCode

    A languageCode is a language code in conformance with [RFC3066], as specified in [XSL].
  - character

    A character value is a string with only one character.
  - length

    A (positive or negative) physical length, consisting of magnitude and unit, in conformance with §5.9.11 of [XSL]. Supported units are „cm“, „mm“, „in“, „pt“ and „pc“. Applications **shall** support all these units. Applications **may** also support “px” (pixel). Where the description of an attribute explicitly states that pixel lengths are supported, applications **should** support them.

Examples for valid lengths are “2.54cm” and “1in”.

– nonNegativeLength

  Like length, except that the value must be zero or positive.

– positiveLength

  Like length, except that the value must be positive.

– percent

  (Positive or negative) percentage values in conformance with §5.9.11 of [XSL], e.g., “40%”.

– relativeLength

  A relative length is a positive integer, followed by a '*' character.

– coordinate

  Like a length, except that the physical length denotes a certain point.

– distance

  Like a length, except that the physical length measures the distance between to points.

– color

  A RGB color in conformance with §5.9.11 of [XSL], that is a RGB color in notation “#rrggbb”, where rr, gg and bb are hexadecimal digits.

– styleName

  A NCName as specified in [xmlschema-2] that is the name of a style.

– StyleNameRef

  A NCName as specified in [xmlschema-2] that is the name of a referenced style, or an empty value.

– StyleNames

  A whitespace separated list of NCNames as specified in [xmlschema-2] that are the names of a styles.

– VariableName

  A string specifying the name of a variable

– formula

  A string containing a formula. Formulas don't have a predefined syntax, but should start with a namespace prefix that specifies the syntax used within the formula.

– valueType

  A list of value types supported for certain generic values, such as “string” or “date”.

– targetFrameName

  The name of a target frame in conformance with §6.16 of [HTML4].

– points

  A sequence of points. The points are two integer coordinates separated by a comma. The points are separated by white space.

– pathData

  Path data as described in §8 of [SVG].

– vector3D

  A 3-element vector that is represented by floating point x,y,z coordinates. The coordinates are encapsulated between parentheses and the coordinates are noted in the order x, y and z, separated by whitespaces. If this value represents a normal, then it should be normalized.

  Example: A directional vector with the coordinates x = 0.5, y = 0 and z = 1 looks like "(0.5 0 1)".

– namespacedToken

  A namespaced token is a token id that makes use of the XML namespace mechanism for modularization purposes.

  Example: The predefined chart types make use of the chart namespace urn:oasis:names:tc:opendocument:xmlns:`chart`:1.0. Assuming a namespace declaration of `xmlns:chart="`urn:oasis:names:tc:opendocument:xmlns:`chart`:1.0`"`, a bar chart would be identified as `chart:bar`.

Relax-NG definitions for custom data types:

```
<define name="boolean">
    <choice>
        <value>true</value>
        <value>false</value>
    </choice>
</define>
<define name="dateOrDateTime">
    <choice>
        <data type="date"/>
        <data type="dateTime"/>
    </choice>
</define>
<define name="timeOrDateTime">
    <choice>
        <data type="time"/>
        <data type="dateTime"/>
    </choice>
</define>
<define name="language">
    <data type="language"/>
</define>
<define name="countryCode">
    <data type="token">
        <param name="pattern">[A-Za-z0-9]{1,8}</param>
    </data>
</define>
<define name="languageCode">
    <data type="token">
        <param name="pattern">[A-Za-z]{1,8}</param>
    </data>
</define>
<define name="character">
```

```
    <data type="string">
        <param name="length">1</param>
    </data>
</define>
<define name="length">
    <data type="string">
        <param name="pattern">-?([0-9]+(\.[0-9]*)?|\.[0-9]+)((cm)|(mm)|(in)|
(pt)|(pc)|(px))</param>
    </data>
</define>
<define name="nonNegativeLength">
    <data type="string">
        <param name="pattern">([0-9]+(\.[0-9]*)?|\.[0-9]+)((cm)|(mm)|(in)|(pt)|
(pc)|(px))</param>
    </data>
</define>
<define name="positiveLength">
    <data type="string">
        <param name="pattern">([0-9]*[1-9][0-9]*(\.[0-9]*)?|0+\.[0-9]*[1-9][0-
9]*|\.[0-9]*[1-9][0-9]*)((cm)|(mm)|(in)|(pt)|(pc)|(px))</param>
    </data>
</define>
<define name="percent">
    <data type="string">
        <param name="pattern">-?([0-9]+(\.[0-9]*)?|\.[0-9]+)%</param>
    </data>
</define>
<define name="relativeLength">
    <data type="string">
        <param name="pattern">[0-9]+\*</param>
    </data>
</define>
<define name="coordinate">
    <ref name="length"/>
</define>
<define name="distance">
    <ref name="length"/>
</define>
<define name="color">
    <data type="string">
        <param name="pattern">#[0-9a-fA-F]{6}</param>
    </data>
</define>
<define name="styleName">
    <data type="NCName"/>
</define>
<define name="styleNameRef">
    <choice>
        <data type="NCName"/>
        <empty/>
    </choice>
</define>
<define name="styleNameRefs">
    <list>
        <zeroOrMore>
            <data type="NCName"/>
        </zeroOrMore>
    </list>
</define>
<define name="variableName">
    <data type="string"/>
```

```
</define>
<define name="formula">
    <!-- A formula should start with a namespace prefix, -->
    <!-- but has no restrictions-->
    <data type="string"/>
</define>

<define name="targetFrameName">
    <choice>
        <value>_self</value>
        <value>_blank</value>
        <value>_parent</value>
        <value>_top</value>
        <ref name="string"/>
    </choice>
</define>

<define name="valueType">
    <choice>
        <value>float</value>
        <value>time</value>
        <value>date</value>
        <value>percentage</value>
        <value>currency</value>
        <value>boolean</value>
        <value>string</value>
    </choice>
</define>

<define name="points">
    <data type="string">
        <param name="pattern">-?[0-9]+,-?[0-9]+([ ]+-?[0-9]+,-?[0-9]+)*</param>
    </data>
</define>
<define name="pathData">
    <data type="string"/>
</define>

<define name="vector3D">
    <data type="string">
        <param name="pattern">\([ ]*-?([0-9]+(\.[0-9]*)?|\.[0-9]+)([ ]+-?([0-
9]+(\.[0-9]*)?|\.[0-9]+)){2}[ ]*\)</param>
    </data>
</define>

<define name="namespacedToken">
    <data type="string">
        <param name="pattern">[0-9a-zA-Z_]+:[0-9a-zA-Z._\-]+</param>
    </data>
</define>
```

16.2 Other Definitions

To provide for extensibility of the format, inclusion of custom content is allowed on several occasions. The following definitions allow for inclusion of arbitrary attributes or elements (with arbitrary content models).

```
<define name="anyAttListOrElements">
    <zeroOrMore>
        <attribute>
            <anyName/>
            <text/>
        </attribute>
    </zeroOrMore>
    <ref name="anyElements"/>
</define>
<define name="anyElements">
    <zeroOrMore>
        <element>
            <anyName/>
            <mixed>
                <ref name="anyAttListOrElements"/>
            </mixed>
        </element>
    </zeroOrMore>
</define>
```

16.3 Relax-NG Schema Suffix

Suffix for the normative Relax-NG schema:

```
</grammar>
```

17 Packages

This chapter describes the package format that optionally can be used in OpenDocument. It contains the following sections:

- Introduction
- Zip File Structure
- Encryption
- Preview Image
- Manifest File

17.1 Introduction

As XML has no native support for binary objects such as images, [OLE] objects, or other media types, and because uncompressed XML files can get very large, OpenDocument uses a package file to store the XML content of a document together with its associated binary data, and to optionally compress the XML content. This package is a standard Zip file, whose structure is discussed below.

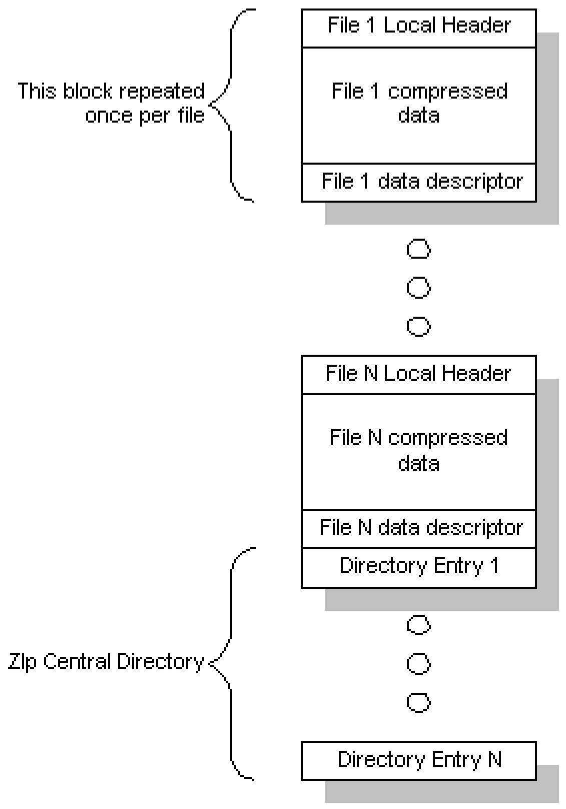
Information about the files contained in the package is stored in an XML file called the manifest file. The manifest file is always stored at the pathname META-INF/manifest.xml. The main pieces of information stored in the manifest are as follows:

- A list of all of the files in the package.
- The media type of each file in the package.
- If a file stored in the package is encrypted, the information required to decrypt the file is stored in the manifest.

17.2 Zip File Structure

A Zip file starts with a sequence of files, each of which can be compressed or stored in raw format. Each file has a local header immediately before its data, which contains most of the information about the file, including time-stamps, compression method and file name. The compressed file contents immediately follow, and are terminated by an optional data descriptor. The data descriptor contains the CRC and compressed size of the file, which are frequently not available when writing the local file header. If these details were included, the data descriptor can be skipped.

Each file in the archive is laid down sequentially in this format, followed by a central directory at the end of the Zip archive. The central directory is a contiguous set of directory entries, each of which contains all the information in the local file header, plus extras such as file comments and attributes. Most importantly, the central directory contains pointers to the position of each file in the archive, which makes navigation of the Zip file quick and easy.



For more details about the Zip file format, see [ZIP].

17.3 Encryption

The encryption process takes place in the following multiple stages:

1. A 20-byte SHA1 digest of the user entered password is created and passed to the package component.
2. The package component initializes a random number generator with the current time.
3. The random number generator is used to generate a random 8-byte initialization vector and 16-byte salt for each file.
4. This salt is used together with the 20-byte SHA1 digest of the password to derive a unique 128-bit key for each file. The algorithm used to derive the key is PBKDF2 using HMAC-SHA-1 (see [RFC2898]) with an iteration count of 1024.
5. The derived key is used together with the initialization vector to encrypt the file using the Blowfish algorithm in cipher-feedback (CFB) mode.

Each file that is encrypted is compressed before being encrypted. To allow the contents of the package file to be verified, it is necessary that encrypted files are flagged as 'STORED' rather than 'DEFLATED'. As entries which are 'STORED' must have their size equal to the compressed size, it is necessary to store the uncompressed size in the manifest. The compressed size is stored in both the local file header and central directory record of the Zip file.

17.4 MIME Type Stream

If a MIME type for a document that makes use of packages is existing, then the package **should** contain a stream called "mimetype". This stream **should** be first stream of the package's zip file, it **shall not** be compressed, and it **shall not** use an 'extra field' in its header (see [ZIP]).

The purpose is to allow packaged files to be identified through 'magic number' mechanisms, such as Unix's file/magic utility. If a ZIP file contains a stream at the beginning of the file that is uncompressed, and has no extra data in the header, then the stream name and the stream content can be found at fixed positions. More specifically, one will find:

- a string 'PK' at position 0 of all zip files
- a string 'mimetype' at position 30 of all such package files

- the mimetype itself at position 38 of such a package.

17.5 Usage of IRIs Within Packages

Within a file that is contained in a package, relative IRIs are used to reference other sub files of the package, but can also be used to reference files within the file system.

The following restrictions exist for IRIs that are used within a package:

- only sub files within the same package and files outside the package can be referenced.
- IRIs that reference a sub file of a package **shall** be relative, and they **shall not** contain paths that are not within the package. This especially means that sub files of a package **shall not** be referenced by an absolute IRI.
- sub file of a package can not be referenced from outside the package, for instance from the file system or another package.

A relative-path reference (as described in §6.5 of [RFC3987]) that occurs in a file that is contained in a package has to be resolved exactly as it would be resolved if the whole package gets unzipped into a directory at its current location. The base IRI for resolving relative-path references is the one that has to be used to retrieve the (unzipped) file that contains the relative-path reference.

All other kinds of IRI references, namely the ones that start with a protocol (like http:), an authority (i.e., //) or an absolute-path (i.e., /) do not need any special processing. This especially means that absolute-paths do not reference files inside the package, but within the hierarchy the package is contained in, for instance the file system. IRI references inside a package may leave the package, but once they have left the package, they never can return into the package or another one.

17.6 Preview Image

A thumbnail representation of a document should be generated by default when the file is saved. It should be a representation of the first page, first sheet, etc. of the document. For maximum reusability of the thumbnails they have to be generated without any effects, surrounding frames, or borders. Such effects might interfere with effects added to the thumbnails by the different file system explorers or may not be desired at all for certain use cases.

The thumbnail must be saved as “thumbnail.png” in a separate folder named “Thumbnails”.

The “Thumbnails” folder must not get a media type in the manifest.xml file, since it is not actually part of the document.

Encrypted files are intended to be unreadable for unauthorized users that's why a thumbnail for such files must not be generated. Instead of saving a thumbnail of the first page a replacement representation that doesn't depend on the contents of the document is saved for encrypted files which makes obvious that the corresponding file is encrypted.

In order to conform to the Thumbnail Managing Standard (TMS) at www.freedesktop.org, thumbnails must be saved as 24bit, non-interlaced PNG image with full alpha transparency. The required size for the thumbnails is 128x128 pixel.

17.7 Manifest File

The elements and attributes in the manifest file are in the namespace: urn:oasis:names:tc:opendocument:xmlns:manifest:1.0.

17.7.1 Relax-NG Schema

The normative XML Schema for OpenDocument Manifest files is embedded within this specification. It can be obtained from the specification document by concatenating all schema fragments contained in this chapters. All schema fragments have a gray background color and line numbers.

The schema language used within this specification is Relax-NG (see [RNG]).

Prefix for the normative Relax-NG Manifest schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    OASIS OpenDocument v1.1
    OASIS Standard, 1 Feb 2007
    Relax-NG Manifest Schema

    $Id$

    © 2002-2007 OASIS Open
    © 1999-2007 Sun Microsystems, Inc.
-->

<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"

    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"

    xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
```

17.7.2 Manifest Root Element

The root element is called manifest. The root element contains one fixed attribute which specifies the namespace as described above and multiple `<manifest:file-entry>` elements, each of which describes a single file in the package.

```
<define name="manifest">
    <element name="manifest:manifest">
        <oneOrMore>
            <ref name="file-entry"/>
        </oneOrMore>
    </element>
</define>

<start>
    <choice>
        <ref name="manifest"/>
    </choice>
</start>
```

17.7.3 File Entry

The `<manifest:file-entry>` element represents a single file within the package, and stores the files location in the package, the mime-type of the file and optionally the data required to decrypt this file.

Directories only receive `<manifest:file-entry>` entries if they have inherent semantics. For example, a directory that constitutes a sub-document referenced as an object from within the main document would contain a `<manifest:file-entry>` with a suitable media type. A directory for administrative or convenience purposes, such as a directory that contains various image files, would not receive an entry in the manifest file.

```
<define name="file-entry">
    <element name="manifest:file-entry">
        <ref name="file-entry-attlist"/>
        <optional>
            <ref name="encryption-data"/>
        </optional>
    </element>
</define>
```

The attributes associated with a `<manifest:file-entry>` are as follows:

- Full path
- Size
- Media type

Full Path

The `manifest:full-path` attribute describes the location of the file within the package.

```
<define name="file-entry-attlist" combine="interleave">
    <attribute name="manifest:full-path">
        <data type="string"/>
    </attribute>
</define>
```

Size

The `manifest:size` attribute is only present if the file is stored in an encrypted format. The reason why this attribute is required is explained in section 17.3. This attribute is only used for encrypted files.

```
<define name="file-entry-attlist" combine="interleave">
    <optional>
        <attribute name="manifest:size">
            <data type="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Media Type

The `manifest:media-type` attribute specifies the mime type of the specified file. For a full list of mime types see http://www.isi.edu/in-notes/iana/assignments/media-types/media-types. As an example, all XML streams have the media type "text/xml".

```
<define name="file-entry-attlist" combine="interleave">
    <attribute name="manifest:media-type">
        <data type="string"/>
    </attribute>
</define>
```

17.7.4 Encryption Data

The `<manifest:encryption-data>` element contains all of the information required to decrypt the file.

```
<define name="encryption-data">
    <element name="manifest:encryption-data">
        <ref name="encryption-data-attlist"/>
        <ref name="algorithm"/>
        <ref name="key-derivation"/>
    </element>
</define>
```

The `<encryption-data>` element contains the following elements:

- Algorithm
- Key Derivation

Checksum Type

The `manifest:checksum-type` attribute specifies the name of digest algorithm that can be used to check password correctness. Currently, the only supported digest algorithm is SHA1.

```
<define name="encryption-data-attlist" combine="interleave">
    <attribute name="manifest:checksum-type">
        <data type="string"/>
    </attribute>
</define>
```

Checksum

The `manifest:checksum` attribute specifies the digest in BASE64 encoding (as defined in [RFC2045]) that can be used to detect password correctness as specified within `manifest:checksum-type` attribute.

```
<define name="encryption-data-attlist" combine="interleave">
    <attribute name="manifest:checksum">
        <data type="base64Binary"/>
    </attribute>
</define>
```

17.7.5 Algorithm

The `<manifest:algorithm>` element contains information about the algorithm used to encrypt the data.

```
<define name="algorithm">
    <element name="manifest:algorithm">
        <ref name="algorithm-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes associated with `<manifest:algorithm>` are as follows:

- Algorithm name
- Initialization vector

Algorithm Name

The `manifest:algorithm-name` attribute specifies the name of the algorithm used to encrypt the file, and also specifies in which mode this algorithm was used. Currently, the only supports algorithm is the `Blowfish` algorithm in `CFB` mode.

```
<define name="algorithm-attlist" combine="interleave">
    <attribute name="manifest:algorithm-name">
        <data type="string"/>
    </attribute>
</define>
```

Initialization Vector

The `manifest:initialisation-vector` attribute specifies the 8 bytes used as an initialization vector to the stream cipher. The initialization vector is an 8 byte binary sequence, and so is encoded in BASE64 (as defined in [RFC2045]) when written to the manifest file.

```
<define name="algorithm-attlist" combine="interleave">
    <attribute name="manifest:initialisation-vector">
        <data type="base64Binary"/>
    </attribute>
</define>
```

17.7.6 Key Derivation

The `<manifest:key-derivation>` element contains the information that was used to derive the encryption key for this file from the user specified password.

```
<define name="key-derivation">
    <element name="manifest:key-derivation">
        <ref name="key-derivation-attlist"/>
        <empty/>
    </element>
</define>
```

The attributes associated with the `<manifest:key-derivation>` element are as follows:

- Key derivation name
- Salt
- Iteration count

Key Derivation Name

The `manifest:key-derivation-name` attribute specifies the name of the algorithm used to derive the name. At this time, the packages only supports the use of the PBKDF2 key derivation method. For further details see [RFC2898].

```
<define name="key-derivation-attlist" combine="interleave">
    <attribute name="manifest:key-derivation-name">
        <data type="string"/>
    </attribute>
</define>
```

Salt

The `manifest:salt` attribute specifies the 16-byte sequence used as the 'salt' by the key derivation algorithm. The salt is a 16-byte binary sequence, and thus is encoded in BASE64 (as defined in [RFC2045]) before being written to the manifest file.

```
<define name="key-derivation-attlist" combine="interleave">
    <attribute name="manifest:salt">
        <data type="base64Binary"/>
    </attribute>
</define>
```

Iteration Count

The `manifest:iteration-count` attribute specifies the number of iterations used by the key derivation algorithm to derive the key.

```
<define name="key-derivation-attlist" combine="interleave">
    <attribute name="manifest:iteration-count">
            <data type="nonNegativeInteger"/>
        </attribute>
</define>
```

Sample Manifest

```
<manifest:manifest
    xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
    <manifest:file-entry
        manifest:media-type="application/vnd.oasis.opendocument.text"
        manifest:full-path="/"/>
    <manifest:file-entry manifest:media-type="image/jpeg"
        manifest:full-path="Pictures/1000000000000032000000258912EB1C3.jpg"
          manifest:size="66704">
        <manifest:encryption-data>
            <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
                manifest:initialisation-vector="T+miu403484="/>
            <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
                manifest:iteration-count="1024"
                manifest:salt="aNYdmqv4cObAJSJjm4RzqA=="/>
        </manifest:encryption-data>
    </manifest:file-entry>
    <manifest:file-entry
        manifest:media-type="text/xml" manifest:full-path="content.xml"
        manifest:size="3143">
        <manifest:encryption-data>
            <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
                manifest:initialisation-vector="T+miu403484="/>
            <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
                manifest:iteration-count="1024"
                manifest:salt="aNYdmqv4cObAJSJjm4RzqA=="/>
        </manifest:encryption-data>
    </manifest:file-entry>
    <manifest:file-entry manifest:media-type="text/xml"
        manifest:full-path="styles.xml" manifest:size="5159">
        <manifest:encryption-data>
            <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
                manifest:initialisation-vector="bChL2No5I+A="/>
            <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
                manifest:iteration-count="1024"
                manifest:salt="/kfasyu7X0Ae+1uopdeCtA=="/>
        </manifest:encryption-data>
    </manifest:file-entry>
    <manifest:file-entry
        manifest:media-type="text/xml" manifest:full-path="meta.xml"/>
    <manifest:file-entry
        manifest:media-type="text/xml"
        manifest:full-path="settings.xml" manifest:size="5317">
        <manifest:encryption-data>
            <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
                manifest:initialisation-vector="JQxEm6rD+4c="/>
            <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
                manifest:iteration-count="1024"
                manifest:salt="PlpDaxloh4KUKx+v1g4V9g=="/>
        </manifest:encryption-data>
    </manifest:file-entry>
</manifest:manifest>
```

17.7.7 Relax-NG Schema Suffix

Suffix for the normative Relax-NG Manifest schema:

```
</grammar>
```

Appendix A.Strict Relax NG Schema

The Relax-NG (see [RNG])schema provided in this appendix equals the schema defined in chapters 1 to 16 of this specification, but restricts the content of meta information elements and formatting properties elements to the attributes and elements defined in this specification. See also section 1.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    OASIS OpenDocument v1.1
    OASIS Standard, 1 Feb 2007
    Strict Relax-NG Schema

    $Id$

    © 2002-2007 OASIS Open
    © 1999-2007 Sun Microsystems, Inc.
-->

<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="OpenDocument-schema-v1.1.rng">
        <define name="office-meta-content">
            <ref name="office-meta-content-strict"/>
        </define>
        <define name="style-page-layout-properties-content">
            <ref name="style-page-layout-properties-content-strict"/>
        </define>
        <define name="style-header-footer-properties-content">
            <ref name="style-header-footer-properties-content-strict"/>
        </define>
        <define name="style-drawing-page-properties-content">
            <ref name="style-drawing-page-properties-content-strict"/>
        </define>
        <define name="style-text-properties-content">
            <ref name="style-text-properties-content-strict"/>
        </define>
        <define name="style-paragraph-properties-content">
            <ref name="style-paragraph-properties-content-strict"/>
        </define>
        <define name="style-ruby-properties-content">
            <ref name="style-ruby-properties-content-strict"/>
        </define>
        <define name="style-section-properties-content">
            <ref name="style-section-properties-content-strict"/>
        </define>
        <define name="style-list-level-properties-content">
            <ref name="style-list-level-properties-content-strict"/>
        </define>
        <define name="style-table-properties-content">
            <ref name="style-table-properties-content-strict"/>
        </define>
        <define name="style-table-column-properties-content">
            <ref name="style-table-column-properties-content-strict"/>
        </define>
        <define name="style-table-row-properties-content">
            <ref name="style-table-row-properties-content-strict"/>
        </define>
        <define name="style-table-cell-properties-content">
            <ref name="style-table-cell-properties-content-strict"/>
        </define>
        <define name="style-graphic-properties-content">
            <ref name="style-graphic-properties-content-strict"/>
        </define>
        <define name="style-chart-properties-content">
            <ref name="style-chart-properties-content-strict"/>
        </define>
    </include>
</grammar>
```

Appendix B.References

[CSS2] Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs, *Cascading Style Sheets, level 2*, http://www.w3.org/TR/1998/REC-CSS2-19980512, W3C, 1998.

[CSS3Text] Michel Suignard, *CSS3 Text Module*, http://www.w3.org/TR/2003/CR-css3-text-20030514, W3C, 2003.

[DAISY] *ANSI/NISO Z39.86-2005 Specifications for the Digital Talking Book*, http://www.niso.org/standards/resources/Z39-86-2005.html, 2005

[DCMI] -, *Dublin Core Metadata Element Set, Version 1.1: Reference Description*, http://www.dublincore.org/documents/dces/, Dublin Core Metadata Initiative, 2003.

[DOM2] W3C, *Document Object Model Level 2 Core Specification*, http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113, W3C, 2000.

[DOMEvents2] Tom Pixley, *Document Object Model (DOM) Level 2 Events Specification*, http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113, W3C, 2000.

[DOMEvents3] Philippe Le Hégaret, Tom Pixley, *Document Object Model (DOM) Level 3 Events Specification*, http://www.w3.org/TR/DOM-Level-3-Events/, W3C, 2003.

[HTML4] Dave Raggett, Arnoud Le Hors, Ian Jacobs, *HTML 4.01 Specification*, http://www.w3.org/TR/1999/REC-html401-19991224, W3C, 1999.

[ISO/IEC Directives] ISO/IEC Directives, Part 2 *Rules for the structure and drafting of International Standards*, 2004

[JDBC] Jon Ellis, Linda Ho, Maydene Fisher, *JDBC 3.0 Specification*, http://java.sun.com/products/jdbc/, Sun Microsystems, Inc., 2001.

[MathML] David Carlisle, Patrick Ion, Robert Miner, Nico Poppelier, *Mathematical Markup Language (MathML) Version 2.0 (Second Edition)*, http://www.w3.org/TR/2003/REC-MathML2-20031021/, W3C, 2003.

[MIMETYPES] , *List of registered MIME types*, ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/, IANA, .

[OLE] Kraig Brockschmidt, Inside OLE, Microsoft Press, 1995, ISBN: 1-55615-843-2

[OOo] , *OpenOffice.org XML File Format 1.0 Technical Reference Manual*, http://xml.openoffice.org/xml_specification.pdf, Sun Microsystems, Inc., 2002.

[PNG] Thomas Boutell, *PNG (Portable Network Graphics) Specification*, http://www.w3.org/TR/REC-png-multi.html, W3C, 1996.

[RFC2045] N. Freed and N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, http://www.ietf.org/rfc/rfc2045.txt, IETF, 1996.

[RFC2048] N. Freed, J. Klensin, J. Postel, *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*, http://www.ietf.org/rfc/rfc2048.txt, IETF, 1996.

[RFC2616] IETF, *Hypertext Transfer Protocol -- HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt, IETF, 1999.

[RFC2898] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, http://www.ietf.org/rfc/rfc2898, IETF, 2000.

[RFC3066] H. Alvestrand, *Tags for the Identification of Languages*, http://www.ietf.org/rfc/rfc3066.txt, IETF, 2001.

[RFC3987] M. Duerst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/rfc/rfc3987.txt, IETF, 2005.

[RNG] ISO/IEC 19757-2 *Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG*, 2003

[RNG-Compat] James Clark, MURATA Makoto, *RELAX NG DTD Compatibility*, http://www.oasis-open.org/committees/relax-ng/compatibility-20011203.html, OASIS, 2001.

[SMIL20] W3C, *Synchronized Multimedia Integration Language 2.0 (SMIL 2.0)*, http://www.w3.org/TR/smil20/, W3C, 2001.

[SVG] Jon Ferraiolo, 藤沢 淳 (FUJISAWA Jun), Dean Jackson, *Scalable Vector Graphics (SVG) 1.1*, http://www.w3.org/TR/2003/REC-SVG11-20030114/, W3C, 2003.

[UAX9] Mark Davis, Unicode Standard Annex #9: *The Bidirectional Algorithm, Version 15 or later*, http://www.unicode.org/reports/tr9/tr9-15.html, 2005

[UNICODE] The Unicode Consortium. The Unicode Standard, Version 4.0.0, defined by: *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1)

[UTR20] Martin Dürst and Asmus Freytag, Unicode Technical Report #20: *Unicode in XML and other Markup Languages*, http://www.unicode.org/reports/tr20/, 2003

[XForms] W3C, *XForms*, http://www.w3.org/TR/xforms/, W3C, 2004.

[XLink] Steve DeRose, Eve Maler, David Orchard, *XML Linking Language*, http://www.w3c.org/TR/xlink/, W3C, 2001.

[xml-names] Tim Bray, Dave Hollander, Andrew Layman, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names/, W3C, 1999.

[XML1.0] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau , *Extensible Markup Language (XML) 1.0 (Third Edition)*, http://www.w3.org/TR/2004/REC-xml-20040204, W3C, 2004.

[xmlschema-2] Paul V. Biron, Ashok Malhotra, *XML Schema Part 2: Datatypes Second Edition*, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, W3C, 2004.

[XSL] W3C, *Extensible Stylesheet Language (XSL)*, http://www.w3.org/TR/2001/REC-xsl-20011015/, W3C, 2001.

[XSLT] James Clark, *XSL Transformations (XSLT) Version 1.0*, http://www.w3.org/TR/1999/REC-xslt-19991116, W3C, 1999.

[XSLT2] Michael Kay, *XSL Transformations (XSLT) Version 2.0*, http://www.w3.org/TR/2003/WD-xslt20-20031112/, W3C, 2003.

[ZIP] *Info-ZIP Application Note 970311*, ftp://ftp.uu.net/pub/archiving/zip/doc/appnote-970311-iz.zip, 1997

Appendix C.MIME Types and File Name Extensions (Non Normative)

The MIME types and extensions contained in this section are applicable only to office documents that are contained in a package (see section 2.1). See section 1.7 for the MIME type to use for documents that are not contained in a package.

The following table contains a list of MIME types and extensions for documents that conform to this specification, that, at the time this specification is published, have been registered according to [RFC2048]. Please see [MIMETYPES] for a current list of registered MIME types.

MIME type	***Ext.***	***Description***
No registered MIME types exist at the time this specification is published.		

The following table contains a list of MIME types and extensions for office documents that conform to this specification where a registration according to [RFC2048] is in progress at the time this specification is published.

Please check [MIMETYPES] before using these MIME types. If a MIME type is not listed there, the MIME type that is the result of inserting "`x-`" behind the "/" character (i.e., `application/x-vnd.oasis.opendocument.text`) should be used.

MIME type	***Ext.***	***Description***
`application/vnd.oasis.opendocument.text`	`odt`	Text document
`application/vnd.oasis.opendocument.text-template`	`ott`	Text document used as template
`application/vnd.oasis.opendocument.graphics`	`odg`	Graphics document (Drawing)
`application/vnd.oasis.opendocument.graphics-template`	`otg`	Drawing document used as template
`application/vnd.oasis.opendocument.presentation`	`odp`	Presentation document
`application/vnd.oasis.opendocument.presentation-template`	`otp`	Presentation document used as template
`application/vnd.oasis.opendocument.spreadsheet`	`ods`	Spreadsheet document
`application/vnd.oasis.opendocument.spreadsheet-template`	`ots`	Spreadsheet document used as template
`application/vnd.oasis.opendocument.chart`	`odc`	Chart document
`application/vnd.oasis.opendocument.chart-template`	`otc`	Chart document used as template
`application/vnd.oasis.opendocument.image`	`odi`	Image document
`application/vnd.oasis.opendocument.image-template`	`oti`	Image document used as template

MIME type	*Ext.*	*Description*
`application/vnd.oasis.opendocument.formula`	`odf`	Formula document
`application/ vnd.oasis.opendocument.formula-template`	`otf`	Formula document used as template
`application/vnd.oasis.opendocument.text-master`	`odm`	Global Text document (see section 2.3.1)
`application/vnd.oasis.opendocument.text-web`	`oth`	Text document used as template for HTML documents

Appendix D.Core Features Sets (Non Normative)

The OpenDocument specification does not specify which elements and attributes conforming application must, should, or may support. The intention behind this is to ensure that the OpenDocument specification can be used by as many implementations as possible, even if these applications do not support some or many of the elements and attributes defined in this specification. Viewer applications for instance may not support all editing relates elements and attributes (like change tracking), other application may support only the content related elements and attributes, but none of the style related ones.

Even typical office applications may only support a subset of the elements and attributes defined in this specification. They may for instance not support lists within text boxes or may not support some of the language related element and attributes.

The follow table provides an overview which element and attributes usually are supported by typical office application. It lists the chapters and sections contained in this specification and some typical office application classes. An "X" in this table indicates that most (or at least a significant number) of the elements and attributes defined in a section usually are supported by a certain application classes. An "(X)" indicates that only a limited number of elements and attributes usually is supported.

Sect-ion.	***Title***	***Text***	***Spread-sheet***	***Draw-ing***	***Presen-tation***	***Chart***	***Image***
2.2	Document Metadata	X	X	X	X	X	X
2.3	Body Element and Document Types	X	X	X	X	X	X
2.4	Application Settings	X	X	X	X	X	X
2.5	Scripts	X	X	X	X	X	X
2.6	Font Face Declarations	X	X	X	X	X	
2.7	Styles	X	X	X	X	X	X
2.8	Page Styles and Layout	X	X	X	X		
3	Metadata Elements	X	X	X	X	X	X
4.1	Paragraphs and Basic Text Structure	X	X(1)	X(2)	X(2)	X(3)	
4.1	Headings	X					
4.2	Page Sequences	X					
4.3	Lists	X		X(2)	X(2)		
4.4	Text Sections	X					
4.5	Page-bound graphical content	X					
4.6	Text Change Tracking	X					

Sect-ion.	***Title***	***Text***	***Spread-sheet***	***Draw-ing***	***Presen-tation***	***Chart***	***Image***
4.7	Text Declarations	X	(X)	(X)	(X)	(X)	
5.1	Basic Text Content	X	X(1)	X(2)	X(2)	X(3)	
5.2	Bookmarks and References	X					
5.3	Notes	X					
5.4	Ruby	X					
5.5	Text Annotation	X					
5.6	Index Marks	X					
5.7	Change Tracking and Change Marks	X					
5.8	Inline graphics and text-boxes	X					
6	Text Fields	X	(X)	(X)	(X)		
7	Text Indices	X					
8.1	Basic Table Model	X	X				
8.2	Advanced Table Model	X	X				
8.3	Advanced Tables		X				
8.4	Advanced Table Cells		X				
8.5	Spreadsheet Document Content		X				
8.6	Database Ranges		X				
8.7	Filters		X				
8.8	Data Pilot Tables		X				
8.9	Consolidation		X				
8.10	Table DDE Links		X				
8.11	Change Tracking in Spreadsheets		X				
9.1	Enhanced Page Features for Graphical Applications			X	X		
9.2	Drawing Shapes	X	X	X	X		
9.3	Frames	X	X	X	X		X(4)
9.4	3D Shapes	X	X	X	X		

Sect-ion.	*Title*	*Text*	*Spread-sheet*	*Draw-ing*	*Presen-tation*	*Chart*	*Image*
9.5	Custom Shapes	X	X	X	X		
9.6	Presentation Shapes				X		
9.7	Presentation Animations				X		
9.8	SMIL Presentation Animations				X		
9.9	Presentation Events				X		
9.10	Presentation Text Fields				X		
9.11	Presentation Document Content				X		
10	Chart Content					X	
11	Form Content	X	X	X	X		
12.1	Annotation	X(5)	X(1)				
12.2	Number Format for page numbers, etc.	X	X	X	X		
12.3	Change Tracking Metadata	X	X				
12.4	Event Listener Tables	X	X	X	X		
12.5	Mathematical Content	X	X	X	X		
12.6	DDE Connections	X	X				
13	SMIL Animations				X		
14.1	Style Element	X	X	X	X	X	X
14.2	Default Styles	X	X	X	X	X	X
14.3	Page Layout	X	X	X	X		
14.4	Master Pages	X	X	X	X		
14.5	Table Templates	X	X				
14.6	Font Face Declaration	X	X	X	X	X	
14.7	Data Styles	X	X	X	X	X	
14.8	Text Styles	X	X(6)	X(6)	X(6)	X(6)	
14.9	Enhanced Text Styles	X					
14.10	List Style	X		X	X		
14.11	Outline Style	X					

Sect-ion.	***Title***	***Text***	***Spread-sheet***	***Draw-ing***	***Presen-tation***	***Chart***	***Image***
14.12	Table Styles	X	X				
14.13	Graphic Styles	X	X	X	X		
14.14	Enhanced Graphic Style Elements	X	X	X	X	X	
14.15	Presentation Page Layouts				X		
14.16	Chart Styles					X	
15.2	Page Layout Formatting Properties	X	X	X	X		
15.3	Header Footer Formatting Properties	X	(X)				
15.4	Text Formatting Properties	X	X	X	X	X	
15.5	Paragraph Formatting Properties	X	X	X	X	X	
15.6	Ruby Text Formatting Properties	X					
15.7	Section Formatting Properties	X					
15.8	Table Formatting Properties	(X)	X				
15.9	Column Formatting Properties	(X)	X				
15.10	Table Row Formatting Properties	(X)	X				
15.11	Table Cell Formatting Properties	(X)	X				
15.12	List-Level Style Properties	X		X	X		
15.13	Stroke Properties	X(7)	X(7)	X	X	X	
15.14	Fill Properties	X(7)	X(7)	X	X	X	
15.15	Text Animation Properties	X(7)	X(7)	X	X		
15.16	Text Alignment Properties	X(7)	X(7)	X	X		
15.17	Color Properties	X(7)	X(7)	X	X		X
15.18	Shadow Properties	X(7)	X(7)	X	X		
15.19	Connector Properties	X(7)	X(7)	X	X		
15.20	Measure Properties	X(7)	X(7)	X	X		

Sect-ion.	***Title***	***Text***	***Spread-sheet***	***Draw-ing***	***Presen-tation***	***Chart***	***Image***
15.21	Caption Properties	X(7)	X(7)	X	X		
15.22	3D Geometry Properties	X(7)	X(7)	X	X	X	
15.23	3D Lighting Properties	X(7)	X(7)	X	X	X	
15.24	3D Texture Properties	X(7)	X(7)	X	X	X	
15.25	3D Material Properties	X(7)	X(7)	X	X	X	
15.26	3D Shadow Properties	X(7)	X(7)	X	X	X	
15.27	Frame Formatting Properties	X	(X)	(X)	(X)	(X)	
15.28	Floating Frame Formatting Properties	X	X	X	X		
15.29	Chart Formatting Properties					X	
15.30	Chart Subtype Properties					X	
15.31	Chart Axes Properties					X	
15.32	Common Chart Properties					X	
15.33	Statistical Properties					X	
15.34	Plot Area Properties					X	
15.35	Regression Curve Properties					X	
15.36	Presentation Page Attributes				X		

(1) within table cells

(2) within text boxes

(3) within some chart objects

(4) only frames that contain images

(5) within text

(6) only automatic styles

(7) only for drawing shapes

Appendix E.Accessibility Guidelines (Non Normative)

E.1. Title, Description and Caption of Graphical Elements

User agents supporting platform accessibility APIs should follow the following conventions for supporting the accessible name, accessible description (accessible help on some systems), and caption-id relationships (see sections 9.2.20 and 9.2.15:Caption-ID for a description of these elements and attributes):

If an `<svg:title>` element is provided it should map to the accessible name. If not, the name should use the text referenced by the `draw:caption-id` attribute. The `<svg:desc>` element must be used to support the accessible description. User agents shall not manufacture names for the `<svg:title>` element, such as using the drawing object name followed by a cardinal number in a string as it is used for accessibility. Name assignments such as these provide no semantic meaning to the user.

When transforming from another document format to OpenDocument the short names, like HTML's `alt` text on the `<img>` elements shall be mapped to the `<svg:title>` element.

If the user agent supports a platform which provides a `draw:caption-id` relationship in its accessibility API, this relationship for captions should be used to fulfill the relationship.

Guidance for authors:

Authors should not assign names to objects having no semantic value. If no name is assigned the caption text will be used in its place.`<svg:title>` elements shall take precedence over the caption text for accessible name assignment by the user agent.

Assignment of the long description should only be necessary when a drawing object is significantly complex and the user needs more information to describe it. Long descriptions would be more applicable to drawing groupings than basic drawing shapes.

Authoring tool responsibility for presenting and prompting for the `<svg:title>` and `<svg:desc>` elements:

Authoring tools should provide an option from an objects context menu to allow the user to enter the text for either of these elements as a minimum. More proactive authoring tools should have a facility for prompting the author for this text. Since the `<svg:desc>` element is a long description, a text area vs. a text field should be used to prompt the user accordingly in GUI-based authoring tools like office applications.

Navigation tools used to list the objects in the view should provide the type of object followed by the contents of `<svg:title>` element. The title must have been entered by the author.

For `<draw:g>` elements the drawing objects which are members of the group should visible only when the group is expanded.

E.2. Hyperlink Titles

When transforming from another document format to OpenDocument the alt text of hyperlinks, shall be mapped to the `office:title` attribute of `<text:a>` elements (see section 5.1.4) or `<draw:a>` elements (see section 9.3.9). When exporting OpenDocument documents to HTML, the contents of title text should be mapped to title attribute text on HTML anchor tags. As a minimum, authoring tools should provide a mechanism to provide the hint text.

The title text should be made accessible to the assistive technology and user. The user agent should allow for programmatic access through standard accessibility APIs such as the accessible description. Users should experience visible access to the hint text via the keyboard or mouse.

E.3. Tables in Presentations

Users importing non-OpenDocument slides that contain tables need access to the table structure via their assistive technology. Therefore tables imported into an OpenDocument application from another file format must have their structure preserved, and when saved as OpenDocument should be saved as as embedded spreadsheets.

E.4. Further Guidelines

Please see the additional, detailed Accessibility Guidelines http://docs.oasis-open.org/office/office-accessibility/guidelines. That more comprehensive document will be the up-to-date set of recommendations for what all OpenDocument applications should do in order to fully support accessibility.

Appendix F.Bidirectional (BiDi) Scripts,Numeric Digits Presentation and Calendars (Non Normative)

This appendix describes how bidirectional (BiDi) scripts and related information are represented in OpenDocument.

Paragraph and Layout Direction

In OpenDocument, the direction of text runs inside a paragraph is calculated using the Unicode BiDi Algorithm (see [UAX9]). The paragraph direction, as required by the BiDi Algorithm (see BD5 of [UAX9]), and the display direction of layout objects like table or page columns (in the following called layout direction) is controlled by a writing mode attribute (`style:writing-mode`) that can be used within styles.

The writing mode attribute can be applied individually to paragraph styles, page styles, section styles, table styles and graphic styles. If present within a paragraph style, it controls the paragraph direction of those paragraphs, to which the style is applied. If present within a page style, section style, table style or graphic style, it controls the layout direction of those pages, text sections, tables and text-boxes to which the styles is applied.

Section 15.2.19 describes the `style:writing-mode` attribute for page styles. It may, among other values, take the the values `lr-tb` (left-to-right, top-to-bottom) and `rl-tb` (right-to-left, top-to-bottom). The writing-mode attribute of a page style specifies the layout direction of page columns (left-to-right or right-to-left) for pages that are formatted using the page style.

Section 15.5.36 describes the `style:writing-mode` attribute for paragraph styles. It specifies the paragraph direction as defined in BD5 of [UAX9] for all paragraphs that have the paragraph style assigned. For paragraphs that are contained in lists, it further specifies whether the list numbers and bullets are displayed on the left or on the right of the paragraph.

The writing mode attribute for paragraph styles takes the same values as the writing mode attribute for page styles, but may also take the value `page`. This value specifies that the paragraph direction is inherited from the layout direction of the closest layout object (section, table or text-box) in which the paragraph is contained, and which has a layout direction other than `page`. If the paragraph is not contained in any of these layout objects, the paragraph direction is inherited from the page on which the paragraph appears.

The paragraph direction determines the default bidirectional orientation of the text in that paragraph. The result of the BiDi Algorithm can be manually changed by inserting BiDi embedding control characters (U+202A ... U+202E) and implicit directional marks (U+200E ...U+200F) into the text (see [UTR20]).

OpenDocument further has a `style:automatic-writing-mode` attributes (described in section 15.5.37) that specifies that an application is allowed to recalculate the value of the paragraph's writing-mode attribute based on its content whenever the content changes.

Section 15.7.8 describes the `style:writing-mode` attribute for section styles. It may take the same values as the writing mode attribute for paragraph styles.

The writing-mode attribute of a section style specifies the layout direction of section columns (left-to-right or right-to-left) for text sections that have the section style assigned. If the attribute's value is `page`, then the layout direction is inherited from the layout direction of the closest layout object (section, table or text-box) in which the section is contained, and which has a layout direction other than `page`.

Section 15.8.13 describes the `style:writing-mode` attribute for table styles. It may take the same values as the writing mode attribute for paragraph styles.

The writing-mode attribute of a table style specifies the layout direction of table cells (left-to-right or right-to-left) for tables that have the table style assigned. If the attribute's value is `page`, then the layout direction is inherited from the layout direction of the closest layout object (section, table or text-box) in which the table is contained, and which has a layout direction other than `page`.

Section 14.13.1 describes the `style:writing-mode` attribute for graphic styles. It may take the same values as the writing mode attribute for paragraph styles.

The writing-mode attribute of a graphic style specifies the layout direction of columns (left-to-right or right-to-left) for text-boxes that have the graphic style assigned. If the attribute's value is `page`, then the layout direction for text-boxes that are anchored to a page is inherited from the layout direction of the page on which the text-box is displayed. For text-boxes that have a different anchor type, the layout direction is inherited from the paragraph direction of the paragraph that contains the text-box.

Numeric Digits Presentation and Calendars

All digits that have a Unicode code point can be included in an OpenDocument document.

Note: Some office application have a feature that allows the user to specify whether the ASCII digits U+0030 ... U+0039 should be displayed as Arabic digits or as Indic digits (U+0660 ... U+0669). Since this feature effects only what digits are displayed and does not influence the representation of digits in the document itself, OpenDocument only allows storing this setting as an application specific setting, not as document or style content.

For list numbers, that are calculated automatically, OpenDocument provides a generic mechanism to specify the applicable numbering formats (see section 12.2.2).

Note: The specification currently mentions only "1, 2, 3...", "I, II, III...", and "i, ii, iii" explicitly, but the schema also allows a generic string here.

OpenDocument further supports data styles, which describe how different types of data are displayed, for example, a number or a date. Data styles are described in section 14.7. The presentation of numeric digits can be controlled by the transliteration attributes described in section 14.7.10. The presentation of date information can be controlled by the number:calendar attribute specified in section 14.7.11.

Appendix G. Changes From Previous Specification Versions (Non Normative)

G.1. Changes from “Open Office Specification 1.0 Committee Draft 1”

The following are the changes since the “Open Office Specification 1.0 Committee Draft 1”:

- The name of this specification has been changed to “Open Document Format for Office Applications (OpenDocument) 1.0”.
- The namespace URIs (section 1.3) have been adapted to the new specification name.
- The MIME type recommendations have been moved into a non normative appendix (appendix C) and have been adapted to the new specification name.
- Various new definitions have been added. Among them are:
  - Custom Shapes (section 9.5)
  - SMIL Animations (section 9.8 and chapter 13)
  - Support for XForms (section 11.2)
- Various errors in the schema and descriptions have been corrected.
- Some descriptions have been rewritten for easier understanding.

G.2. Changes from “Open Document Format for Office Applications (OpenDocument) 1.0 Committee Draft 2”

The following are the changes since the “Open Document Format for Office Applications (OpenDocument) 1.0 Committee Draft 2”:

- Namespaces for compatible elements and attributes have been added as follows:
  - urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0 for attributes that are compatible with [XSL];
  - urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0 for elements and attributes that are compatible with [SVG];
  - urn:oasis:names:tc:opendocument:xmlns:smil-compatible:1.0 for attributes that are compatible with [SMIL20].
- The following namespaces are not imported any longer:
  - http://www.w3.org/1999/XSL/Format
  - http://www.w3.org/2000/svg
  - http://www.w3.org/2001/SMIL20/
- The attribute `xforms:submission` specified in section 11.3.11 has been renamed to `form:xforms-submission`.

- Information for Custom Shapes (section 9.5), SMIL Animations (sections 9.8 and chapter 13) and Presentation Text Fields (section 9.10) has been added to the core feature set table in appendix D; the chapter numbers in the table have been updated.

G.3. Changes from "Open Document Format for Office Applications (OpenDocument) v1.0"

The following are the changes between the "Open Document Format for Office Applications (OpenDocument) v1.0" specification and the "Open Document Format for Office Applications (OpenDocument) v1.0 (Second Edition)" specification.

- The usage of key words for "shall", "may", etc. conforms now to Annex H of the ISO directives.
- Various ambiguous references were replaced with explicit references to chapter and section numbers.
- Various spelling and grammatical errors were corrected.
- All occurrences of "unicode" and "UNICODE" were replaced with the bibliographic reference "[UNICODE]". A bibliographic entry for Unicode was added to appendix B.
- All occurrences of the term URI, with the exception of one in appendix E.1, were replaced with the term IRI, because the W3C Schema "anyURI" datatype that is used in the OpenDocument schema actually takes IRIs rather than URIs. References to [RFC2396]) were replaced with references to [RFC3987]. In appendix B, the bibliographic entry for RFC2396 was replaced with one for RFC3987.
- A reference to the RELAX NG DTD Compatibility specification was added to the second paragraph of section 1.4. A bibliography entry for the RELAX NG DTD Compatibility specification was added to appendix B.
- References to [RFC2045] were added to some usages of the term "BASE64", and occurrences of "base64" were corrected to "BASE64". A bibliography entry for RFC2045 has been added to appendix B.
- The description of the `draw:z-index` attribute in section 9.2.5 was corrected.
- The references to the W3C CSS3 Text Module were clarified. In appendix B, the URL "http://www.w3.org/TR/2003/CR-css3-text-20030514" was added to the bibliographic entry for CSS3Text.
- In appendix B, the bibliographic entry for [RNG] now references to the ISO Relax-NG specification document rather than the OASIS Relax-NG specification document.
- In appendix B, the bibliographic entry for ZIP was updated.
- The contributor list was moved from the title page into an appendix.

G.4. Changes from "Open Document Format for Office Applications (OpenDocument) v1.0 (Second Edition)"

The following are the changes between the "Open Document Format for Office Applications

The following are the changes between the "Open Document Format for Office Applications (OpenDocument) v1.0 (Second Edition)" specification and the "Open Document Format for Office Applications (OpenDocument) v1.1" specification.

- The accessibility support of OpenDocument was improved by the following changes:

  - Soft Page Breaks were added.
  - The usage of table header and row columns was clarified (sections 8.2.2 and 8.2.4).
  - A logical navigation order for presentation slides was added (section 9.1.4).
  - Alternative texts for graphical objects, image maps, drawing layers and hyperlinks were added (sections 9.2.20, 9.3.9 and 5.1.4).
  - An attribute to establish a relationship between graphical objects and captions was added (section 9.2.15).
  - An appendix E containing accessibility guidelines was added.
- The use of DDE and OLE was clarified (section 12.6).
- The measure units supported by attribute values of type "length" was clarified (chapter 16).
- The definition of the "positiveLength" data type was improved (chapter 16).
- The recommendations for event names to be used in event listener definitions was clarified (sections 11.6 and 12.4.1).
- A style:writing-mode attribute was added for graphic styles (section 15.27.32).
- An appendix F containing information on bidirectional (BiDi) scripts, numeric digits presentation and calendars was added.
- The following errors in the schema were corrected:
  - Section 8.5.2:Null Date: The attribute "table:date-value" was misspelled "table:date-value-type".
  - Section 9.2.19:Align: An "<optional>" element was missing.
  - Section 13.1: For the elements described in this section, the schema referenced "common-fill-timing-attlist" instead of "common-timing-attlist".
  - Section 13.4.1:Repeating Elements: The value "indefinite" was missing for the "smil:repeatCount" attribute, and an "<optional/><optional>" pair was missing between the attribute definitions of "smil:repeatCount" and "smil:repeatDur".
  - Section 13.4.3: The element content for "<anim:seq>" was missing.
  - Section 13.4.4: The define "anim-iterate-attlist" was misspelled "anin-iterate-attlist".
  - Section 13.4.4:The Target Element: Instead of referencing "common-anim-target-attlist", the schema defined a subset of the defined attributes itself.
  - Section 14.4.1: A reference to "text-decls" was missing in the definition of "header-footer-content".
  - Section 14.4.2: A reference to "office-forms" was missing in the definition of "presentation-notes".
  - Section 14.6.1: The name attribute was missing the svg: namespace prefix.
  - Section 14.9.1:Position : The attribute value "right" was misspelled "rigth".
  - Section 15.3.8: The definition was named "style-header-footer-attlist" instead of "style-header-footer-properties-attlist".
  - Section 15.4.18: The schema for the "style:font-charset-asian" and "style:font-charset-complex" attributes was missing.

  - Section 15.5.35: The value "baseline" was missing.
  - Section 15.24.2:Kind: The attribute value “intensity” was misspelled “intesity”.
  - Section 15.27.22:Dynamic Wrap Threshold: The attribute “draw:dynamic-wrap-threshold” was misspelled “draw:dynamic-wrap-treshold”.
  - Section 15.31.3: The attribute “chart:interval-minor-divisor” was misspelled “chart:interval-minor”.
  - Section 15.36.7: The attribute value type of "smil:fadeColor" was not "color".
  - Appendix A: The "style-chart-properties-content" define referenced "style-properties-content" instead of "style-chart-properties-content-strict".
- The referenced version of xmlschema part 2 has been updated to xmlschema part 2 second edition.
- The text and schema in section 8.3.1:Referencing Table Cells the text and schema was extended to allow for apostrophe characters in table names by escaping them through doubling in quoted names.
- In section 15.5.39, and “auto”-value has been added to the style:page-number attribute.
- In section 14.5.1:Row and Column Styles, the text:paragraph-style-name attribute was added.
- The presentation:show-end-of-presentation-slide attribute has been added to section 9.11.5: Presentation Settings.
- The example for addressing of sub-table cells has be clarified in section 8.2.6:Subtables.
- The descriptive text in section 4.6.4 was clarified and some examples were corrected.
- The example in section 4.3.2 was corrected.
- In section 17.5, the restrictions that exist for IRIs that are used within a packages were clarified.
- The descriptive texts in sections 8.1.2:Default Cell Style and 8.2.1:Default Cell Style were clarified.
- In the description of the example in section 15.5.35, "middle" was referred to as "center".
- The descriptive texts of sections 15.10 and 15.10.4 were corrected to refer to rows rather than columns.
- The white-space processing in section 5.1.1 was clarified.
- Various spelling errors were corrected.

Appendix H.Acknowledgments (Non Normative)

Current Contributors:

Daniel Brotsky, Adobe Systems
Jerome Dumonteil, Ars Aperta
Charles Schulz, Ars Aperta
Jerry Berrier, BayState Council of the Blind (BSCB)
Donglin Wang, Beijing Sursen International Information Technology Co., Ltd.
Rui Zhao, Changfeng Open Standards Platform Software Alliance
Stephen Noble, Design Science, Inc.
John Madden, Duke University
Chieko Asakawa, IBM
Nathaniel Borenstein, IBM
Pete Brunet, IBM
Yue Ma, IBM
Richard Schwerdtfeger, IBM
Robert Weir, IBM
Zhi Yu Yue, IBM
John Barstow, Individual
Patrick Durusau, Individual
Michael Paciello, Individual
Janina Sajka, Individual
David Clark, Institute for Community Inclusion
Waldo Bastian, Intel Corporation
James Mason, ISO/IEC JTC1/SC34
David Faure, KDE e.V
Jody Goldberg, Novell
David Pawson, Royal National Institute for the Blind
Michael Brauer, Sun Microsystems, Inc.
Peter Korn, Sun Microsystems, Inc.
Lars Oppermann, Sun Microsystems, Inc.
Eike Rathke, Sun Microsystems, Inc.
Svante Schubert, Sun Microsystems, Inc.
Frank Stecher, Sun Microsystems, Inc.
Malte Timmermann, Sun Microsystems, Inc.
Daniel Bricklin, The OpenDocument Foundation, Inc.
Daniel Carrera, The OpenDocument Foundation, Inc.
Bruce D'Arcus, The OpenDocument Foundation, Inc.
Gary Edwards, The OpenDocument Foundation, Inc.
Elmar Geese, The OpenDocument Foundation, Inc.
Sam Hiser, The OpenDocument Foundation, Inc.
Michael Kleinhenz, The OpenDocument Foundation, Inc.
Tomas Mecir, The OpenDocument Foundation, Inc.
Thomas Metcalf, The OpenDocument Foundation, Inc.
Stefan Nikolaus, The OpenDocument Foundation, Inc.
Florian Reuter, The OpenDocument Foundation, Inc.
Daniel Vogelheim, The OpenDocument Foundation, Inc.
David A. Wheeler, The OpenDocument Foundation, Inc.
Chris Nokleberg, Tonic Systems, Inc.

Previous Contributors:

Paul Grosso, Arbortext
Tom Magliery, Blast Radius
Doug Alberg, Boeing

Paul Langille, Corel
John Chelsom, CSW Informatics
Monica Martin, Drake Certivo
Jason Harrop, Individual
Uche Ogbuji, Individual
Lauren Wood, Individual
Simon Davis, National Archive of Australia
Mark Heller, New York State Office of the Attorney General
Phil Boutros, Stellent