Open Office Specification

Proposal 1, 7 Feb 2003

Document identifier:
p-Open Office-Specification-1

Location:
http://www.oasis-open.org/committees/office/

Editors:
Michael Brauer, Sun Microsystems <michael.brauer@sun.com>
Gary Edwards <garyedwards@yahoo.com>
Daniel Vogelheim, Sun Microsystems <daniel.vogelheim@sun.com>

Contributors:
Doug Alberg, Boeing <doug.alberg@boeing.com>
Phil Boutros, Stellent <pboutros@stellent.com>
John Chelsom, CSW Informatics <john.chelsom@csw.co.uk>
Simon Davis, National Archive of Australia <simond@naa.gov.au>
Patrick Durusau, Society of Biblical Literature <pdurusau@emory.edu>
David Faure, <faure@kde.org>
Paul Grosso, Arbortext <pgrosso@arbortext.com>
Jason Harrop, SpeedLegal <jharrop@speedlegal.com>
Mark Heller, New York State Office of the Attorney General
<Mark.Heller@oag.state.ny.us>
Paul Langille, Corel <paul.langille@corel.com>
Tom Magliery, Corel <Tom.Magliery@corel.com>
Monica Martin, Drake Certivo <mmartin@certivo.net>
Uche Ogbuji <uche.ogbuji@fourthought.com>
Lauren Wood <lauren@textuality.com>

Abstract:

[Supply your own summary of the technical purpose of the document.]

Status:

This document is a draft, and will be updated periodically on no particular schedule. Send comments to the editors.

Committee members should send comments on this specification to the office@lists.oasis-open.org list. Others should subscribe to and send comments to the office-comment@lists.oasis-open.org list. To subscribe, send an email message to office-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of Contents

1 Introduction

1.1 Namespaces

Table 1 lists the namespaces used in the Open Office format and their default prefixes. For more information about XML namespaces, please refer to the *Namespaces in XML* specification [xml-names].

Table 1: Open Office XML Namespaces

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

Prefix	***Description***	***Namespace***
xlink	The XLink namespace.	http://www.w3.org/1999/xlink
fo	The XSL formatting objects and properties namespace.	http://www.w3.org/1999/XSL/Format
svg	The SVG namespace.	http://www.w3.org/2000/svg
dialog	For elements and attributes that describe dialogs.	http://openoffice.org/2000/dialog

1.2 *Relax-NG Schema Prefix*

Prefix for the normative Relax-NG schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    OASIS Open Office format
    preliminary Relax-NG schema

    $Id: p-open-office-specification.xml,v 1.6 2003/05/06 16:01:30 dvo Exp $

    © 2002 OASIS Open
    © 1999-2002 Sun microsystems
-->

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

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

    xmlns:office="http://openoffice.org/2000/office"
    xmlns:text="http://openoffice.org/2000/text"
    xmlns:table="http://openoffice.org/2000/table"

    xmlns:number="http://openoffice.org/2000/datastyle"
    xmlns:form="http://openoffice.org/2000/form"
    xmlns:dr3d="http://openoffice.org/2000/dr3d"
    xmlns:draw="http://openoffice.org/2000/drawing"
    xmlns:meta="http://openoffice.org/2000/meta"
    xmlns:chart="http://openoffice.org/2000/chart"
    xmlns:style="http://openoffice.org/2000/style"
    xmlns:presentation="http://openoffice.org/2001/presentation"
    xmlns:config="http://openoffice.org/2001/config"
    xmlns:script="http://openoffice.org/2000/script"

    xmlns:math="http://www.w3.org/1998/Math/MathML"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:svg="http://www.w3.org/2000/svg"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
>
```

2 Document Structure

This chapter introduces the structure of the Open Office XML format. The chapter contains the following sections:

- Document Roots
- Document Meta Data
- Body Element and Document Types
- Settings
- Script
- Styles

Each structural component in an Open Office XML document is represented by an **element**, with associated **attributes**. The structure of XML documents applies to all Open Office 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. You can exchange document content that is common to all document types from one type of document to another.

2.1 Document Roots

A **document root element** is the primary element of an Open Office XML document. It contains the entire document. All types of Open Office documents, for example, text documents, spreadsheets, and drawing documents use the same types of document root elements.

An Open Office document can be represented in the following two ways:

- As a single XML document.
- As a collection of several subdocuments, 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	***XML Document***
`<office:document>`	Complete office document in a single XML document.
`<office:document-content>`	Document content and automatic styles used in the content.
`<office:document-styles>`	Styles used in the document content and automatic styles used in the styles themselves.

Root Element	*XML Document*
`<office:document-meta>`	Document meta information, such as the author or the time of the last save action.
`<office:document-settings>`	Application-specific settings, such as the window size or printer information.

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.

> **Note:** Applications like OpenOffice.org typically store subdocuments in a single storage file. The OpenOffice.org application uses storages based on the popular ZIP archive format, which contain subdocuments `content.xml`, `styles.xml`, `meta.xml` and `settings.xml` for the four supported subdocument types.

```
<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-meta"/>
        <ref name="office-settings"/>
        <ref name="office-script"/>
        <ref name="office-font-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-attrs"/>
        <ref name="office-script"/>
        <ref name="office-font-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-attrs"/>
        <ref name="office-font-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-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-attrs"/>
        <ref name="office-settings"/>
    </element>
</define>
```

2.1.2 Document Root Attributes

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's **OPTIONAL** to validate the document, but the document **MUST** be well formed.

```
<define name="office-document-attrs">
    <optional>
        <attribute name="office:version">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

2.2 Document Meta Data

Metadata is general information about a document. In an Open Office XML document, all of the metadata elements are contained in an `<office:meta>` element, usually located at start of the document. Meta data 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">
            <zeroOrMore>
                <ref name="office-meta-data"/>
            </zeroOrMore>
        </element>
    </optional>
</define>
```

2.2.1 Pre-Defined vs. Custom Metadata

In Open Office XML the metadata is comprised of pre-defined meta data elements, user defined meta data, as well as custom meta data elements. The pre-defined meta data elements have defined semantics. They **SHOULD** be processed and updated by exiting 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 in an Open Office XML document

```
<office:meta>
  <dc:title>Title of the document</dc:title>
  <dc:description>Description/Comment for the document</dc:description>
  <meta:initial-creator>User Name</meta:initial-creator>
  <meta:creation-date>1999-10-18T12:34:56</meta:creation-date>
  <dc:creator>User Name</dc:creator>
  <dc:date>1999-10-19T15:16:17</dc:date>
  <meta:printed-by>User Name</meta:printed-by>
  <meta:print-date>1999-10-20T16:17:18</meta:print-date>
  <dc:subject>Description of the document</dc:subject>
  <meta: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">Value 1</meta:user-defined>
  <meta:user-defined meta:name="Field 2">Value 2</meta:user-defined>
</office:meta>
```

TODO: update user-defined fields to new specification

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 which numbers. The document content is typically framed by a prelude and epilogue, which contains content elements which may only occur once inside a document of the specific type.

```
<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 are contained in a container element 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-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.

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

```
<define name="office-text-content-main">
    <choice>
```

```
        <ref name="text-h"/>
        <ref name="text-p"/>
        <ref name="text-list"/>
        <ref name="table-table"/>
        <ref name="draw-a"/>
        <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="shape"/>
        <ref name="change-marks"/>
    </choice>
</define>
```

Text documents do not use epilogue elements.

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

2.3.2 Text Declarations

```
<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>
<!-- TODO: What about alphabetical index mark auto file??? -->
    <optional>
        <element name="text:alphabetical-index-auto-mark-file">
            <ref name="anything"/>
        </element>
    </optional>
</define>
```

2.3.3 Drawing Documents

TODO: define drawing document content

```
<define name="office-body-content" combine="choice">
    <element name="office:drawing">
        <ref name="anything"/>
    </element>
</define>
```

2.3.4 Presentation Documents

TODO: define presentation document content

```
<define name="office-body-content" combine="choice">
    <element name="office:presentation">
        <ref name="anything"/>
    </element>
</define>
```

2.3.5 Spreadsheet Documents

TODO: define spreadsheet document content

```
<define name="office-body-content" combine="choice">
    <element name="office:spreadsheet">
        <optional>
            <attribute name="table:structure-protected">
                <ref name="boolean"/>
            </attribute>
        </optional>
        <ref name="anyContent"/>
    </element>
</define>
```

2.3.6 Chart Documents

TODO: define chart document content

```
<define name="office-body-content" combine="choice">
    <element name="office:chart">
        <ref name="anything"/>
    </element>
</define>
```

2.3.7 Image Documents

TODO: define image document content

TODO: Did we ever agree on office:image? I can't remember.

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

2.4 Settings

TODO: define settings element

```
<define name="office-settings">
    <optional>
        <element name="office:settings">
            <ref name="anything"/>
        </element>
    </optional>
</define>
```

2.5 Script

TODO: define settings element

```
<define name="office-script">
    <optional>
        <element name="office:script">
            <ref name="anything"/>
        </element>
    </optional>
</define>
<define name="office-font-decls">
    <optional>
        <element name="office:font-decls">
            <ref name="anything"/>
        </element>
    </optional>
</define>
```

2.6 Styles

TODO: define styles elements

```
<define name="office-styles">
    <optional>
        <element name="office:styles">
            <ref name="anything"/>
        </element>
    </optional>
</define>
<define name="office-automatic-styles">
    <optional>
        <element name="office:automatic-styles">
            <ref name="anything"/>
        </element>
    </optional>
</define>
<define name="office-master-styles">
    <optional>
        <element name="office:master-styles">
            <ref name="anything"/>
        </element>
    </optional>
</define>
```

3 Common Content

3.1 Meta Data Elements

The meta data elements borrow heavily from the

3.1.1 Pre-Defined Meta Data Elements

There is a set of pre-defined meta data elements which **SHOULD** be processed and updated by the applications. Meta data elements **MAY** be omitted or occur multiple times. It is application-specific how to update multiple instances of the same elements.

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.

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 **MUST NOT** export the original identifier belonging to the application that created the document.

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

Title

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

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

Description

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

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

Subject

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

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

```
        <data type="string"/>
    </element>
</define>
```

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">
        <data type="string"/>
    </element>
</define>
```

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">
        <data type="string"/>
    </element>
</define>
```

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.

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

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">
        <data type="string"/>
    </element>
</define>
```

Creation Date and Time

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

To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See [ISO8601].

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

Modification Date and Time

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

To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See [ISO8601].

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

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

Print Date and Time

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

To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See .

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

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 you can associate 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 ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See [ISO8601].

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

```
        <attribute name="xlink:href">
            <ref name="uriReference"/>
        </attribute>
        <optional>
            <attribute name="xlink:type">
                <value>simple</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:actuate">
                <value>onRequest</value>
            </attribute>
        </optional>
        <optional>
            <attribute name="xlink:title">
                <data type="string"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:date">
                <ref name="dateTime"/>
            </attribute>
        </optional>
    </element>
</define>
```

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 you can associate 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 ISO 8601, the format of the value of this attribute is `PnYnMnDTnHnMnS`. See Section 5.5.3.2 of ISO 8601 for more detailed information on this time format. See [ISO8601].

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

Hyperlink Behavior

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

The attribute that you can associate 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-behavour>` element. See page for a pointer to the XLink Specification. If the value of the `meta:target-frame-name` attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the `meta:target-frame-name` attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.

```
<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>
```

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 RFC 1766, located at http://info.internet.isi.edu/in-notes/rfc/files/rfc1666.txt. It consists of a two-letter Language Code taken from the ISO 639 standard optionally followed by a hyphen (-) and a two-letter Country Code taken from the ISO 3166 standard. See page for pointers to ISO 639 and ISO 3166.

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

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">
        <data type="nonNegativeInteger"/>
    </element>
</define>
```

Editing Duration

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

The duration is represented in the manner described in Section 5.5.3.2 of ISO 8601. See [ISO8601].

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

Document Statistics

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

Document Type	***Document Statistics Attributes***
Text	`meta:page-count` `meta:table-count` `meta:draw-count` `meta:ole-object-count` `meta:paragraph-count` `meta:word-count` `meta:character-count` `meta:row-count`
Spreadsheet	`meta:page-count` `meta:table-count` `meta:cell-count` `meta:object-count`
Graphic	`meta:page-count` `meta:object-count`

```
<define name="office-meta-data" combine="choice">
    <element name="meta:document-statistic">
        <optional>
            <attribute name="meta:page-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:table-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:draw-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:image-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:ole-object-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:paragraph-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:word-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:character-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
```

```
            <attribute name="meta:row-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:cell-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
        <optional>
            <attribute name="meta:object-count">
                <data type="nonNegativeInteger"/>
            </attribute>
        </optional>
    </element>
</define>
```

3.1.2 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:

TODO: Explain type field, and adapt meta info example below.

- A `meta:name` attribute, which identifies the name of the metadata element.
- The value of the element, which is the metadata.

```
<define name="office-meta-data" combine="choice">
    <element name="meta:user-defined">
        <attribute name="meta:name">
            <data type="string"/>
        </attribute>
        <optional>
            <attribute name="meta:type">
                <data type="string"/>
            </attribute>
        </optional>
        <text/>
    </element>
</define>
```

3.1.3 Custom Metadata

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

4 Text Content

4.1 Headings, Paragraphs and Basic Text Structure

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

The XML elements that represent both headings and paragraphs are called **paragraph elements**. All text content in an Open Office XML file **MUST** be contained in either of these elements.

4.1.1 Headings

Headings define the chapter structure for an Open Office XML document. A (sub-)chapter begins with a heading and extends to the next heading at the same or higher level.

Heading Level

The `text: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="text-h">
    <element name="text:h">
        <attribute name="text:level">
            <data type="positiveInteger"/>
        </attribute>
        <ref name="paragraph-attrs"/>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </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 can be associated with style name, class name or conditional style name attributes. These attributes **MUST** reference a paragraph style or class.

If a conditional style is applied to a paragraph, the `text:style-name` attribute contains the name of the style that is applied under that condition. The `text:style-name` attribute does not contain the name of the conditional style that contains the conditions and maps to other styles. The conditional style name is the value of the `text:cond-style-name` attribute.

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.

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

Example: Styles and conditional styles in Open Office XML

```
<text:p text:style-name="Heading 1">
"Heading 1" is not a conditional style.
</text:p>
<text:p text:style-name="Numbering 1" text:cond-style-name="Text body">
"Text body" is a conditional style. If it is contained in a numbered
paragraph, it maps to "Numbering 1". This is assumed in this example.
</text:p>
```

4.2 Lists

TODO: Describe list model

```
<define name="text-list">
    <element name="text:list">
        <ref name="anything"/>
    </element>
</define>
```

4.3 Text Sections

TODO: describe text sections

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

4.4 Page-bound graphical content

TODO: supply information; maybe link to graphical content?

4.5 Tracked Changes

```
<define name="text-tracked-changes">
    <optional>
        <element name="text:tracked-changes">
            <ref name="anything"/>
        </element>
    </optional>
</define>

<define name="change-marks">
    <choice>
        <element name="text:change"><ref name="anything"/></element>
        <element name="text:change-start"><ref name="anything"/></element>
        <element name="text:change-end"><ref name="anything"/></element>
    </choice>
</define>
```

5 Paragraph Elements Content

5.1 Basic Text Content

Paragraph element's children make up the text content of any **Open Office** 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, in other words they are processed in the same way that HTML processes them. The following Unicode characters are normalized to a SPACE character:

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

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

White-space processing takes place within the following elements:

- `<text:p>`
- `<text:h>`
- `<text:span>`

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

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

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. You do not get an error if the character preceding the element is not a white-space character, but it is good practice to use this element for the second and all following SPACE characters in a sequence. This way, an application recognizes a single space character without recognizing this element.

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

Tab Character

The `<text:tab>` element represents the Unicode tab character HORIZONTAL TABULATION (0x0009) in a heading or paragraph.

TODO: describe and clarify tab-ref attribute

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

Back-Tab Character

TODO: Back-Tab Character? Apparently no corresponding Unicode symbol exists.

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>
```

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-name` attributes, respectively, attached to the `<text:span>` element. These attributes **MUST** refer to text styles or classes.

You can nest `<text:span>` elements.

White-space characters contained in this element are collapsed.

```
<define name="paragraph-content" combine="choice">
    <element name="text:span">
        <ref name="class-attrs"/>
        <zeroOrMore>
            <ref name="paragraph-content"/>
        </zeroOrMore>
    </element>
</define>
```

Example: Text style in Open Office XML:

```
<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, `<script:events>`, which contains the events assigned to the hyperlink. See Chapter XXX for more information on the event table element.

TODO: set reference to script-events chapter

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

The attributes that you can associate 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 `text: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">
            <data type="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="uriReference"/>
    </attribute>
    <optional>
        <attribute name="xlink:type">
            <value>simple</value>
        </attribute>
    </optional>
    <optional>
        <attribute name="xlink:actuate">
            <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. See page for a pointer to the XLink Specification. If the value of the attribute is _blank, the xlink:show attribute value is new. If the value of the attribute is any of the other value options, the value of the xlink:show attribute is replace. See [XLink].

```
<define name="text-a-attlist" combine="interleave">
    <optional>
        <attribute name="office:target-frame-name">
            <data type="string"/>
        </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="styleName"/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:visited-style-name">
            <ref name="styleName"/>
        </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">
                <data type="string"/>
            </attribute>
        </element>
        <element name="text:bookmark-start">
            <attribute name="text:name">
                <data type="string"/>
            </attribute>
        </element>
        <element name="text:bookmark-end">
            <attribute name="text:name">
                <data type="string"/>
            </attribute>
        </element>
    </choice>
</define>
```

Example: Bookmarks in Open Office XML

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

5.2.2 References

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

- A point reference
  A point reference marks a particular position in text and is represented by a single `<text:reference-mark>` element.

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

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

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

TODO: Do references to OOo software belong here? Maybe they should be moved to an appendix, or just left out?

Point References

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

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

Range References

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

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

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

- Cannot support overlapping references

- Do not interact well with other elements

Take the following example:

Example: Overlapping range references

```
<text:p>
  <text:reference-mark-start 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:note-configuration>` element, which occurs inside the `<style: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.

Open Office XML represents notes 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. Open Office XML uses the same structure but introduces a `text:note-citation` element. The second child contains the note body, just as in XSL.

Additionally, Open Office 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">
                <data type="string"/>
            </attribute>
        </optional>
        <element name="text:note-citation">
            <optional>
                <attribute name="text:label">
                    <data type="string"/>
                </attribute>
            </optional>
            <text/>
        </element>
<!-- Validity constraint: text:footnote and text:endnote elements may not
    contain other text:footnote or text:endnote elements, even though the DTD
    allows this (via the %text; in the foot-/endnote-body).
    Unfortunately, this constraint cannot be easily specified in the DTD.
```

```
-->
        <element name="text:note-body">
            <zeroOrMore>
                <ref name="office-text-content-main"/>
            </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.

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>
```

5.3.2 Notes Configuration Element

An Open Office document 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 you can associate 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

You can include the following element 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="styleName"/>
        </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="styleName"/>
        </attribute>
    </optional>
</define>
```

Default Footnote Paragraph Style

The default footnote paragraph style is only used for footnotes that are inserted into an existing document. It is not used for footnotes that already exist.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:default-style-name">
            <ref name="styleName"/>
        </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="styleName"/>
        </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">
            <data type="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

Number Format

See Chapter 2 for information on the number format for footnotes.

TODO: update reference

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="style:num-prefix">
            <data type="string"/>
        </attribute>
    </optional>
</define>
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="style:num-suffix">
            <data type="string"/>
        </attribute>
    </optional>
</define>
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <ref name="field-num-format"/>
    </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:

- The bottom of the page where the footnote citation is located.
- The end of the document.

> **Note:** XSL does not have the capability to display footnotes at the end of the document. However, you can use an XSLT stylesheet to generate some other flow objects to display such footnotes.

```
<define name="text-notes-configuration-content" combine="interleave">
    <optional>
        <attribute name="text:footnotes-position">
            <choice>
```

```
                <value>document</value>
                <value>page</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 in Open Office XML

```
<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>
```

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 `<text:ruby-text >` element contains the annotation text. It may contain only plain text.

The attributes that you can associate with the `<text:ruby>` element are:

- Ruby style

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

```
        <optional>
            <attribute name="text:style-name">
                <ref name="styleName"/>
            </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="styleName"/>
                </attribute>
            </optional>
            <text/>
        </element>
    </element>
</define>
```

TODO: replace style-name with style-name + class-name attributes?

5.4.1 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.

5.4.2 Ruby Text Formatting Properties

All of the ruby text is displayed using the same formatting properties. The `style:style-name` attribute is used to specify these properties.

Ruby Position

This property specifies the position of the ruby text relative to the ruby base.

```
<define name="ruby-style" combine="interleave">
    <optional>
        <attribute name="style:ruby-position">
            <choice>
                <value>above</value>
                <value>below</value>
            </choice>
        </attribute>
    </optional>
</define>
```

Ruby Alignment

This property specifies the alignment of the ruby text relative to the ruby base.

```
<define name="ruby-style" 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>
```

5.5 Text Annotation

The Open Office XML format allows annotation to appear within a paragraph element. See section 15.1 for details on annotations.

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

5.6 Index Marks

TODO: index marks with Indices in Chapter 7 ???

```
<define name="paragraph-content" combine="choice">
    <element name="text:bibliography-mark">
        <ref name="anything"/>
    </element>
</define>
<define name="paragraph-content" combine="choice">
    <element>
        <choice>
            <name>text:alphabetical-index-mark</name>
            <name>text:alphabetical-index-mark-start</name>
            <name>text:alphabetical-index-mark-end</name>
            <name>text:user-index-mark</name>
            <name>text:user-index-mark-start</name>
            <name>text:user-index-mark-end</name>
            <name>text:toc-mark</name>
            <name>text:toc-mark-start</name>
            <name>text:toc-mark-end</name>
        </choice>
        <ref name="anything"/>
    </element>
</define>
```

TODO: What about the auto-mark file ???

5.7 Change Tracking and Change Marks

TODO: change tracking in the text content chapter, and refer to change marks from here?

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

5.8 Inline graphics and text-boxes

TODO: change tracking in the text content chapter, and refer to change marks from here?

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

6 Text Fields

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

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

This section describes how fields are represented in the Open Office XML 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 Open Office XML, 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

Open Office 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. You can adjust the date to display a date other than the current date. For example, you can change the date on a document that was edited late at night so that it displays the date of the following day or several days later.

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

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

The attributes that you can associate with the `<text:date>` element are:

- Date value
- Date adjustment
- Fixed (see Section 3.7.43)
- Formatting style (see Section 3.7.43). The formatting style must be a date data style, see Section XXX for more information.

TODO: provide proper references

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

Date Value

The `text:date-value` attribute specifies a particular date value. For example, if the date field is marked fixed, you can use this attribute to specify the date on which the field was marked as fixed. You can also use this attribute to specify a future date. Some applications support date and time in addition to date-only values, in accordance with [ISO8601].

The date value should conform with the extended date format described in Section 5.2.1.1 of [ISO8601]. 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="dateTime"/>
        </attribute>
    </optional>
</define>
```

Date Adjustment

You can adjust the value of a date field by a certain time period, which you specify using the `text:date-adjust` attribute. OpenOffice.org Writer truncates the specified time period to a period of full days and adds it to the value of the date field. If the time period is negative, OpenOffice.org Writer subtracts it 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 Section 5.5.3.2 of [ISO8601]. 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">
            <data type="duration"/>
        </attribute>
    </optional>
</define>
```

Note: Currently, OpenOffice.org does not support month or year durations.

TODO: Do OOo implementation restrictions belong here?

6.2.2 Time Fields

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

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

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

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

The attributes that you can associate 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 2.5.5 for more information.

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

Time Value

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

The value of this attribute must conform with the extended time format described in Section 5.4.1 of [ISO8601]. 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="dateTime"/>
        </attribute>
    </optional>
</define>
```

Time Adjustment

You can adjust the value of a time field by a certain time period, which you specify using the `text:time-adjust` attribute.

> **Note:** The OpenOffice.org software truncates the time period to a period of full minutes and adds it to the value of the time field. If the time period is negative, the OpenOffice.org software subtracts it from the value of the time field yielding a time in the past.

TODO: Do OOo implementation restrictions belong here? Is the negative offset thing a format or an application thing?

The value of this attribute must conform to the time period format described in Section 5.5.3.2 of [ISO8601]. 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">
            <data type="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 for recurring content, such as headers and footers. If you insert a page number field into a footer, the current page number is displayed on every page on which the footer appears.

The attributes that you can associate 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="field-num-format"/>
        <ref name="field-fixed"/>
    </interleave>
</define>
```

Page Adjustment

You can adjust the value of a page number field by a specified number, allowing you to display the page numbers of following or preceding pages. You specify the adjustment number using the `text:page-adjust` attribute. When you use this attribute, the application:

1. Adds the value of the attribute to the current page number.
2. Checks to see if the resulting page exists.
3. If the page exists, the number of that page is displayed.
4. If the page does not exist, the value of the page number field remains empty and no number is displayed.

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

```
            <data type="integer"/>
        </attribute>
    </optional>
</define>
```

Display Previous or Following Page Numbers

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

```
<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, you should use a combination of the `text:select page` and `text:page adjust` attributes.

Example: Displaying the current page number on all pages except the first page

```
<text:page-number text:select-page="previous"
                  text:page-adjust="1"
                   text:num-format="1"/>
```

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">
            <data type="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 Open Office 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. chapter 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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <text/>
    </element>
</define>
```

6.2.6 Author Fields

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

You can fix the value of author fields 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="field-fixed"/>
        <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="field-fixed"/>
        <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 you can associate with the `<text:chapter>` element are:

- Display

- Outline level

Display

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

```
<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>
```

> **Note**: In the current version of the OpenOffice.org Writer user interface, `plain-number-and-name` chapter fields are not supported.

TODO: Do OOo issues belong here?

Example: If the current chapter number is 2.4, the chapter title is Working with Tables, the prefix is [, and suffix is], the possible display options and results are as follows:

Value of `text:display` attribute	***Field content displayed***
`number`	[2.4]
`name`	Working with Tables
`number-and-name`	[2.4] Working with Tables
`plain-number`	2.4
`plain-number-and-name`	2.4 Working with Tables

Outline Level

This attribute allows you to specify the outline level to use. The chapter field displays the chapter number or title up to the specified outline level.

```
<define name="text-chapter-attlist" combine="interleave">
    <attribute name="text:outline-level">
        <data type="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 you can associate 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. You can choose whether to display:

- The full file name including the path and the extension
- The file path only
- The file name only
- The file name and the extension

```
<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="field-fixed"/>
</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 attribute that you can associate 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. You can choose to display:

- The full file name including the path and the extension

- The file path only
- The file name only
- The file name and the extension
- The title
- The area of the document template

The latter two values are used in the OpenOffice.org Writer user interface document template dialog. The display 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>
```

6.3 Variable Fields

Open Office Writer documents can contain variables, which are processed or displayed using variable fields. A variable is a name/value pair. The variable name is used throughout the document to identify a particular variable, and therefore variable names cannot be reused for different types of variables. Most variable fields support different value types, such as numbers, dates, strings, and so on. In the Open Office XML file format, a variable must be declared at the beginning of a document.

There are three types of variables in Open Office XML:

- **Simple variables**

  Simple variables, usually called variables, can take different values at different positions throughout a document. You set simple variables 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. You can use simple variables 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 OpenOffice.org user interface, you can set a user variable at any occurrence of a user field, or using user variable input fields. In the Open Office XML file format, you can only set the value of the user variable after the variable is declared.

- **Sequence variables**

  Sequence variables are used to number certain items in an Open Office Writer document, for example, images or tables.

Expression and text input fields are also variable fields, but they are not associated with any particular variables. Since their functionality is closely related to that of the variable fields, they are also described in this section of the manual.

You must declare variables before you can use them. The variable declarations are collected in container elements for the particular variable type. The Open Office XML code for declaring variables is described in the following table.

6.3.1 Declaring Simple Variables

You declare simple variables 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, you attach the following attributes to the `<text:variable-decl>` element:

- `text:name`

  The name of the variable must be unique. The name cannot already be used for any other type of variable. See Section 6.7.3 for information on using this attribute.

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

6.3.2 Setting Simple Variables

You can set simple variables 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 you can attach to the `<text:variable-set>` element are:

- `text:name`

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 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, you can omit this attribute. See Section 6.7.6 for information on using this attribute.

- `text: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, the current OpenOffice.org software implementation allows the use of different value types for different instances of the same variable. In the case of the numeric value types `float`, `percentage`, and `currency`, the value is automatically converted to the different value type. For value types that are stored internally as numbers, such as `date`, `time`, and `boolean` types, the values are reinterpreted as numbers of the respective types. If a variable is used for both string and non-string types, the behavior is undefined, therefore this practice is not recommended.

- `text:display`

  You can use this attribute to specify whether or not to display the value of the `<text:variable-set>` element. If the `text:display` attribute is set to `value`, the value of the variable is displayed. If the attribute is set to `none`, the value is not displayed. See Section 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="field-name"/>
            <ref name="field-formula"/>
            <ref name="field-value-and-type"/>
            <ref name="field-display-value-none"/>
            <ref name="field-data-style-name"/>
        </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 you can attach to the `<text:variable-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:variable-decl>` element. See Section 6.7.3 for information on using this attribute.

- `text:display`

  You can use this attribute to specify whether to display the formula for a simple variable or the computed value of the variable. See Section 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="field-name"/>
            <ref name="field-display-value-formula"/>
            <ref name="field-data-style-name"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.4 Simple Variable Input Fields

As an alternative to setting simple variables using formulas in variable setter fields, the user can be prompted for variable values. To do this, you use the `<text:variable-input>` element. 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 you can attach to the `<text:variable-input>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. It must match the name of a variable that was already declared. See Section 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.

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

  See Section 6.7.1 for information on using these attributes.

- `text:display`

  You can use this attribute to specify whether to display or hide the value of the variable through the variable input field. See Section 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="field-name"/>
            <ref name="field-description"/>
            <ref name="field-value-type"/>
            <ref name="field-display-value-none"/>
            <ref name="field-data-style-name"/>
        </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 because the value of the user variable is specified in the variable declaration.

The attributes that you can associate with the `<text:user-field-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and sequence variables. See Section 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, you can omit this attribute. See Section 6.7.6 for information on using this attribute.

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

6.3.6 Displaying User Variables

You can display the content of user variables using `<text:user-field-get>` elements.

The attributes that you can attach to the `<text:user-field-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:user-field-decl>` element. See Section 6.7.3 for information on using this attribute.

- `text:display`

  You can use this attribute to specify whether to:

  - Display the formula used to compute the value of the user variable.
  - Display the value of the user variable.
  - Hide the user variable fields.

    See Section 6.7.5 for information on using this attribute.

  Note: Since the OpenOffice.org Writer user interface allows users to edit a user field variable by clicking on any user field, a hidden `<text:user-field-get>` element can be used as an anchor to allow easy access to a particular user field variable.

- `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="field-name"/>
            <ref name="field-display-value-formula-none"/>
            <ref name="field-data-style-name"/>
        </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. You can set a user variable in this way using the `<text:user-field-input>` element. Since the value of a user field variable is stored in the `<text:user-field-decl>` element, the <`text:user-field-input>` element does not contain the value and value type attributes from the <`text:variable-input>` field.

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

The attributes that you can attach to the `<text:user-field-input>` element are:

- `text:name`

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 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="field-name"/>
            <ref name="field-description"/>
            <ref name="field-data-style-name"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.3.8 Declaring Sequence Variables

Sequence variables are used to number items within an Open Office Writer document. Sequence variables are most commonly used for sequential numbering. However, you can include expression formulas in sequence fields to support more advanced sequences. See Section 3.7.37 for more information on Using Sequence Fields and their uses.

You declare sequence variables using the `<text:sequence-decl>` element.

To facilitate chapter-specific numbering, you can attach attributes for the chapter level and a separation character to a sequence variable. The attributes that you can attach to the `<text:sequence-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and user variables. See Section 3.7.43 for information on using this attribute.

- `text:display-outline-level`

  See the section *Outline Level* for information about this attribute.

- `text:separation-character`

  See the section *Separation Character* for information about this attribute.

```
<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="field-name"/>
</define>
```

Outline Level

You can number sequences by chapter. To use this feature, use the `text:display-outline-level` attribute to specify an outline level that determines which chapters to reference for the chapter-specific numbering. All chapters that are at or below the specified outline level reset the value of the sequence to zero, the default value. Also, the chapter number of the last chapter at or below the specified outline level is prepended to the sequence number. Choosing an outline level of zero results in a straight sequence of all sequence elements for that sequence variable.

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

Separation Character

If you number sequences by chapter, use this attribute 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, you **MAY** specify a separation character. The default separation character is `"."`.Otherwise, if the value of `text:display-outline-level` is zero, you must omit this attribute.

```
<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, you can use it in sequence fields throughout the document. Most sequence fields simply increment and display the sequence variable. However, sequence fields can also assume a new start value at any given position in a document. This start value is computed using a formula which is contained in the sequence field. If a sequence field without a start value is added to the OpenOffice.org user interface, the OpenOffice.org 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 you can attach to the `<text:sequence>` element are:

- `text:name`

  This attribute specifies the name of the variable that the field is to display. It must match the name of a sequence variable that was already declared. See Section 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 following section Reference Name for more information about this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:sequence">
        <interleave>
            <ref name="field-name"/>
            <ref name="field-formula"/>
            <ref name="field-num-format"/>
            <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 3.7.27 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.

TODO: insert proper reference

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">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

6.3.10Expression 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 you can attach to the `<text:expression>` element are:

- `text:formula`

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

- `text: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 3.7.43 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:expression">
        <interleave>
            <ref name="field-formula"/>
            <optional>
                <ref name="field-value-and-type"/>
            </optional>
            <ref name="field-display-value-formula"/>
            <ref name="field-data-style-name"/>
        </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 OpenOffice.org 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 attribute that you can attach to the <text:text-input> element is:

- text:description

  This attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the purpose of the field and how it is used within the document, to enable them to choose an appropriate value. See Section 3.7.43 for information on using this attribute.

```
<define name="paragraph-content" combine="choice">
    <element name="text:text-input">
        <ref name="field-description"/>
        <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 3.1.

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="field-fixed"/>
        <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="field-fixed"/>
            <ref name="field-data-style-name"/>
            <optional>
                <attribute name="text:date-value">
                    <data type="date"/>
                </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="field-fixed"/>
            <ref name="field-data-style-name"/>
            <optional>
                <attribute name="text:time-value">
                    <ref name="dateTime"/>
                </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="field-fixed"/>
        <text/>
    </element>
</define>
```

6.4.5 User-Defined Document Information

This group of elements contains user-defined information about the document. The fields are not used or interpreted by OpenOffice.org, so the user may use these elements for any purpose.

```
<define name="paragraph-content" combine="choice">
    <element name="text:user-defined">
        <interleave>
            <ref name="field-fixed"/>
            <attribute name="text:name">
                <data type="string"/>
            </attribute>
        </interleave>
        <text/>
    </element>
</define>
```

TODO: Update user-defined fields

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

6.4.10Document Subject

This element represents the subject of the document.

```
<define name="paragraph-content" combine="choice">
    <element name="text:subject">
        <ref name="field-fixed"/>
        <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="field-fixed"/>
        <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="field-fixed"/>
        <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, OpenOffice.org 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="field-fixed"/>
            <ref name="field-data-style-name"/>
            <optional>
                <attribute name="text:duration">
                    <data type="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 `<meta: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="field-fixed"/>
            <ref name="field-data-style-name"/>
            <optional>
                <attribute name="text:time-value">
                    <data type="time"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.15Document Modification Date

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

This element displays the information from the `<meta: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="field-fixed"/>
            <ref name="field-data-style-name"/>
            <optional>
                <attribute name="text:date-value">
                    <data type="date"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.4.16Document 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="field-fixed"/>
        <text/>
    </element>
</define>
```

6.4.17Document Statistics Fields

TODO. documentation got statistics fields is missing!

```
<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="field-num-format"/>
        <text/>
    </element>
</define>
```

6.5 Database Fields

Open Office XML documents can reference databases and display database information as text content. To display database information, Open Office XML uses a group of text fields, collectively called database fields. OpenOffice.org Base can use database tables from SQL servers, therefore you can use database fields to access any SQL database, provided that the appropriate drivers are available.

In OpenOffice.org Base, a database contains 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 XML text content, therefore they are not discussed in this chapter. From the point of view of embedding database information in OpenOffice.org text documents, queries and tables are considered the same. Therefore for the remainder of this section, the phrase *database table* refers to both database tables and database queries.

Every database in OpenOffice.org Base has a name and this name is used by all of the OpenOffice.org components to identify a database. All database fields contain a database name and most database fields also contain the name of a database table, which must be stored in the named database. An additional attribute determines whether the database table refers to an SQL table, an OpenOffice.org query, or the result of an SQL command.

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

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 OpenOffice.org user interface, you can add database rows in one of the following ways:

- Manually, using the Beamer and the Data to Fields function.
- Using the Form Letter 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. Using the `<text:database-select>` and `<text:database-next>` elements, you can determine which row within the current selection to display. You can display the current row number for a particular table 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 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 you can associate 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 the section *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 OpenOffice.org Base is used. See Section 3.7.43 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="field-database-table"/>
</define>
<define name="text-database-display-attlist" combine="interleave">
    <ref name="field-data-style-name"/>
</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">
        <data type="string"/>
    </attribute>
</define>
```

6.5.2 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 you can attach to 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 the section *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="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. Please note that you can use database field values in the expression by enclosing in square brackets, the database name, the table name, and the column name, separated by dots.

If the `text:condition` attribute is not present, OpenOffice.org assumes 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='[address book file.address.FIRSTNAME] == "Julie"'
```

This example specifies a condition that is true if the current row from the OpenOffice.org address book is the address for a person named Julie. If the condition shown in this example is used in a `<text:database-next>` element, the following happens:

- The `<text:database-display>` elements display the data from the first row of the current selection.
- If the `FIRSTNAME` column of the current row reads `Julie`, the current row is changed. Otherwise, nothing happens.
- If the first row is `Julie`, the following `<text:database-display>` elements display data from the second row. Otherwise, they display data from the first row.

See Section 3.7.43 for more information on the formula syntax of a `text:condition` attribute, which is the same as that of the `text:formula` attribute.

TODO: Find proper reference!

6.5.3 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, you can specify a condition so that the given row is only selected if the condition is `true`.

The attributes that you can associate 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 3.7.12 for a full explanation of how to use this attribute.

TODO: Find proper reference

- `text:row-number`

  See the section *Selecting the Row Number* for information 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="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">
            <data type="nonNegativeInteger"/>
        </attribute>
    </optional>
</define>
```

6.5.4 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 you can associate 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.

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

  These attributes determine how the number should be formatted. See Section 6.7.8 for more information on how to use this attribute.

- `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="field-database-table"/>
            <ref name="field-num-format"/>
            <optional>
                <attribute name="text:value">
                    <data type="nonNegativeInteger"/>
                </attribute>
            </optional>
        </interleave>
        <text/>
    </element>
</define>
```

6.5.5 Display Current Database and Table

OpenOffice.org keeps track of the last database and table that was used in the document. In other words, the table that is used by the last field that was inserted into the document. In the OpenOffice.org user interface, the database is displayed in the Beamer. The `<text:database-name>` element displays the database and table name of the most recently used table.

The attributes that you can associate 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="field-database-table"/>
        <text/>
    </element>
</define>
```

6.6 More Fields

6.6.1 Page Variable Fields

Page variables allow you to define an alternative page numbering scheme. There is only one page variable, and it is set by any set page variable field in the document. The value of the page variable is increased on each page, in the same way as regular page numbers.

Setting Page Variable Fields

To set a page variable field, you use the `<text: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. You can use the `text:active` attribute 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">
            <data type="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 you can associate with the `<text:get-page-variable>` element are:

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

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

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

6.6.2 Placeholders

OpenOffice.org Writer uses placeholder fields to indicate locations in a document where the user must fill in some information. For example in a letter template, you can have a section of the document 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 you can associate with the `<text:placeholder>` element are:

- Placeholder type
- Placeholder description

Placeholder Type

There are five different types of placeholder, representing the five possible types of content: text, tables, text boxes, images, or objects. The `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, you can associate a `text:description` attribute with the placeholder element. This attribute is optional. The purpose of the attribute is to contain a more elaborate description of the purpose of the placeholder than the description stored in the element content. See Section 6.7.4 for information on using the `text:description` attribute.

```
<define name="text-placeholder-attlist" combine="interleave">
    <ref name="field-description"/>
</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 you can associate with the `<text:conditional-text>` element are:

- Condition
- Text to display if the condition is true
- Text to display if the condition is false
- 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">
        <data type="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">
        <data type="string"/>
    </attribute>
</define>
```

Current 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 you can associate 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">
        <data type="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

Open Office XML 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 identified by a name that is usually generated automatically when a document is exported.

```
<define name="paragraph-content" combine="choice">
    <element>
        <choice>
            <name>text:reference-ref</name>
            <name>text:bookmark-ref</name>
            <name>text:note-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: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 you can associate with the reference field elements are:

- Reference name
- Reference format

Reference Name

The `text:ref-name` attribute identifies the referenced element. Since bookmarks and references have a name, this name is used by the respective reference fields. Footnotes, endnotes, and sequences are assigned names by the application used to create the OpenOffice.org XML file format when the document is exported.

```
<define name="text-common-ref-content" combine="interleave">
    <optional>
        <attribute name="text:ref-name">
            <data type="string"/>
        </attribute>
    </optional>
</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`	78
`chapter`	3.7.27

Reference format	Reference display
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. You can use the field 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 HTML, so that the content of a `<script>` element in HTML can be imported, edited, and exported using the OpenOffice.org 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 `text:href` attribute to specify the location of the source file.

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

```
<define name="paragraph-content" combine="choice">
    <element name="text:script">
        <interleave>
            <choice>
                <attribute name="text:href">
                    <ref name="URL"/>
                </attribute>
                <text/>
            </choice>
            <optional>
                <attribute name="script:language">
                    <data type="string"/>
                </attribute>
            </optional>
        </interleave>
    </element>
</define>
```

Script URL

The `text: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.

TODO: Why not an XLink? We use XLink everywhere else!

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 attribute that you can associate 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">
                <data type="string"/>
            </attribute>
        </optional>
        <optional>
            <ref name="office-events"/>
        </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 you to hide or display a paragraph of formatted text, depending on whether a condition is `true` or `false`.

Hidden paragraph fields are often used together with form letters. For example, if a condition depends on a database field, a hidden paragraph field can be used to selectively include paragraphs in the form letter depending on the database content. Multiple paragraph fields can be contained one paragraph. The paragraph is displayed if the condition associated with at least one hidden paragraph field is `false`. Alternatively, you can combine the conditions associated with several hidden paragraph fields into a single condition for a single field using logical operations on the conditions.

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

The attributes that you can associate 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 you to display information from a DDE connection. The only parameter required for the DDE field is the name of the DDE connection that supplies the data to this field. This DDE connection element specifies the actual DDE field that appears in the text body.

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.

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

The attribute that you can associate with the <text:dde-connection> element is:

- DDE connection name

DDE Connection Name

The text:name attribute specifies the name of the DDE connection to which the field refers.

6.6.10Measure Fields

Information to be supplied.

```
<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 version of StarWriter. It should not be used in new documents. It stores a formula to eb 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="field-formula"/>
            <ref name="field-display-value-formula"/>
            <ref name="field-data-style-name"/>
        </interleave>
        <text/>
    </element>
</define>
```

6.7 Common Field Attributes

You can use the attributes described in this section 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 `text:value-type` attribute.

```
<define name="field-value-type">
    <attribute name="text:value-type">
        <choice>
            <value>float</value>
            <value>time</value>
            <value>date</value>
            <value>percentage</value>
            <value>currency</value>
            <value>boolean</value>
            <value>string</value>
        </choice>
    </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`	`text:value`	Numeric value	`"12.345"`
`percentage`	`text:value`	Numeric value	`"0.50"`
`currency`	`text:value` and `text:currency`	Numeric value and currency symbol	`"100"` `"USD"`
`date`	`text:date-value`	[ISO8601] §5.2.1.1, extended format	`"2003-04-17"`
`time`	`text:time-value`	[ISO8601] §5.4.1 a), extended format	`"PT03H30M00S"`
`boolean`	`text:boolean-value`	`true` or `false`	`"true"`
`string`	`text:string-value`	Strings	`"abc def"`

The Open Office XML concept of field values and value types and their encoding in XML is modeled on the corresponding XML for table cell attributes. See Section XXX for more detailed information on these attributes.

TODO: Supply proper reference

The definition of the entity `%value-attlist;` is as follows:

```
<define name="field-value-and-type">
    <choice>
        <group>
            <attribute name="text:value-type">
                <value>float</value>
            </attribute>
            <attribute name="text:value">
                <data type="double"/>
            </attribute>
        </group>
        <group>
            <attribute name="text:value-type">
```

```
                <value>percentage</value>
            </attribute>
            <attribute name="text:value">
                <data type="double"/>
            </attribute>
        </group>
        <group>
            <attribute name="text:value-type">
                <value>currency</value>
            </attribute>
            <attribute name="text:value">
                <data type="double"/>
            </attribute>
            <optional>
                <attribute name="text:currency">
                    <data type="string"/>
                </attribute>
            </optional>
        </group>
        <group>
            <attribute name="text:value-type">
                <value>date</value>
            </attribute>
            <attribute name="text:date-value">
                <data type="date"/>
            </attribute>
        </group>
        <group>
            <attribute name="text:value-type">
                <value>time</value>
            </attribute>
            <attribute name="text:time-value">
                <data type="duration"/>
            </attribute>
        </group>
        <group>
            <attribute name="text:value-type">
                <value>boolean</value>
            </attribute>
            <attribute name="text:boolean-value">
                <ref name="boolean"/>
            </attribute>
        </group>
        <group>
            <attribute name="text:value-type">
                <value>string</value>
            </attribute>
            <optional>
                <attribute name="text:string-value">
                    <data type="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="field-fixed">
    <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 you are declaring, setting, or displaying a variable. You can use this attribute with any of the following elements:

- `<text:variable-decl>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-decl>`
- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:sequence-decl>`
- `<text:sequence>`

When you are using this attribute to specify the name of a variable to display, a variable of the appropriate type with the same name must already have been declared.

```
<define name="field-name">
    <attribute name="text:name">
        <ref name="variable-name"/>
    </attribute>
</define>
```

6.7.4 Description

The `text:description` attribute contains a brief message that is displayed when users are prompted for input. You can use this attribute with any of the following elements:

- `<text:placeholder>`
- `<text:variable-input>`
- `<text:user-field-input>`
- `<text:text-input>`

```
<define name="field-description">
    <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 you to display the formula rather than the value of the field. Some fields do not support this value. In these cases, the `text:display` attribute only takes the values `value` or `none`, and `value` or `formula`, respectively.

- `none`
  Several variable fields support this value, which hides the field content. This allows you to set variables in one part of the document and display them in another part of the document.

You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-get>`
- `<text:expression>`

```
<define name="field-display-value-none">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>value</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="field-display-value-formula-none">
    <optional>
        <attribute name="text:display">
            <choice>
                <value>value</value>
                <value>formula</value>
                <value>none</value>
            </choice>
        </attribute>
    </optional>
</define>
<define name="field-display-value-formula">
    <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. You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:user-field-decl>`
- `<text:sequence>`
- `<text:expression>`

```
<define name="field-formula">
    <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 1. For more information about data styles, see Chapter 2.

For string variables you must omit this attribute. Otherwise, this attribute is required.

The name must match the name of a data style.

You can use this attribute with any of the following elements:

- `<text:date>`
- `<text:time>`
- `<text:page-number>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:expression>`

```
<define name="field-data-style-name">
    <optional>
        <attribute name="style:data-style-name">
            <ref name="styleName"/>
        </attribute>
    </optional>
</define>
```

6.7.8 Number Formatting Style

You can format numbers that are used for number sequences such as page numbers or sequence fields according to the number styles described in Chapter 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 XSL 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 Open Office 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 Chapter 2 for more information.

TODO: provide proper references

```
<define name="field-num-format">
    <choice>
        <attribute name="style:num-format">
            <choice>
                <value>1</value>
                <value>i</value>
                <value>I</value>
            </choice>
        </attribute>
        <group>
            <attribute name="style:num-format">
                <choice>
                    <value>a</value>
                    <value>A</value>
                </choice>
            </attribute>
            <optional>
                <attribute name="style:num-letter-sync">
                    <ref name="boolean"/>
                </attribute>
            </optional>
        </group>
        <empty/>
    </choice>
</define>
```

7 Indices

TODO: copy index description from OOo spec

```
<define name="text-table-of-content">
    <element name="text:table-of-content">
        <ref name="anyAttrs"/>
        <element name="text:table-of-content-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-illustration-index">
    <element name="text:illustration-index">
        <ref name="anyAttrs"/>
        <element name="text:illustration-index-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-table-index">
    <element name="text:table-index">
        <ref name="anyAttrs"/>
        <element name="text:table-index-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-object-index">
    <element name="text:object-index">
        <ref name="anyAttrs"/>
        <element name="text:object-index-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-user-index">
    <element name="text:user-index">
        <ref name="anyAttrs"/>
        <element name="text:user-index-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-alphabetical-index">
    <element name="text:alphabetical-index">
        <ref name="anyAttrs"/>
        <element name="text:alphabetical-index-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
```

```
</define>

<define name="text-bibliography">
    <element name="text:bibliography">
        <ref name="anyAttrs"/>
        <element name="text:bibliography-source">
            <ref name="anything"/>
        </element>
        <ref name="text-index-body"/>
    </element>
</define>

<define name="text-index-body">
    <element name="text:index-body">
        <ref name="anything"/>
    </element>
</define>
```

8 Tables

```
<define name="table-table">
    <element name="table:table">
        <ref name="anything"/>
    </element>
</define>
```

9 Chart

10 UI

11 Dialog

12 Packages

13 Form Content

TODO: add form content

```
<define name="office-forms">
    <optional>
        <element name="office:forms">
            <ref name="anything"/>
        </element>
    </optional>
</define>
```

14 Graphic Content

TODO: add graphical content

```
<define name="draw-a">
    <element name="draw:a">
        <ref name="anything"/>
    </element>
</define>
<define name="shape">
    <element>
        <choice>
            <name>draw:rect</name>
            <name>draw:line</name>
            <name>draw:polyline</name>
            <name>draw:polygon</name>
            <name>draw:path</name>
            <name>draw:circle</name>
            <name>draw:ellipse</name>
            <name>draw:g</name>
            <name>draw:page-thumbnail</name>
            <name>draw:text-box</name>
            <name>draw:image</name>
            <name>draw:object</name>
            <name>draw:object-ole</name>
            <name>draw:applet</name>
            <name>draw:floating-frame</name>
            <name>draw:plugin</name>
            <name>draw:measure</name>
            <name>draw:caption</name>
            <name>draw:connector</name>
            <name>chart:chart</name>
            <name>dr3d:scene</name>
            <name>draw:control</name>
            <name>draw:a</name>
        </choice>
        <ref name="anything"/>
    </element>
</define>
```

15 Other and Common Content

TODO: This chapter contains content elements which have not yet been placed elsewhere

15.1 Annotation

The `<office:annotation>` element specifies an Open Office annotation.

```
<define name="office-annotation">
    <element name="office:annotation">
        <ref name="office-annotation-info"/>
        <zeroOrMore>
            <ref name="text-p"/>
        </zeroOrMore>
    </element>
</define>
```

TODO: It seems more consistent with the rest of the spec if the annotation element would have meta-data element children; that's what we do everywhere else

TODO: The base spec allows only text:p element as children. Is that what we want? Do we want arbitrary text as children, or do we want only rather simple text?

TODO: In the base spec, the author/date attributes were declared #IMPLIED, but a comment said they were required. I just dropped the required here. Is that what we want?

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

- Author
- Creation date
- Display

Author

The `office:author` attribute specifies the author of the annotation.

```
<define name="office-annotation-info" combine="interleave">
    <optional>
        <attribute name="office:author">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

Creation Date

The `office:create-date` attribute specifies the creation date and time of the annotation. If the application only has a date string and cannot parse this string, it must write the string to the `office:create-date-string` attribute.

```
<define name="office-annotation-info" combine="interleave">
    <optional>
        <attribute name="office:create-date">
            <ref name="dateTime"/>
        </attribute>
    </optional>
</define>
<define name="office-annotation-info" combine="interleave">
    <optional>
        <attribute name="office:create-date-string">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

Display

The `office:display` attribute specifies whether or not the annotation is visible.

```
<define name="office-annotation-info" combine="interleave">
    <optional>
        <attribute name="office:display">
            <ref name="boolean"/>
        </attribute>
    </optional>
</define>
```

15.2 Events

```
<define name="office-events">
    <element name="office:events">
        <ref name="anything"/>
    </element>
</define>
```

15.3 Mathematical Content

Mathematical content is represented by MathML 1.01.

```
<!-- to avoid inclusion of the complete MathML schema, we will simply allow
     anything within a math:math top-level element
-->
<define name="math-math">
    <element name="math:math">
        <ref name="anything"/>
    </element>
</define>
```

TODO: Do we want to fix this to a particular version of MathML?

15.4 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.

15.4.1Container for DDE Connection Declarations

The DDE connection declarations are contained in one declarations element, .<text:dde-connection-decls>.

15.4.2Declaring DDE Connections

Every DDE connection is declared using a declaration element. Multiple DDE fields can refer to one DDE connection by using the same name. The declaration element has no content.

```
<define name="text-dde-connection-decl">
    <element name="text:dde-connection-decl">
        <ref name="text-dde-connection-decl-attlist"/>
    </element>
</define>
```

The attributes that you can associate with the <text:dde-connection-decl> element are:

- Connection name
- DDE target application
- DDE target file name
- DDE command
- Automatic update flag

Connection Name

The text:name attribute specifies the name by which the connection will be referred.

```
<define name="text-dde-connection-decl-attlist" combine="interleave">
    <attribute name="text:name">
        <data type="string"/>
    </attribute>
</define>
```

TODO: Shouldn't this be office:name?

Target Application

The text:dde-application attribute specifies the name of the target application to use for the DDE connection.

```
<define name="text-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-application">
        <data type="string"/>
    </attribute>
</define>
```

Note: The target name for OpenOffice.org is soffice. Therefore, internal DDE links have the attribute text:dde-application="soffice".

Target Topic

The text:dde-topic attribute specifies the name of the topic to use for the DDE connection.

```
<define name="text-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-topic">
```

```
        <data type="string"/>
    </attribute>
</define>
```

> **Note**: If the target application is OpenOffice.org, it interprets the DDE topic as the name of the file.

Target Item

The `text:dde-item` attribute specifies which information the target application should deliver.

```
<define name="text-dde-connection-decl-attlist" combine="interleave">
    <attribute name="office:dde-item">
        <data type="string"/>
    </attribute>
</define>
```

> **Note**: If the target application for the DDE connection is OpenOffice.org Writer, the item represents the name of a bookmark. OpenOffice.org delivers the current text content to the requesting application.

Automatic Update

OpenOffice.org Writer can automatically update DDE links. If preferred, you can use the `text:automatic-update` attribute to specify that the DDE connection links should only be updated at the request of the user.

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="text-dde-connection-decl-attlist" combine="interleave">
    <optional>
        <attribute name="office:automatic-update">
            <data type="string"/>
        </attribute>
    </optional>
</define>
```

16 Styles and Classes

```
<define name="class-attrs">
    <optional>
        <attribute name="text:style-name">
            <text/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:class-name">
<!-- what is the actual type of the class attributes? They contain a sequence of reference, right? -->
            <text/>
        </attribute>
    </optional>
    <optional>
        <attribute name="text:cond-style-name">
            <text/>
        </attribute>
    </optional>
</define>
```

16.1 Other: Text Styles

3.4 Outline Numbering 140

3.5 Line Numbering 142

3.10 Text Formatting Properties 203

3.11 Paragraph Formatting Properties 216

3.12 Section Formatting Properties 229

3.1.5 Text Styles 121

Text Formatting Properties 121

17 TODO

TODO: fo:left/right-margin, fo: (Ken Holman)

TODO: database-name element and attribute: rename element to database-display-name or something

TODO: document: DDE connections not needed to read documents. content stored in elements.

```
<!--
the content definitions from the base format: Here only for reference while
editing; this is to be deleted for the final spec.
<define name="office-text-prelude">
    <optional>
        <ref name="office.forms"/>
    </optional>
    <optional>
        <choice>
            <ref name="text.tracked-changes"/>
            <ref name="table.tracked-changes"/>
        </choice>
    </optional>
    <ref name="text-decls"/>
    <optional>
        <ref name="table.calculation-settings"/>
    </optional>
    <optional>
        <ref name="table.content-validations"/>
    </optional>
    <optional>
        <ref name="table.label-ranges"/>
    </optional>
</define>
<define name="office-text-content">
    <zeroOrMore>
        <choice>
            <ref name="text.h"/>
            <ref name="text.p"/>
            <ref name="text.ordered-list"/>
            <ref name="text.unordered-list"/>
            <ref name="table.table"/>
            <ref name="draw.page"/>
            <ref name="draw.a"/>
            <ref name="shape"/>
            <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="change-marks"/>
        </choice>
    </zeroOrMore>
</define>
<define name="office-text-epilogue">
    <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>
    <optional>
        <ref name="presentation.settings"/>
    </optional>
</define>
-->
```

Appendix A.Appendix

A.1. Used Datatypes

The following data types are used within this specification:

- W3C Schema data types (referenced by `<data>` elements)
  - string
  - date
  - time
  - duration
  - nonNegativeInteger
  - positiveInteger
  - double
  - integer
- custom data types (referenced by `<ref>` elements)
  - boolean

    A boolean value may have either of the values `true` or `false`.
  - dateTime

    A dateTime value is essentially an ISO 8601 date and time value with an optional time component. In other words, it may contain either a date, or a date and time value.

Relax-NG definitions for the custom data types:

```
<define name="boolean">
    <choice>
        <value>true</value>
        <value>false</value>
    </choice>
</define>
<define name="dateTime">
    <choice>
        <data type="date"/>
        <data type="dateTime"/>
    </choice>
</define>
```

A.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="foreignAttribute">
```

```
    <attribute>
        <anyName/>
        <text/>
    </attribute>
</define>
<define name="foreignElement">
    <element>
        <anyName/>
        <zeroOrMore>
            <choice>
                <ref name="foreignAttribute"/>
                <ref name="foreignElement"/>
                <text/>
            </choice>
        </zeroOrMore>
    </element>
</define>
```

A.3. References

[xml-names] Tim Bray, Dave Hollander, Andrew Layman, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names/, W3C REC-xml-names, 1999.

[ISO8601] International Organization for Standardization, *Data elements and interchange formats - Information interchange - Representation of dates and times*, ftp://ftp.qsl.net/pub/g1smd/8601.pdf, ISO 8601, 1998.

[XLink] Steve DeRose, Eve Maler, David Orchard, *XML Linking Language*, http://www.w3c.org/TR/xlink/, W3C XLink, 2001.

A.4. *Relax-NG Schema Postfix*

Postfix for the normative Relax-NG schema:

```


<!-- these should be replaced by proper data types along the way -->
<define name="targetFrameName"><text/></define>
<define name="uriReference"><text/></define>
<define name="cLanguage"><text/></define>
<define name="styleName"><text/></define>
<define name="formula"><text/></define>
<define name="variable-name"><text/></define>
<define name="character"><text/></define>
<define name="URL"><text/></define>

<!-- anything, anyContent: use as a placeholder for items not yet defined. The
     committee draft specification should not contain any references to
     'anything' and 'anyContent' any more.
 -->
<define name="anything">
    <zeroOrMore>
        <choice>
            <attribute>
                <anyName/>
                <text/>
            </attribute>
            <element>
                <anyName/>
                <ref name="anything"/>
            </element>
            <text/>
        </choice>
    </zeroOrMore>
</define>
<define name="anyElement">
    <element>
        <anyName/>
        <ref name="anything"/>
    </element>
</define>
<define name="anyContent">
    <zeroOrMore>
        <choice>
            <element>
                <anyName/>
                <ref name="anything"/>
            </element>
            <text/>
        </choice>
    </zeroOrMore>
</define>
<define name="anyAttrs">
    <zeroOrMore>
        <attribute>
            <anyName/>
            <text/>
        </attribute>
    </zeroOrMore>
</define>


</grammar>
```

Appendix B.Notices